Saibot: A Differentially Private Data Search Platform

Zezhou Huang
zh2408@columbia.edu
Columbia University

Jiaxiang Liu
jl6235@columbia.edu
Columbia University

Daniel Gbenga Alabi
alabid@cs.columbia.edu
Columbia University

Raul Castro Fernandez
raulcf@uchicago.edu
University of Chicago

Eugene Wu
ewu@cs.columbia.edu
DSI, Columbia University

ABSTRACT

Recent data search platforms rely on ML task-based utility measures rather than metadata-based keyword search to search among a large corpus of datasets. Users submit a training dataset and these platforms search for *augmentations*—join or union-compatible datasets—that, when used to augment the user's dataset, most improve model (e.g., linear regression) performance. Although effective, providers that manage personally identifiable data demand differential privacy (DP) guarantees before allow these platforms access to their data. Unfortunately, making data search differentially private is nontrivial, as a single search can involve training and evaluating datasets hundreds or thousands of times, and quickly deplete available privacy budgets.

We present *Saibot*, a differentially private data search platform that employs a new DP mechanism called FDP, which calculates sufficient semi-ring statistics to support ML models over different combinations of datasets. These factorized sufficient statistics are privatized once by the data provider, and can be freely reused by the platform. This allows *Saibot* to scale to arbitrary numbers of datasets in the corpus and numbers of requests, while minimizing the amount that DP noise affects search results. We optimize the sensitivity of FDP for commmon augmentation operations, and analyze its properties with respect to linear regression models. Specifically, we develop an unbiased estimator for many-to-many joins, prove its bounds, and develop and optimization to redistribute DP noise to minimize the impact on the model. Our evaluation on a real-world dataset corpus of 329 datasets demonstrates that *Saibot* can return augmentations that achieve model accuracy within 50−90% of non-private search, while the leading alternative DP mechanisms (LDP, GDP, SDP) are several orders of magnitude worse.

PVLDB Reference Format:
Zezhou Huang, Jiaxiang Liu, Daniel Gbenga Alabi, Raul Castro Fernandez, and Eugene Wu. Saibot: A Differentially Private Data Search Platform. PVLDB, 14(1): XXX-XXX, 2020.
doi:XX.XX/XXX.XX

1 INTRODUCTION

The ability to augment training data with more samples or features can greatly improve ML performance [48]. However, finding such data in large corpora—public portals [10, 11], enterprise data warehouses or lakes—is a nontrivial challenge. To address this, a new form of data search platform [17, 41, 43, 49, 62] is emerging where a user submits a search request consisting of training and testing datasets for augmentation. The platform then finds provider datasets that augment the training dataset in a way that improves prediction accuracy. To do so, platforms use a data discovery tool [15, 27] to find a candidate set of union- or join-compatible tables (*augmentations*); they augment the training set with each candidate, retrain and evaluate the model to assess its *utility*, and rank the augmentations by *utility*. Platforms largely vary in the discovery tool procedure, the models they support, and how they accelerate model retraining and evaluation. Recent works [16, 49, 62] suggest that using linear regression as a proxy for the model provides a good balance of search quality and search runtime.

Unfortunately, privacy is a major barrier to sharing for many potential data providers and users with sensitive data (e.g., personally identifiable information (PII), and protected health information (PHI)). In these cases, providers are required by law to prevent personal data leakage [6, 7, 9]. Rather than prohibit access outright, differential privacy (DP) [20] supports data analysis over sensitive data while bounding the degree of privacy loss based on the budget ϵ set by the data provider. Every time the dataset is queried, it adds noise to the results in inverse proportion to the amount of budget it consumes; once $\epsilon = 0$, the dataset cannot be accessed.

Ideally, a differentially private data search platform would let providers and users set privacy budgets on their datasets, and enforce these budgets as new datasets and requests arrive. In addition, since the platform is often a third-party service that data providers (and the individuals that they collect data from) may not trust, it should not have access to raw data. Unfortunately, integrating DP with data search platforms is non-trivial. To illustrate, Figure 1 summarizes where existing DP mechanisms would add noise in a two level data-sharing architecture that matches many real-world settings. Individuals (e.g., patients) generate sensitive data aggregated by providers/users (e.g., hospitals), and the search platform further aggregates their datasets.

Global DP (GDP) adds noise after executing e.g., a query over private data. It is widely used by private DBMSes [34, 38, 59], however it needs to "split the budget" across every dataset for every request. The budget ends up being so small that the noise drowns any signal in the data. Further, GDP runs in the platform, and thus requires it to have access to raw data. A common optimization is to apply DP to an intermediate, so it can be freely reused with no additional privacy cost. Local DP (LDP) applies DP to individual tuples (e.g., via randomized response [18, 25]), but adds so much noise that it can cripple utility [58]. Shuffle DP (SDP) [24, 26] introduces an additional step after applying LDP that shuffles the primary keys of tuples at the point of aggregation to dissociate their relationship with an individual; this "amplifies privacy" so that LDP can apply less noise. Variation SDP-1 runs at the provider/user, but still requires considerable noise when the datasets are small; SDP-2 runs

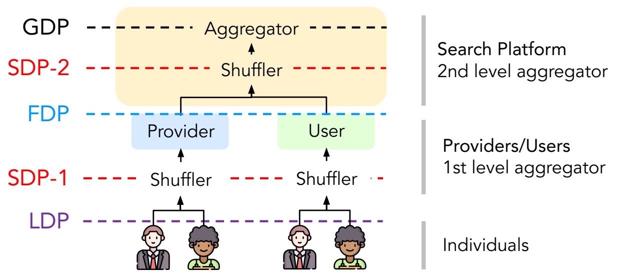


Figure 1: Summary of DP-approaches in a typical data sharing architecture. Providers/users aggregate data from individuals; the search platform aggregates data from providers/users. At the extremes, `LDP` adds noise to individual tuples, while `GDP` adds noise to query results in a trusted search aggregator. `SDP` enhances `LDP` using a shuffler at a point of aggregation: either at the first (`SDP-1`) or second levels (`SDP-2`). `FDP` adds noise to sufficient statistics computed by providers/users.

in the search platform, but needs to trust the platform. An alternative to `SDP`, widely used by federated ML [52, 53, 58, 63], is to let providers/users compute and add `DP` to model gradients locally, and let an untrusted aggregator compute the final model. However, these gradients are specific to a single augmentation's model, so the budget is still split across all candidate augmentations.

Is it is possible for a DP search platform to return search results of comparable quality to non-private search, *and* for the platform to scale to many datasets and requests? We are motivated by the recent data search platform Kitana [62], which uses semi-ring aggregation to quickly evaluate a candidate augmentation's *utility* on a linear regression model without materializing the augmented table and fully retraining the model. These semi-rings can be computed for each dataset offline, and Kitana only needs these semi-rings to evaluate a candidate augmentation in $\approx 1-5ms$, independent of the dataset size. *Our main observation is that these precomputed semi-rings also serve as ideal intermediates for* `DP`.

This paper presents *Saibot*, a differentially private data search platform for tabular datasets that is scalable to unlimited datasets and requests, can return results comparable to non-private search, and doesn't need to be trusted. Data providers upload their privatized datasets to the platform. When a user submits a privatized training dataset, the platform searches for the most best combinations of privatized datasets which, when augmented with the user's dataset, most improve the accuracy of a linear regression model. The privatized datasets can be directly returned to users for ML, and users can contact the providers to negotiate access to the raw datasets (e.g., following Health Information Exchange guidance).

Our key innovation is a new `DP` mechanism called Factorized Differential Privacy (`FDP`), where each user or provider computes and privatizes sufficient statistics on their own datasets based on their privacy requirements. These sufficient statistics provide high utility, can be freely reused for models over different augmentations, and only require the search platform to store privatized datasets. Its threat model sits between `GDP` and `LDP`. Unlike `GDP`, the second level aggregators (i.e. search platform) are considered as external entities that can be malicious. Unlike `LDP`, `FDP` assumes that the first-level aggregators (providers/users) are not malicious. In practice, first-level aggregators as the direct data governors (e.g., healthcare providers, schools) have been required by regulations [6–9] to securely store individual data.

The main algorithmic challenge `FDP` solves is to design privatized sufficient statistics for ML that are composable to support various join and union augmentations. Previous works have applied `DP` to sufficient statitics for privitizied linear regression [57] and GLM [33, 39], but these sufficient statistics can be used for only a single dataset. Our key insight is to design these sufficient statistics as a semi-ring [29], which includes addition and multiplication operators for union and join. Although sufficient semi-ring statistics have been utilized for ML [50, 51] over joins, we are the first to explore their application in a `DP` setting. The results of our real-world experiments indicate that *Saibot* is capable of identifying augmentations that achieve an average $r2$ score of $\sim 50-90\%$ compared to non-private searches. Additionally, *Saibot* can support a large data corpus and unlimited user requests. In contrast, the other baseline mechanisms achieve $r2$ scores <0.02.

To summarize, our contributions are as follows:

- We design *Saibot*, a differentially private dataset search platform that scales to large volumes of datasets and requests. *Saibot* uses Factorized `DP` (`FDP`), a novel privacy mechanism, to ensure differentially private data storage, data discovery, and data search.
- We optimize `FDP` based on the parity of the statistics order. For the special case of tables containing a single feature, we reduce the expected error by a further factor of $\sqrt{2}$, which composes well with dimensionality reduction (Section 3.5).
- We provide a deep analysis of `FDP` to linear regression models. Specifically, we study the statistical bias introduced in many-to-many joins, and design an unbiased estimator to address this. We further study its bounds on errors over the model parameters.
- We design an optimization that carefully redistributes noise across the sufficient statistics to imrpove model accuracy.
- We thoroughly evaluate `FDP` against `GDP`, `LDP`, and `SDP` across a real-world data corpus with >300 datasets. Our results show that `FDP` can accurately identify augmentations that achieve $r2$ scores close to ($\sim 50-90\%$) those of a non-private search. We further use ablation studies to validate our theoretical analyses and study the sensitivity to different parameters.

2 PRIVATE TASK-BASED DATA SEARCH

In this section, we formalize the problem of task-based private data search. We start with the introduction of non-private problem and current solutions. We then provide the primer of differential privacy, and present the differentially private data search problem.

2.1 Non-Private Task-based Data Search

We provide the background of previous task-based data search problem [17, 41, 43, 49], which is non-private, and previous solutions.

Data Model. We adhere to the traditional relational data model, using R to denote relations, A to denote attributes, and $dom(A)$ to denote their respective domains. The schema of a relation is expressed by $S_R = [A_1, \cdots, A_n]$, with tuples denoted as t and attribute values by $t[A]$. For clarity, the schema is enclosed in square brackets after the relation $R[A_1, \cdots, A_n]$ in examples. Finally, the domain of a relation is the cartesian product of its attribute domains: $dom(R) = dom(A_1) \times \cdots \times dom(A_n)$. We assume that each dataset is a relational table, and use the terms interchangeably.

Machine Learning. A ML task M, such as linear regression or logistic regression, aims to fit a good model to predict target attribute based on features. Given a training dataset R_{train} consists of features $X \subset S_R$ and a target attribute $y \in S_R$, M has a training function $M.Train(\cdot)$ that takes a relation R_{train} as input, and outputs a model m; m transforms X in a way that best predicts y, even when the specific X, y pairs have not been seen before. To evaluate m, M has a function $M.Evaluate(\cdot)$ that takes m and a testing dataset R_{test} as input and outputs the performance of the model on R_{test}, such as accuracy, that needs to be maximized.

Task-based Data Search. Given a data corpus with datasets from different providers, users send a request with datasets to augment and a task (e.g., ML). Task-based data search aims to identify a set of augmentable (join/union) datasets that maximize task utility.

To formalize this, let $\mathcal{R} = \{R_1, R_2, ...\}$ be a data corpus with a set of relations, where each is from some provider. User sends a request of training and testing dataset (R_{train}, R_{test}), and choose a model M. User's goal is to train a model M on R_{train} and maximize its performance on R_{test}, which we call the task's *utility*.

To improve the *utility*, the user wants to find a set of provider datasets in $\mathcal{R}$ that can be used to augment their data and enhance model performance. The function $Discover(R, augType)$ is used to find datasets in the data corpus $\mathcal{R}$ that can be joined or unioned with R, given a dataset R and $augType \in \{\bowtie, \cup\}$. The user wants to try different combinations of subsets of these datasets to augment[1] and find the combination that maximizes *utility*.

Putting everything together, the problem can be formulated as:

Problem 1 (Task-Based Data Search.). *For each user request (R_{train}, R_{test}, M), find the set of augmentable provider datasets $\mathbf{R}^*_\cup, \mathbf{R}^*_\bowtie \subseteq \mathcal{R}$ from data corpus such that*

$$\begin{aligned}
\mathbf{R}^*_\cup, \mathbf{R}^*_\bowtie = \quad & \underset{\mathbf{R}_\cup, \mathbf{R}_\bowtie}{\operatorname{argmax}} M.Evaluate(m, R_{testAug}) \\
s.t. \quad & \mathbf{R}_\cup \subseteq Discover(R, \cup), \mathbf{R}_\bowtie \subseteq Discover(R, \bowtie), \\
& R_{trainAug} = (R_{train} \cup_{R_1 \in \mathbf{R}_\cup} R_1) \bowtie_{R_2 \in \mathbf{R}_\bowtie} R_2 \\
& R_{testAug} = (R_{test} \cup_{R_1 \in \mathbf{R}_\cup} R_1) \bowtie_{R_2 \in \mathbf{R}_\bowtie} R_2 \\
& m = M.Train(R_{trainAug})
\end{aligned}$$

Solutions. The current task-based data search platforms [17, 41, 43, 49] follow the architecture as illustrated in black in Figure 2. Offline, when providers upload raw datasets to *Data storage*, the repo (or provider) computes minhashes of tables and columns for data discovery [15, 27], and sketches to accelerate retraining [17, 43, 49, 62]. Online, the platform solves Problem 1 for each user request (R_{train}, R_{test}, M). First, *data discovery* [15, 27] uses the minhashes or sketches to return a set of candidate datasets. *Data search* then identifies a subset that maximizes task utility. The brute-force search evaluates all possible combinations and can be expensive due to retraining costs and the large set of combinations, so approaches use various heuristics and greedy algorithms [17, 41, 49].

The major instance of this solution that our work builds upon is Kitana [62]. Kitana follows the architecture in Figure 2 and employs specialized sketches for factorized ML; factorized ML trains models over joins without materializing them, thus accelerating the model retraining and evaluation after any candidate augmentation. This enables Kitana to perform task-based search orders of magnitude faster, with competitive task utility. Our insight is that these sketches not only enhance performance but also serve as the ideal sufficient statistics for DP, as discussed in Section 3.2.

[1]For simplicity, we consider datasets that can be directly joined or unioned with user R_{train}. The search space could be further expanded by, e.g., first joining provider datasets; our solution can be easily adapted to this larger search space.

2.2 Differential Privacy Primer

Before we introduce our solution to differentially private dataset search, we need to first introduce differential privacy (DP). Our paper will focus on the Gaussian mechanism, a widely used and easy-to-implement technique that can be compared with other baselines, (e.g., it can offer the same approximate DP by SDP). In practice, our solution can support pure DP by Laplace mechanism as shown in Section 5.2 (where SDP falls short).

Differential Privacy. DP [21] is a technique used to protect reconstruction, membership, and inference attacks [23] by bounding the information leakage from individual records. DP guarantees that the probability that an algorithm will produce the same output on two datasets that differ by only one record is bounded. Formally:

Definition 1 ($(\epsilon, \delta) - DP$). *Let f be a randomized algorithm that takes a relation R as input. f is $(\epsilon, \delta) - DP$ if, for all relations R_1, R_2 that differ by adding or removing a row, and for every set S of outputs from f, the following holds:*

$$Pr[f(R_1) \in S] \leq e^\epsilon Pr[f(R_2) \in S] + \delta$$

where ϵ and δ are non-negative real numbers (called privacy budget). ϵ controls the level of privacy, and δ controls the level of approximation. For the special case when $\delta = 0$, $(\epsilon, 0) - DP$ is also called pure DP.

DP algorithms can be global (GDP) or local (LDP) depending on inputs. While GDP algorithms take a relation as input to privatize, as discussed above, LDP algorithms take each tuple (or a relation with a cardinality of 1) as input.

There are three important theorems of DP:

Theorem 1 (Robustness to Post-Processing). *Let f be a randomized algorithm that provides $(\epsilon, \delta) - DP$. Let g be an arbitrary function. Then, the composition $g \circ f$ provides $(\epsilon, \delta) - DP$.*

Theorem 2 (Sequential Composition). *Let $f_1, \ldots, f_n$ be a sequence of independent algorithms that provide $(\epsilon_1, \delta_1), \ldots, (\epsilon_n, \delta_n) - DP$, respectively. Then, the algorithm that applies each of them in sequence, i.e., $f_n \circ f_{n-1} \cdots \circ f_1$, is $(\sum_{i=1}^n \epsilon_i, \sum_{i=1}^n \delta_i) - DP$.*

Theorem 3 (Parallel Composition). *Let $dom_1, \ldots, dom_n$ be n disjoint subsets of $dom(R)$. Let $f_1, \ldots, f_n$ be a set of independent algorithms that provide $(\epsilon_1, \delta_1), \ldots, (\epsilon_n, \delta_n) - DP$ and take relations from $dom_1, \ldots, dom_n$ as input, respectively. Then, the algorithm that applies them on disjoint subsets of R is $(\max_{i=1}^n \epsilon_i, \max_{i=1}^n \delta_i) - DP$.*

To ensure $(\epsilon, \delta) - DP$ when Q queries need to be executed, the privacy budget (ϵ, δ) can be split among the queries using sequential composition, such as allocating $(\epsilon/Q, \delta/Q)$ for each query. This

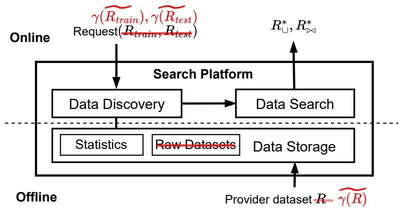

$\gamma(\widetilde{R_{train}}), \gamma(\widetilde{R_{test}})$
Online
Request(~~R_{train}, R_{test}~~)
$R^*_{\cup}, R^*_{\bowtie}$
Search Platform
Data Discovery
Data Search
Statistics
~~Raw Datasets~~
Data Storage
Offline
Provider dataset ~~R~~ $\widetilde{\gamma(R)}$

Figure 2: ***Saibot*** **architecture. Previous data search platforms (black) store raw datasets in data storage, use data discovery to identify augmentable datasets, and search the datasets for task improvement . To ensure privacy,** ***Saibot*** **additionally applies FDP (red) to compute sufficient semi-ring statistics, that are aggregated (γ) and privatized ($\sim$) before being sent to the search platform. These statistics can support join and union queries to train and evaluate ML as post-processing.**

work employs (basic) sequential composition for simplicity, but it could be further optimized by advanced composition [22].

Gaussian Mechanism. The Gaussian mechanism [19] adds noise to a query function to satisfy (ϵ, δ)-differential privacy. Formally:

Theorem 4 (Gaussian Mechanism.). *Given $\epsilon, \delta \in (0, 1]$, let query q be a function that takes R as input and outputs a vector of real numbers. The Gaussian mechanism independently adds random noise to each output to satisfy (ϵ, δ)-differential privacy:*

$$q'(R) = q(R) + \mathcal{N}(0, \sigma^2)$$

where $\mathcal{N}(0, \sigma^2)$ denotes a Gaussian distribution with mean 0 and standard deviation $\sigma = \sqrt{2\ln(1.25/\delta)}\Delta_q/\epsilon$. Δ_q is the l_2-sensitivity of q defined as: for all possible neighbouring relations R_1, R_2, Δ_q is the maximum l_2 distance of q outputs $\|q(R_1) - q(R_2)\|_2$.

There are different definitions of neighbouring relations. We adopt bounded DP [13], where neighbouring relations R_1, R_2 have the same number of rows, but the data in one row differ; our systems can be easily extended for unbounded DP.

2.3 Private Task-based Data Search

We first lay out the privacy requirements based on the criteria (Section 1) and motivated by real-world use cases. Then, we define the differentially private data search problem, and discuss the challenges and the intuition for solutions.

Privacy Requirements. We adopt a standard two-level aggregator setting (Figure 1): the first level aggregators are providers/users (e.g., hospitals, schools), and the second level aggregator is the search platform. An individual shares data with their direct first-level aggregator, who is not malicious (e.g., the hospital collects data from patients and stores them securely). However, other non-direct providers/users are malicious (e.g., patients don't trust other hospitals). Following previous works [44, 60], we assume that each individual contributes to exactly one row of one dataset. Providers and users hope to disclose datasets to the search platform for augmentation; the search platform is assumed to be malicious. Each provider or user sets a privacy budget (ϵ, δ) for each of their datasets; each budget is independent of other datasets and the search platform. Following prior work [34, 38, 59], we assume that the schemas and the domains of join keys (as group-by attributes) are public.

The requirement sits between those of LDP, SDP and GDP. GDP assumes that the centralized (second level) aggregator is trusted, while LDP assumes both levels of aggregators are untrusted. The shuffler in SDP can be implemented as first (SDP-1, similar to ours) or second-level (SDP-2, similar to GDP) aggregators. In practice, we believe our model fits the structure of many organizations, where individuals solely trust their immediate data aggregator (like a hospital or service provider), but do not trust any other aggregators. Further, regulatory requirements like HIPAA [8] and CCPA [6] place privacy protection requirements on the first level aggregator.

The **differentially private task-based data search** problem is defined as Problem 1 with the above privacy requirements satisfied.

Example 1. *Healthcare providers collect data from patients (individuals); these datasets are classified as Protected Health Information (PHI) by HIPAA. They are obligated by HIPAA to protect the security of PHI, thus trusted by patients. Healthcare providers want to share data to a search platform for public benefit (providers) or improve ML accuracy for better (users). Following HIPPA guidance, they de-identify collected datasets through DP; these de-identified datasets are no longer considered PHI and can be disclosed.*

Private task-based data search is particularly challenging because a single request requires retraining models over a potentially combinatorial space of augmented datasets that are constructed by joining and unioning candidate datasets. Even for a single request, how can we avoid exhausting the user's and the providers' privacy budgets? And how can we scale to massive numbers of datasets and requests, without degrading search quality? Is there an intermediate representation [14, 31] of a dataset that can be made differentially private once, but is useful for the whole search?

Our main insight is inspired by the recent Kitana system [62] that uses factorized linear regression to accelerate data search. As described in Section 3.1, Kitana computes the gram matrix semi-ring for each dataset, and uses the semi-ring rather than the raw dataset to quickly join/union with a candidate dataset and evaluate the linear regression model accuracy. While semi-ring was used by Kitana for performance, we find them an ideal intermediate representation for DP. We propose FDP which privatizes semi-ring to support private ML over joins and unions. We further design *Saibot* based on FDP to make the whole process of data storage, data discovery [15, 27], and data search differentially private.

3 FACTORIZED DIFFERENTIAL PRIVACY

In this section, we introduce FDP, a mechanism that privatizes sufficient statistics to support differentially private task-based data search. We start with the background of factorized ML, then extend it to general monomial semi-ring, and next present our main mechanism algorithms and analyze its errors.

3.1 Factorized Machine Learning Primer

We start with the fundamental concepts of annotated relations and aggregation pushdown, then introduce factorized ML [12, 45].

Annotated Relations. The annotated relational model [29] maps $t \in R$ to a commutative semi-ring $(D, +, \times, 0, 1)$, where D represents a set, $+$ and $\times$ are commutative binary operators closed over D, and $0/1$ are the zero/unit elements. An annotation for a tuple $t \in R$ is denoted as $R(t)$. Annotated relations allow for query optimizations of different aggregation functions discussed next. For example, the natural numbers semi-ring supports the count aggregation.

Semi-ring Aggregation Query. Semi-ring aggregation queries can now be reformulated using annotated relations by translating group-by, union and join operations into addition (+) and multiplication (×) operations over the semi-ring annotations, respectively.

$$(\gamma_{\mathbf{A}}R)(t) = \sum\{R(t_1) |\ t_1 \in R, t = \pi_{\mathbf{A}}(t_1)\}$$
$$(R_1 \cup R_2)(t) = R_1(t) + R_2(t)$$
$$(R_1 \bowtie R_2)(t) = R_1(\pi_{S_{R_1}}(t)) \times R_2(\pi_{S_{R_2}})(t))$$

(1) The annotation for each group-by result in $\gamma_{\mathbf{A}}R$ is the sum of the annotations of all tuples within the corresponding group. (2) The annotation for union $R_1 \cup R_2$ is the sum of semi-ring annotations. (3) The annotation for join $R_1 \bowtie R_2$ is the product of semi-ring annotations from the contributing tuples in R_1 and R_2.

Aggregation Pushdown. The optimization of factorized ML [12, 51] involves the distribution of aggregations (additions) through joins (multiplications). For example, consider the query $\gamma_D(R_1[A,B] \bowtie R_2[B,C] \bowtie R_3[C,D])$. Rather than applying the aggregation on the join result (which is $O(n^3)$ where n is relation size), the aggregation can be performed on R before joining it with S, and this process can be repeated two more times (in $O(n)$):

$$\gamma_D(\gamma_C(\gamma_B(R_1[A,B]) \bowtie R_2[B,C]) \bowtie R_3[C,D])$$

The associativity of additions can be similarly exploited for union $\gamma_A(R_1[A,B] \cup R_2[A,B]) = \gamma_A(R_1[A,B]) \cup \gamma_A(R_2[A,B])$.

Factorized Linear Regression. The fundamental optimization of factorized ML is aggregation pushdown, but different semi-rings are used for different models. We use linear regression as an example.

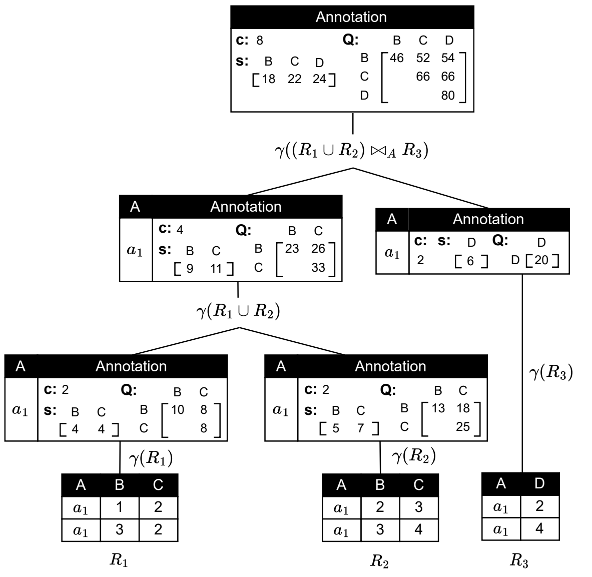
We first provide an overview of linear regression and its core computations. Linear regression is a statistical technique that aims to identify the optimal fit parameters θ between one or more features and a target variable. Using matrix notation, we represent the training data as $\mathbf{X}$ (with an added column of 1 for the intercept), and the target variable as $Y \in \mathbb{R}^n$. The objective is to minimize the square loss function $\theta^* = argmin_\theta \|Y - \mathbf{X}\theta\|^2$, which has a closed-form solution $\theta^* = (\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^TY$. If we consider Y as a special feature and append it to $\mathbf{X}$ to create $\mathbf{X}' = [\mathbf{X} \mid Y]$, we observe that $\mathbf{X}'^T\mathbf{X}'$ is the core sufficient statistics to compute, where each cell represents the sum of products between pairs of features.

We can compute $\mathbf{X}'^T\mathbf{X}'$ over the join $R_1 \bowtie ... \bowtie R_k$ by the covariance matrix semi-ring [51]. Given m features, the semi-ring is defined as a triple $(c, \mathbf{s}, \mathbf{Q}) \in (\mathbb{Z}, \mathbb{R}^m, \mathbb{R}^{m\times m})$, which contains the count, sums, and sums of pairwise products respectively. The zero and one elements are $\mathbf{0} = (0, \mathbf{0}^m, \mathbf{0}^{m\times m})$ and $\mathbf{1} = (1, \mathbf{0}^m, \mathbf{0}^{m\times m})$. + and × between two annotations a and b are defined as:

$$a + b = (c_a + c_b, \mathbf{s}_a + \mathbf{s}_b, \mathbf{Q}_a + \mathbf{Q}_b)$$
$$a \times b = (c_a c_b, c_b\mathbf{s}_a + c_a\mathbf{s}_b, c_b\mathbf{Q}_a + c_a\mathbf{Q}_b + \mathbf{s}_a\mathbf{s}_b^T + \mathbf{s}_b\mathbf{s}_a^T)$$

Then, computing $\mathbf{X}'^T\mathbf{X}'$ is reduced to executing $\gamma(R_1 \bowtie ... \bowtie R_k)$, where aggregation can be pushed down as discussed before.

Example 2. *Consider a database with three relations R_1, R_2, R_3 and suppose we aim to train linear regression on $\mathbf{X} = \gamma((R_1 \cup R_2) \bowtie_A R_3)$ with B as the feature and C as the target variable. The optimized query plan is shown in Figure 3, where aggregations are pushed down: $\gamma((\gamma_A(R_1) \cup \gamma_A(R_2)) \bowtie_A \gamma_A(R_3))$. We can push down the aggregation to derive $\gamma((\gamma_A(R_1) \cup \gamma_A(R_2)) \bowtie_A \gamma_A(R_3))$. Finally, we can use the aggregation result to fit the linear regression model, and obtain:*



Annotation
c: 8
s: B C D [18 22 24]
Q: B C D; B [46 52 54]; C [66 66]; D [80]
γ((R1 ∪ R2) ⋈A R3)
A Annotation
a1 c: 4 s: B C [9 11] Q: B C; B [23 26]; C [33]
A Annotation
a1 c: 2 s: D [6] Q: D; D [20]
γ(R1 ∪ R2)
A Annotation
a1 c: 2 s: B C [4 4] Q: B C; B [10 8]; C [8]
A Annotation
a1 c: 2 s: B C [5 7] Q: B C; B [13 18]; C [25]
γ(R1)
γ(R2)
γ(R3)
A B C; a1 1 2; a1 3 2
A B C; a1 2 3; a1 3 4
A D; a1 2; a1 4
R1
R2
R3

Figure 3: Optimized query plan of $\gamma((R_1 \cup R_2) \bowtie_A R_3)$ for factorized ML. Aggregations are pushed down before joins.

$$\theta = (\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^TY = \begin{bmatrix} \sum B^2 & \sum B \\ \sum B & \sum 1 \end{bmatrix}^{-1} \begin{bmatrix} \sum BC \\ \sum C \end{bmatrix} = \begin{bmatrix} 46 & 18 \\ 18 & 8 \end{bmatrix}^{-1} \begin{bmatrix} 52 \\ 24 \end{bmatrix}$$

After obtaining the model parameters θ, the model performance can also be evaluated. For square loss, $\sum(y-\theta x)^2 = \sum(y^2 - 2\theta^T x^T y + \theta^T x^T x\theta) = \sum y^2 - 2\theta\mathbf{X}^TY + \theta^T\mathbf{X}^T\mathbf{X}\theta$. The final aggregation result provides the necessary statistics to compute this expression.

3.2 Monomial Semi-ring

In the preceding section, we discuss the gram-matrix semi-ring that trains linear regression over joins and unions. In the literature on differentially private ML, similar sufficient statistics as a vector of monomials [33, 39, 57] have been developed which, after privatization, train differentially private linear regression, decision tree and generalized linear models. However, these sufficient statistics are for a single dataset. This section combines these two lines of work: we extend sufficient statistics as semi-ring with algebraic operators (+ and ×) to support ML over arbitrary augmentation for data search. In the next section, we propose `FDP` which privatizes semi-ring to support differentially private search.

We first define the k-order monomial in sufficient statistics:

Definition 2 (k-order Monomial). *Given n random variables $f_1, f_2, ..., f_n$, the k-order monomials are random variables of monomials of the form $p = f_1^{k_1} f_2^{k_2} ... f_n^{k_n}$, where $k_1, k_2..., k_n$ is n non-negative integers such that $\sum_{i=1}^n k_i = k$.*

The core statistics we aim to compute for ML are the expected value of each monomial $E[p]$. For example, 1-order monomials can estimate means, and 1,2-order monomials can estimate covariance, the core sufficient statistics for linear regression. Additionally,

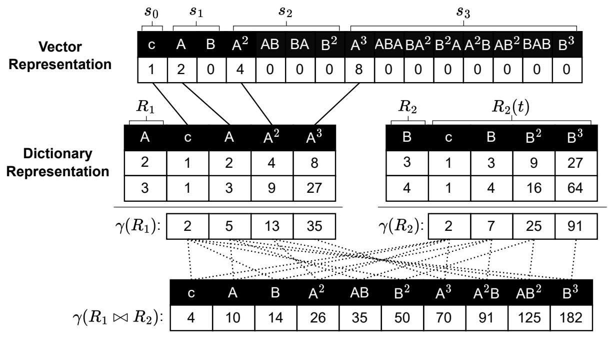


Figure 4: Example of aggregating 3-order monomial semi-ring over the join: $\gamma(R_1[A] \bowtie R_2[B])$. Dictionary representation eliminates redundancy (the same monomials are mapped to the same key) and sparsity (keys containing 0 are not materialized) from vector representation. The dotted line maps the contributing components to aggregated results.

generalized linear models can be approximated by high-order polynomials using Taylor series expansions [33].

We next define the k-order monomial semi-ring, which can combine the monomials from multiple relations to compute the polynomial statistics over join and generalizes factorized linear regression [51] (which only uses 2-order monomial semi-ring).

Definition 3 (k-order monomial semi-ring). *Given m features $f_1, f_2, ..., f_m$, the k-order monomial semi-ring has domain of a vector with size $\frac{1-m^k}{1-m}$. The domain breaks into k subvectors $[s_0, s_1, ..., s_k]$ where s_i is a vector of size m^i. Then, given two semi-ring element $a = [s^a_0, s^a_1, ..., s^a_k]$ and $b = [s^b_0, s^b_1, ..., s^b_k]$, we define:*

$$a + b = [t^a_0 + s^b_0, s^a_1 + s^b_1, ..., s^a_k + t^b_k]$$

$$a \times b = [s^a_0 \otimes s^b_0, \sum_{i=0}^{1} s^a_i \otimes s^b_{1-i}, ..., \sum_{i=0}^{k} s^a_i \otimes s^b_{k-i}]$$

where $\otimes : R^p \times R^q \longrightarrow R^{pq}$ is the tensor product defined as: for $\mathbf{a} = [a_1, a_2, ..., a_p]$ and $\mathbf{b} = [b_1, b_2, ..., b_q]$, tensor prouduct computes the pairwise product $\mathbf{a} \otimes \mathbf{b} = [a_1b_1, a_1b_2, ..., a_pb_1, a_pb_2, ..., a_pb_q]$.

The zero element is a vector of all zeroes, and the one element is a vector with non-zero $s_0 = [1]$, but the rest as all zeroes.

Intuitively, each subvector s^a_k holds the k-order monomials with a size m^k, as there are m^k possible permutations with repetition. In order to compute statistics using a k-order monomial semi-ring, we annotate R by assigning to each tuple t its monomials (non-existing features are considered to be all zeros). Note that, while this vector representation provides a straightforward way to define semi-rings for arbitrary orders, it is inherently inefficient and can be optimized by the dictionary representation discussed next.

Dictionary Representation. The vector representation has redundancies (e.g., $f_1f_2 = f_2f_1$) and sparsity (features that do not exist are zeros). To overcome these, dictionary representations [37] are utilized: monomials serve as keys to deduplicate, and monomials with zeros are not materialized. We next provide an example of semi-ring operations over the dictionary representation for join-aggreagation:

Example 3. *Consider two relations of a single feature $R_1[A] = [2, 3]$ and $R_2[B] = [3, 4]$, and the aggregation query $\gamma(R_1 \bowtie R_2)$ for 3-order monomial semi-ring. Figure 4 illustrates the annotated relations and the query processing. To start, the aggregations are pushed down by summing each monomial. Next, the monomials are combined according to the multiplication operator for join.*

Assuming the join result as the population, we can use the aggregated semi-ring to compute statistics (i.e. the expected monomials). Given the join result $R_\bowtie$ as the population, and let $s = \gamma(R_\bowtie)$. Then, for monomial p: E[p] = s[p]/s[c], where c is the count (0-order monomial). For example, in Figure 4, $E[AB] = s[AB]/s[c] = 35/4 = 8.75$.

The monomial semi-ring has to potentially support differentially private data search on a large data corpus: both providers and users compute locally aggregated monomials (e.g., $\gamma(R_1)$ and $\gamma(R_2)$ in Figure 4) which are privatized once and re-used as sufficient statistics for ML across various future augmentations for search. In the next section, we implement this idea as `FDP` mechanism.

3.3 FDP Mechanism

In this section, we describe the `FDP` mechanism, which applies Gaussian mechanism to locally aggregated monomials. These privatized aggregated monomials can then be used (as post-processing without additional privacy cost) to evaluate various data augmentations and solve the *differentially private data search* problem.

We make the following simplification: (1). features consist only of numerical attributes, and join keys consist only of categorical attributes. (2). The group-by operator γ has been extended to annotate group-by keys without tuples with zero elements (group-by attribute domains are assumed public in Section 2.3). (3). Datasets have been pre-processed such that the $l2$ norm of the features in each tuple is bounded by a constant value B.

Algorithms. Algorithm 1 illustrates the `FDP` mechanism,which operates as follows: each user or provider provides (1) the relation R to be privatized before uploading to the search platform, (2) the join key A[2], which = *null* when the dataset is only for union, (3) the order of polynomial k based on the model to support, and (4) the privacy budget (ϵ, δ) associated with the dataset. The `FDP` mechanism takes these inputs, computes locally aggregated monomials and applies Gaussian mechanism for R'.

Theorem 5. *`FDP` is $(\epsilon, \delta) - DP$.*

Proof Sketch. `FDP` is an instantiation of Gaussian mechanism [19]. Therefore, we only need to show the correctness of the sensitivity Δ. Here, we study the simple cases of the union and join of 1 feature (with smaller Δ), and the union of 2 features with 1/2-order monomial semiring; these cases are sufficient to illustrate the key intuitions, and the full proof can be found in Appendix A.

• *(1 feature, Union, any order)* For union, count (0-order monomial) remains unchanged as we consider bounded `DP`, where the neighbour relation has one tuple modified (instead of removed/added). Let the modified feature value be $a \longrightarrow a'$ where both a and a' have a domain of $[-B, B]$. Then, for the i-th monomial, the squared difference

[2]A could be composite. To support different join keys, the privacy budget can be split among different key combinations. Additionally, optimization techniques can be applied to take advantage of the correlations between join keys [47].

is $(a^i - a'^i)^2$. When i is odd, $a^i \in [-B^i, B^i]$, and $(a^i - a'^i)^2 \le (2B^i)^2$. When i is even, $a^i \in [0, B^i]$, and $(a^i - a'^i)^2 \le (B^i)^2$.

•*(1 feature, Join, any order)* For join, the query additionally groups results by the join key A. This can be considered as a histogram [61], where each bin key is the join key, and the value is the k-order monomial semi-ring. The neighbour relation has two cases: the modified tuple has the join key changed or not. If the join key doesn't change, this is the same as the union case. If the join key changes, there are two bins with a maximum square difference of $\sum_{i=0}^{k} B^{2i}$ (note that, unlike the union, the counts change). Therefore, the sensitivity is bounded by $\sqrt{2\sum_{i=0}^{k} B^{2i}}$. Finally, we take the maximum.

•*(2 features, Union, 1-order)* Let the modified feature value be $(a, b) \rightarrow (a', b')$ where both $a^2 + b^2$ and $a'^2 + b'^2$ are $\le B^2$. Then, consider the 1-order monomials $(a, b), (a', b')$. The squared difference is:

$$(a - a')^2 + (b - b')^2 \le (2a^2 + 2a'^2) + (2b^2 + 2b'^2) = 4B^2$$

The sensitivities for higher odd orders are similar.

• *(2 features, Union, 2-order)* For even-orders, we can obtain a tighter bound. Consider the 2-order monomials $(a^2, ab, b^2), (a'^2, a'b', b'^2)$. The squared difference is:

$$\begin{aligned}
&(a^2 - a'^2)^2 + (ab - a'b')^2 + (b^2 - b'^2)^2 \\
\le&(a^2 - a'^2)^2 + 2(ab - a'b')^2 + (b^2 - b'^2)^2 \\
=&(a^4 - 2a^2a'^2 + a'^4) + (2a^2b^2 - 4aba'b' + 2a'^2b'^2) + (b^4 - 2b^2b'^2 + b'^4) \\
=&(a^2 + b^2)^2 + (a'^2 + b'^2)^2 - 2(aa' + bb')^2 \le B^4 + B^4 - 0 = 2B^4
\end{aligned}$$

For higher even orders, we can similarly amplify the monomials by the binomial coefficients (second line) to find a non-negative red term for even-order monomials, resulting in a tighter bound. Extending to joins follows a similar approach as the single feature case, where we consider group-by queries as histograms. □

Algorithm 1: `FDP` mechanism

```
inputs : Relation $R$, Join Key $A$, Order $k$, DP budget $(\epsilon, \delta)$
output : Privatized Aggregated Relation $R'$
1  if $A = null$ then
2      // union only;
3      $\Delta = \sqrt{\sum_{i=1}^{k} (\text{if i odd: 4, elif \#fea=1: 1, else: 2}) \cdot B^{2i}}$;
4      $\sigma = \sqrt{2\ln(1.25/\delta)}\Delta/\epsilon$;
5      $R' = \gamma(R)$;
6      // add i.i.d. noises to each 1-k order monomial $s$;
7      $R' = \{s : R'[s] + e \sim \mathcal{N}(0, \sigma^2)$ for 1-k monomial $s\}$;
8  else
9      $\Delta = max(\sqrt{\sum_{i=1}^{k} (\text{if i odd: 4, elif \#fea=1: 1, else: 2}) \cdot B^{2i}}, \sqrt{2\sum_{i=0}^{k} B^{2i}})$;
10     $\sigma = \sqrt{2\ln(1.25/\delta)}\Delta/\epsilon$;
11     $R' = \gamma_A(R)$;
12     foreach $a \in dom(A)$ do
13         // add i.i.d. noises to each 0-k order monomial $s$;
14         $R'(a) = \{s : R'(a)[s] + e \sim \mathcal{N}(0, \sigma^2)$ for 0-k monomial $s\}$;
15 return $R'$;
```

3.4 Comparison with Other Mechanisms

We next analyze the error of `FDP` in estimating the statistics s (expected values of monomials). Generally, the expected errors of s are correlated with the error of the target model parameter β and accuracy; we will study the confidence bound for linear regression parameter in the next section, where the s error is the key factor.

Setting. We consider a data corpus with size n_{corp} (defined as the number of provider datasets) and has received n_{req} requests. To simplify the analysis, we assume that: (1) the search only uses union operations (and we will discuss the extension to join). (2) each dataset shares the common characteristic of having a single feature, n tuples, and a privacy budget of (ϵ, δ). The search platform evaluates all possible augmentations, with each augmentation corresponding to a unique combination of provider datasets.

Metrics. The goal is to evaluate, for each augmentation, the expected $l2$ error of the privatized set of monomials $\tilde{s}$: $E[\|s - \tilde{s}\|_2]$.

Mechanisms. We consider three types of mechanisms:

- `FDP` applies Algorithm 1 to the local aggregates of all datasets, and combines the aggregates with factorized ML.
- `LDP`[3] applies Algorithm 1 to privatize each tuple.
- `GDP`[4] first computes the non-privatized estimation for each augmentation, and then applies the Gaussian mechanism to privatize the estimation. To ensure $(\epsilon, \delta) - DP$ for all $n_{req}(2^{n_{corp}-1} - 1)$ augmentations, the privacy budget has to be split.
- `SDP` privatizes each tuple, similar to `LDP`. However, it applies Laplace mechanism with the amplified privacy budget. These tuples are shuffled either locally (`SDP`-1) or globally (`SDP`-2); similar to `GDP`, `SDP`-2 needs to split the budget.

Proposition 6. *For the estimation of each augmentation (assuming that the number of augmented datasets and the order of s are small constants), `FDP`/`SDP`-1 has expected l2 error of $\tilde{O}(\Delta/n\epsilon)$, while `LDP` has error of $\tilde{O}(\Delta/\sqrt{n}\epsilon)$ and `GDP`/`SDP`-2 has error of $\tilde{O}(n_{req}2^{n_{corp}}\Delta/n\epsilon)$, where $\tilde{O}(\cdot)$ hides at most a logarithmic term.*

The proof is in Appendix B.

Remark. Proposition 6 highlights the limitations of prior mechanisms: `GDP`/`SDP`-2 is competitive only for a small corpus and its budget is depleted quickly for larger # requests/corpus size. Additionally, it requires trust in the search platform for centralized aggregators/shufflers. `LDP` adds excessive noise to each tuple, requiring quadratically more tuples to achieve the same level of error as `FDP`. Although theoretically, `SDP`-1 can achieve the same complexity as `FDP` by amplifying privacy, such amplification is significant only for a large number of tuples n. For instance, given target budget ϵ=1 and δ=10^{-6}, ϵ is amplified when n reaches ~650 [24, 26]. However, amplification is most necessary for small n, where `SDP` provides a much larger error than `FDP` shown in Section 5.2.

To expand the analysis to include joins, we need to additionally consider the errors from group-bys based on the domain size and the multiplication of privatized monomials. The comparisons between `FDP` and the other mechanism are nonetheless similar: `LDP` still requires a quadratically larger data size. While for small corpus

[3]An alternative is to apply Gaussian mechanism to raw tuple then compute monomial semi-ring; this however results in an even larger error.
[4]There are other alternatives like perturbing objectives or gradients; however they are similarly limited by the combinatorially large number of models to train.

and requests, GDP may outperform FDP by adding noise to full aggregates (instead of to each bin grouped by the join key), its budget depletes for larger requests and corpus size.

3.5 Differentially Private Data Search Platform

In this section, we discuss *Saibot*, a data search platform that integrates FDP to ensure differential privacy.

Provider. The architecture of the *Saibot*, which utilizes FDP to ensure privacy, is illustrated in Figure 2. The data provider possesses a dataset R and determines the desired support operation ($\bowtie$ /$\cup$[5] or $\cup$-only) and model type (M) to be applied to the dataset. If join needs to be supported, the join key A must also be specified. Various model types are supported, such as tree-based models ($k = 1$), linear regression ($k = 2$) and approximated GLM ($k = 3$). The FDP mechanism is applied to compute the privatized sufficient statistics $\gamma(R)'$, which are then uploaded to the *data storage* in the data *Saibot*. If *Saibot* is not trusted, *data storage* only stores statistics privatized locally, but not raw data. All operations over $\gamma(R)'$ are post-processing of $\gamma_A(R)$ without additional DP costs.

User. The user has model type M and R_{train}, and wants to improve accuracy on R_{test}. The user computes and submits to *Saibot* the privatized sufficient statistics $\gamma(R_{train})'$ and $\gamma(R_{test})'$. *Data discovery* returns a set of joinable or unionable relations R from *data storage*. Then, *Data search* applies greedy algorithm (following Kitana [62]): in each iteration, it evaluates each candidate and adds the one that most improves the model accuracy. *Saibot* is agnostic to the search algorithm, and others [17, 54] can also be used.

Data Discovery. Previous data discovery systems [15, 27] leverage MinHash sketches, column type and data distribution statistics; *Saibot* supports all of them. Specifically, for categorical attributes, we utilize minhash sketches, computed from public domains, to measure set similarity. For numerical attributes, we rely on public schemas for column names and types. Additionally, we construct (approximated) data distribution statistics such as count, mean, standard deviation, and correlation from the privatized 2-order monomial semi-rings, without additional DP costs.

Preprocessing. As part of applying the DP mechanism, users and providers can pre-process their datasets to improve task utility and robustness. In *Saibot*, we apply two steps. We first apply outlier removal (>1.5 std from the mean), which typically improves model robustness, but also reduces the bound B used in DP [40] (and thus the amount of noise needed). Secondly, all DP mechanisms (including ours) degrade as a dataset's dimensionality grows, since the bound B increases, resulting in larger noises. To address this, we apply dimensionality reduction [42] (K=1 works best in our experiments), then rescale the tuples to ensure the max $l2$ norm $\leq B$. K=1 also benefits from a lower sensitivity Δ in Algorithm 1. In our real-world experiments (Section 5.1), we apply these to all datasets and DP baselines.

Supporting Varied Privacy Needs. A unique benefit of *Saibot*'s design is that it can readily adapt to different privacy requirements. In cases where pure DP ($\delta = 0$) is required, FDP can be modified to apply Laplace mechanisms [20]. In situations where individuals don't trust providers or users, FDP can be reduced to LDP to add noises to individual tuples. Conversely, SDP only supports approximate DP and GDP always requires a trusted centralized aggregator.

Scope. While *Saibot* can employ FDP to support a wide range of models including tree-based models [50] and approximate generalized linear models [39], this paper will focus on linear regression [51] because it's widely used and is adopted by previous data search platform [16, 49, 62]. In the next section, we analyze the task utility for linear regression and propose further optimizations.

4 LINEAR REGRESSION ANALYSIS AND OPTIMIZATION

This section examines the ML task *utility Saibot* provides and proposes further optimizations. For the ML task, we focus on linear regression due to its popularity and leave the analyses of other models as future works. We start with the assumption of linear regression on many-to-many join (as opposed to one-to-one join [32, 56]), which is challenging due to the issues of duplication and independence. We then propose an unbiased estimator. Next, we explore the confidence bounds for the linear regression parameters and propose optimizations to FDP that can further tighten the bound.

4.1 Linear Regression on Many-to-Many Join

Linear regression assumes a noisy linear relationship between the features and target variable: $y = \beta X + e$, where e is a vector of error terms. This is consistent with our assumption so far if $R_{\bowtie} = \pi_1(R) \bowtie \pi_2(R) \bowtie \ldots$ is the population, and let us use the monomial semi-ring to compute the expected s. However, $R_{\bowtie}$ is usually not the population when many-to-many joins are present because joins create a Cartesian product for each matching join key. This leads to (1) duplicated tuples from different relations and (2) unexpected *independence* between attributes for each join key value. In other words, the same y values are repeated for different combinations of X, and features from different relations with the same join key are uncorrelated, leading to biased estimation.

To the best of our knowledge, many-to-many join is understudied and the closest related work is multi-view learning [28, 30], which proposes pre-aggregating (e.g., averaging) features from other tables. However, pre-aggregation could introduce errors for long join paths due to Simpson paradox [46] (average of averages is not equal to the average). In contrast, we propose an unbiased estimator based on vertical federated ML and takes into account the statistical properties; this complements prior factorized ML work [12, 45] which studied the computational complexity of many-to-many joins.

Our analysis and design will focus on an easy-to-explain case inspired by vertical federated ML [32, 56]. We consider a relation R to train ML models. However, R is not directly observable, and each party only has access to a projection $\pi(R)$. Multiple $\pi(R)$ may have many-to-many relationships on the common attribute (join key), as opposed to the one-to-one relationships studied by federated ML. The objective is to jointly train ML models on R.

Unbiased Estimator. Given R of cardinality n, suppose there are two parties holding different projections $\pi_{F_1}(R)$ and $\pi_{F_2}(R)$, and the goal is to compute the 2-order monomial semi-ring $\gamma(\pi_{F_1 \cup F_2}(R))$. However, factorized ML is trained on $R_{\bowtie} = \pi_{F_1,J}(R) \bowtie_J \pi_{F_2,J}(R)$ with join key $J = F_1 \cap F_2$; $R_{\bowtie}$ is likely to differ from $\pi_{F_1 \cup F_2}(R)$

[5]Any dataset supports join also supports union by aggregating out the join key.

(unless J is primary key), resulting in bias. We propose an unbiased estimator for s based on $s' = \gamma(R_{\bowtie})$ to address this.

Proposition 7 (Unbiased Estimator of s over R). *We make the simplifying assumption that J is uniformly distributed (if $d = |dom(J)|$, each $j \in J$ appears n/d times in R) and is not correlated with any other attribute. Let $s' = \gamma(R_{\bowtie})$. Then,*

$$\hat{s} = \begin{cases} f_1 f_2 = \frac{1-n}{1-d}\frac{s'[f_1 f_2]}{s'[c]} + \frac{n-d}{1-d}\frac{s'[f_1]}{s'[c]}\frac{s'[f_2]}{s'[c]} \\ \qquad\qquad \text{for } f_1 \in F_1, f_2 \in F_2 \\ p = s'[p]/s'[c] \quad \text{for any other monomial } p \end{cases}$$

$\hat{s}$ is an unbiased estimator of monomial semi-ring $s = \gamma(\pi_{F_1 \cup F_2}(R))$.

The proof can be found in Appendix C. We assume vertical partitions of R, but real-world datasets are likely to be also horizontally partitioned; the estimators could be refined for these cases. Our analysis studies the base case, and the unbiased estimator can be recursively applied for more complex cases involving multiple joins and unions. Note that the estimators are post-processing steps without compromising privacy.

4.2 Simple Linear Regression Analysis

Building on the assumption in the previous section, this section studies the confidence bound of factorized linear regression. Compared to [57], our analysis focuses on the simple linear regression with one feature and requires weaker assumptions; this simple scenario is sufficient to demonstrate the advantages of FDP over other mechanisms, and motivates optimization. We begin with the single relation case, and then expand to union and join.

We start with the definition of the confidence bound, then use it to measure the utility of private estimators.

Definition 4 (Confidence Bound). *Given parameter θ, the $(1-p)$ confidence bound $C^{\tilde{\theta}}_{\hat{\theta}}(p)$ for an private estimator $\tilde{\theta}$ is:*

$$C^{\tilde{\theta}}_{\hat{\theta}}(p) = \inf \{b : \mathbb{P}[|\tilde{\theta} - \hat{\theta}| \le b] \ge 1 - p\}$$

where $\hat{\theta}$ is the non-private estimator.

We consider relation $R[x, y]$ with one feature x, target variable y and cardinality n. We want to train $y = \beta_x \cdot x + \beta_0$, and focus on the parameter β_x; β_x has an optimal non-privitized estimator $\hat{\beta_x} = \frac{\widehat{E[xy]} - \widehat{E[x]}\widehat{E[y]}}{\widehat{E[x^2]} - \widehat{E[x]}^2} = \widehat{\sigma^2_{\text{xy}}}/\widehat{\sigma^2_{\text{x}}}$, where $\widehat{\sigma^2_{\text{xy}}}$ and $\widehat{\sigma^2_{\text{x}}}$ are polynomials that can be derived from aggregeted 2-order monomials $\gamma(R)$. We apply FDP to compute the privatized 2-order $\gamma(R)$ and study the confidence bound of the privatized estimator $\tilde{\beta_x}$. Note that more familiar error definitions like mean-squared-error can be upper bounded, roughly, by the square of the confidence bound.

Theorem 4.1 (Confidence Bound of $\tilde{\beta_x}$). *For every p where $\tau_1 < 1$ holds, the $(1-p)$ confidence bound for $\tilde{\beta_x}$ is:*

$$C^{\tilde{\beta_x}}_{\hat{\beta_x}}(p) \le \tau_2 + \frac{\tau_1}{1-\tau_1}\left(\hat{\beta_x} + \tau_2\right)$$

where $\tilde{\beta_x}$ ($\hat{\beta_x}$) is the private (non-private) estimate of β_x. Let B_1 and B_2 be the $(1-p)$ confidence bounds for $\widetilde{\sigma^2_{\text{x}}}$ and $\widetilde{\sigma^2_{\text{xy}}}$ respectively. Then $\tau_1 = B_1/\widehat{\sigma^2_{\text{x}}}$ and $\tau_2 = B_2/\widehat{\sigma^2_{\text{x}}}$ are both $O\left(\frac{B^4 \ln(1/\delta)\ln(1/p)}{\epsilon^2 n \widehat{\sigma^2_{\text{x}}}}\right)$. The probability is taken over the randomness of FDP.

The proof and extension to multi-features can be found in Appendix D. Theorem 4.1 demonstrates that the private estimator $\tilde{\beta_x}$ is asymptotically close to the non-private $\hat{\beta_x}$. The key factors in reducing the discrepancy are B_1 and B_2. GDP and SDP-2 have combinatorially large B_1 and B_2 due to the splits of budget. LDP requires quadratically more data than FDP to achieve the same magnitude of B_1 and B_2.

Extension to Factorized ML. The full procedures to extend the confidence bounds for factorized ML are in Appendix D. For the union of k datasets: $R = R_1 \cup R_2 ... \cup R_k$, τ_1 and τ_2 are reduced by a factor of $\sqrt{k}$, while the rest remain unchanged. For the join of two datasets $R[x, y, J] = R_1[x, J] \bowtie R_2[y, J]$, where $|dom(J)| = d$, there is additional noise to the count which could cause distortion if the privatized count is close to or less than 0. To address this, an additional assumption that noises to count is $o(n/d)$ is needed [57], resulting τ_1 and τ_2 to increase by a factor of $O(\sqrt{d})$ and $O(n/\sqrt{d})$.

4.3 Better Noise Allocation

In Section 4.2, we analyzed the linear regression confidence bounds. We propose to adjust noise allocation to improve the bounds further.

First, previous work (e.g., [13]) has shown that for linear regression over the union, β_x is usually the parameter of interest instead of β_0. In this case, we suggest each provider adding noises directly to $\sigma^2_x, \sigma^2_{xy}$, rather than monomials x^2, xy, x, y. This reduces τ_1 and τ_2 by a factor of $O(B^2\sqrt{\ln(1/\delta)\ln(1/p)}/\epsilon)$ (Appendix E).

Second, join is more challenging to optimize as we add noises locally to monomials to avoid the combinatorially large privacy costs. However, we can still reduce τ_1 and τ_2 by a factor of $O(B^2)$ through smart noise allocation. Our insight is that lower-order monomials are multiplied by more monomials than higher-order ones. For example, in Figure 4, 0-order monomials are multiplied by 0,1,2,3-order ones, while 3-order monomials only are multiplied by 0-order ones. Therefore, we shall decrease the noise to lower-order statistics. FDP-OPT in Algorithm 2 achieves this by dividing the privacy budget for each order monomials; lower order monomials have lower sensitivity and thus fewer noises.

Algorithm 2: FDP-OPT algorithm for Join

```
inputs : Relation $R$, Join Key $A$, Order $k$, DP budget $(\epsilon, \delta)$
output : Privatized Annotated Relation $R'$
1 foreach $i \in \{0, \dots, k\}$ do
2     $\Delta$ = (if i odd: 2, else: $\sqrt{2}$) $\cdot B^i$;
3     $\epsilon', \delta' = \epsilon/(k+1), \delta/(k+1)$ ;
4     $\sigma = \sqrt{2\ln(1.25/\delta')}\Delta/\epsilon'$;
5     $R' = \gamma_A(R)$;
6     foreach $a \in dom(A)$ do
7         // add i.i.d. noises to each i order monomial $s$;
8         $R'(a)$={$s : R'(a)[s] + e \sim \mathcal{N}(0, \sigma^2)$ for i monomial $s$};
9 return $R'$;
```

5 EVALUATION

Can *Saibot* find useful augmentations in a real-world corpus while ensuring DP requirements? Is *Saibot* scalable to the number of requests and data corpus size? How large are the errors from FDP compared to GDP, LDP and FDP? We evaluate *Saibot* using FDP against prior DP over NYC Open Data [11] corpus of 329 datasets for an end-to-end dataset search. We then use ablation studies via synthetic datasets to validate our theoretical analyses.

5.1 Real-world Experiments

Data and Workload. We construct a large data corpus of 329 NYC Open Data [11] datasets. Since prior DP mechanisms need to know the number of requests up front, we create a workload of 5 requests using the following random datasets:

- **Regents** [4] contains 2014-17 regents exams data.
- **ELA** [1] contains 2013-18 Early Learning Assessment (ELA) data.
- **Gender** [3] contains 2013-16 ELA data by grades and gender.
- **Grad** [5] contains 2016-17 graduation outcomes.
- **Math** [2] contains 2013-18 Math grades.

For each request, we look for a single dataset to join/union with the requested dataset. We turn off data discovery so every dataset in the platform is considered. By default, each dataset has privacy budget ($\epsilon = 1, \delta = 10^{-6}$). We report the final $r2$ score evaluated non-privately. For reliability, we run each request 10 times.

Baselines. We consider different DP mechanisms. **Non-P**: doesn't use DP and provides $r2$ upper bound. **FDP**: applies Algorithm 2 to each dataset. **GDP**: following Wang [57], GDP simply applies Algorithm 2 to the augmented dataset. Note that this splits the budget across all augmentations. We use attribute max-frequence from Flex [34] to bound join sensitivity. **LDP**: applies Algorithm 2 to each tuple and uses half the ϵ to perturb the join key with generalized random response [35]. **SDP** is similar to LDP, but applies the Laplace mechanism to each tuple with amplified budget [26]. Since SDP-2 doesn't support joins, we use SDP-1, which shuffles each dataset locally. In each case, we use a failure mechanism that reports $r2 = \infty$ if the privatized $\widetilde{X^TX}$ is not positive definite [13].

Results. Figure 5 shows the non-private $r2$ of 10 runs of private data search for the 5 requests. FDP dominates the DP alternatives and is ~50–90% of the non-DP case. FDP's performance depends on dataset cardinality: the *Gender* dataset contains on average ~40 tuples per join key (compared to >100 tuples per join key in other datasets) and is more vulnerable to noise.

We next vary the number of datasets by sampling $n_{corp} \in \{10, 50, 100, 300\}$ datasets and rerunning each baseline over the smaller corpus. Figure 6 reports the median $r2$. For a small corpus ($n_{corp} = 10$), GDP outperforms FDP because it imputes noise to the aggregated statistics across join key values and there are fewer budget splits, while FDP has to add noise to the individual statistics for each join key. LDP and SDP have low $r2$ due to high noise.

Finally, we vary the number of requests ($n_{req} \in \{1, 10, 50, 100\}$) by sending the same request n_{req} times, and report median $r2$. Figure 7 shows that each baseline is almost invariant to n_{req}, and FDP dominates. In theory, GDP is worse for more requests but is already poor due to the large dataset corpus.

5.2 Synthetic Dataset Experiments

We next validate our theoretical analysis of linear regression using synthetic data, and conduct ablation tests to study the impact of various parameters (number of tuples n, privacy budget ϵ, δ, corpus size n_{corp}, number of requests n_{req} and join key domain size d).

5.2.1 Setup. We generate datasets by first creating a symmetric positive-definite matrix (`make_spd_matrix` in *sklearn*) as the covariance X^TX. We then sample from a multivariate normal distribution with this covariance to create a relation. To ensure the $L2$ norms of tuples $\leq B$=5, we resample for any tuples that exceed this limit.

By default, for union, we generate relations with $n = 1000$ tuples and 3 numerical attributes $[y, x_1, x_2]$. For join, we generate relations with $n = 10000$ tuples and include a categorical join key J uniformly distributed with a domain size of $d = 100$. We construct two vertical partitions with projections $[y, x_1, J]$ and $[x_2, J]$, respectively. We start with $n_{corp} = 2$ datasets, $n_{req} = 1$ requests.

We will report the $l2$ distance to the non-private sufficient statistics (s error) and regression parameter (β error) as metrics. Each experiment will be repeated 100 times, and we will present the medians (dots), as well as the 25th and 75th percentiles (error bars).

5.2.2 DP for Union. Baselines include GDP, LDP (same as in Section 5.1), SDP-1 which shuffles tuples locally, SDP-2 which shuffles the unioned dataset, and FDP using Algorithm 1 rather than Algorithm 2 (which is for join).

First, we vary $n \in \{10, 100, 500, 1K, 10K\}$. Figure 8a and Figure 8b report s and β errors. Since there are $n_{corp} = 2$ datasets, GDP and FDP perform similarly. In contrast, LDP requires a quadratically larger number of data to achieve the same s errors, consistent with our analysis in Section 3.4. SDP's amplification is not significant for small n, when it is needed most, and both variants have high s errors. β error eventually converges to 0 for all baselines, but FDP does so at a comparable rate to GDP (n = 500 vs 10K for the others).

Figure 8c shows that β error naturally correlates with s error, and higher s error beyond increases the chance of failure (β error $=\infty$). The remaining results will focus β error, as it is of interest.

Next, we vary the privacy budget ϵ=0.1 or δ=0 (pure DP). The results are shown in Figure 8d and Figure 8e, respectively. For ϵ=0.1, the plot naturally shifts to the right for a smaller budget. In the case of pure DP with δ=0, FDP, GDP and LDP can adapt to it by applying Laplace mechanism, and achieve similar performance. In contrast, SDP-1 and SDP-2 fail as only approximate DP is supported.

Figure 8f and Figure 8g vary the number of datasets and number of requests $n_{corp}, n_{req} \in \{1, 5, 10, 50, 100\}$, respectively. FDP's β error is flat. LDP, SDP-1 and SDP-2 frequently fail due to high noise, while GDP only performed well when $n_{corp} \leq 5$ or $n_{req} \leq 10$. **GDP is hence unsuitable for large dataset corpora.**

Figure 8h reports the benefit of our linear regression optimization in Section 4.3. For a two-attribute dataset $R[y, x]$, while FDP adds noise to monomials (x, y, x^2, y^2, xy), our optimization adds noise to polynomials ($\sigma^2_x, \sigma^2_{xy}$) because we only care about β_x. We find that FDP-OPT reduces the β error and the likelihood of failure, especially for n<100.

5.2.3 DP for Join. We evaluate different DP mechanisms over the join. Baselines include FDP-OPT, which uses a smart allocation

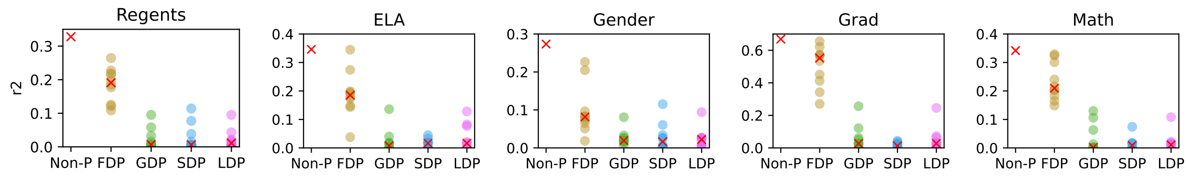


Figure 5: Utility (non-privitized $r2$) of the returned combinations of datasets searched by different baselines over 10 runs, with the median indicated by a red cross. FDP exhibits significantly better utility than the other baselines.

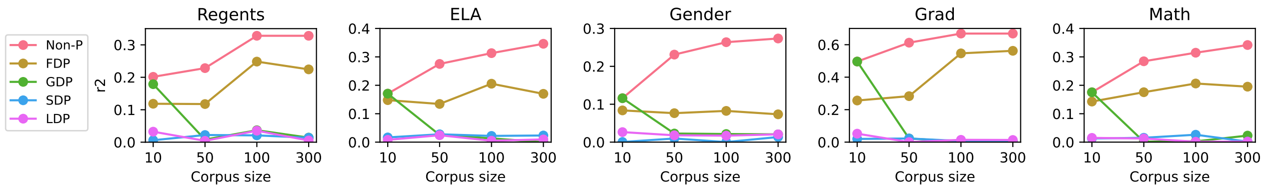


Figure 6: Utility (non-privitized $r2$) of the returned combinations of datasets searched by different baselines when varying the repository size n_{repo}. FDP is scalable for large repository, while GDP only performs well for small n_{repo}.

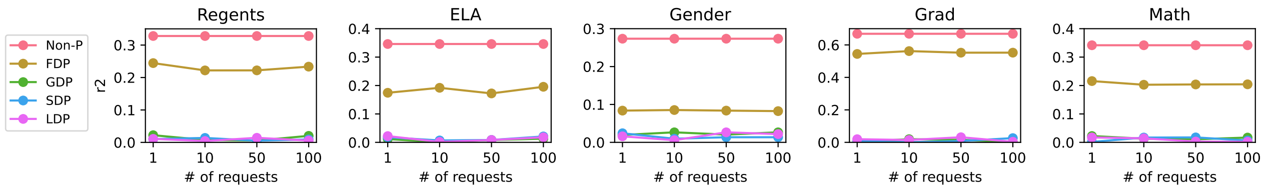
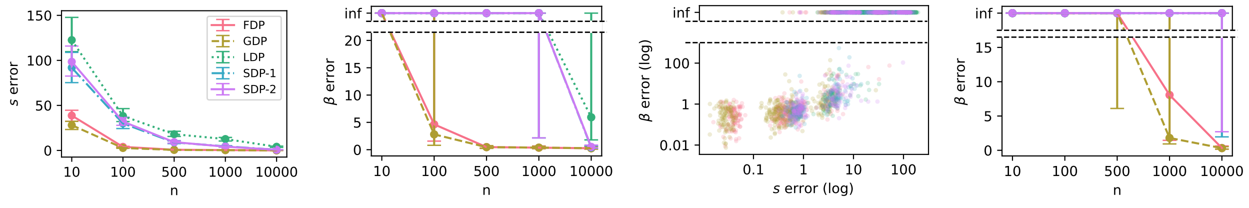
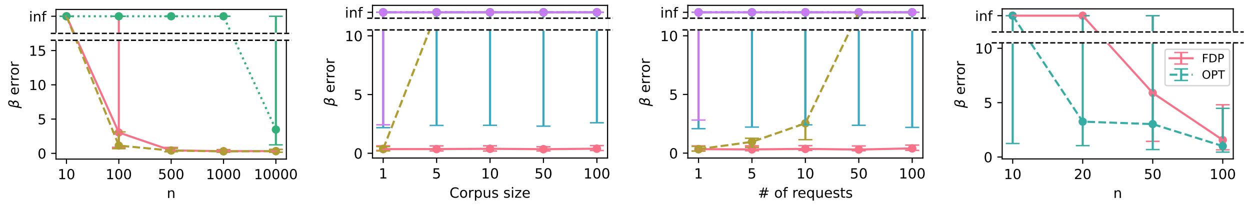


Figure 7: Utility (non-privitized $r2$) of the returned combinations of datasets by different baselines when varying the number of requests n_{req}. FDP consistently performs well, while the others suffer from noise (LDP, SDP) or budget splitting (GDP).

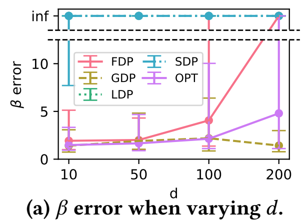


(a) s error when varying n. (b) β error when varying n. (c) Compare β with s error. (d) β error when ϵ=0.1.

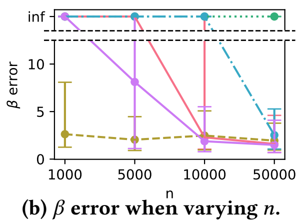


(e) β error when δ=0. (f) β error when varying n_{repo}. (g) β error when varying n_{req}. (h) β error with optimized FDP.

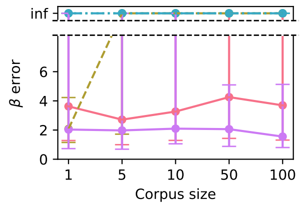
Figure 8: Ablation tests for unions: (a). (b). All baselines show reduced s and β errors as n increases, with FDP and GDP exhibiting the least error; (c). Larger s error results in higher β error and a greater risk of failure; (d). For $\epsilon = 0.1$, the β error plot shifts to the right; (e). For pure DP with $\delta = 0$, FDP can adjust to it and offer comparable utility, while SDP fails; (f). (g). FDP is scalable for large repositories with large numbers of requests, while GDP deteriorates significantly when $n_{repo} > 5$ or $n_{req} > 10$; (h). For simple linear regression when only β_x is of interest, FDP-OPT reduces the β error and failure risk, particularly for small n.



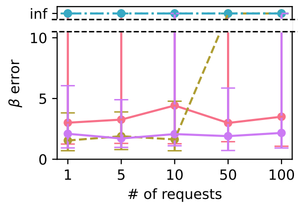
(a) β error when varying d.



(b) β error when varying n.



(c) β error when varying n_{corp}.



(d) β error when varying n_{req}.

Figure 9: Ablation tests for joins: (a). (b). FDP, FDP−OPT and GDP provide low error when varying d and n, while LDP and SDP are dominated by high noises (c). (d). FDP and FDP−OPT show scalability for large repositories with numerous requests, whereas GDP has high errors when n_{corp}>5 or n_{req}>10.

strategy to reduce the noise of lower order statistics, as discussed in Section 4.3. SDP−2 doesn't support joins, so it is not reported.

We use $n_{corp} = 2$ datasets: we fix cardinality $n = 10K$ but vary join key domain size $d \in \{10, 50, 100, 200\}$, then fix $d = 100$ but vary $n \in \{100, 5k, 10k, 50k\}$. The results are shown in Figure 9a and Figure 9b, respectively. FDP, FDP−OPT and GDP have low β error, while LDP and SDP have high failure rates. FDP−OPT outperforms FDP due to better noise allocation. GDP outperforms FDP and FDP−OPT at large d or small n because GDP adds noise directly to the aggregated statistics across join keys, resulting in a smaller amount of noise. In contrast, FDP adds noise for each join key value. However, for large n, FDP and FDP−OPT outperform GDP because it has high sensitivity due to high join fanouts [34].

Figure 9c and Figure 9d respectively vary number of datasets and requests: $n_{corp}, n_{req} \in \{1, 5, 10, 50, 100\}$. Both LDP and SDP have high failure rates, and FDP−OPT outperforms FDP. FDP and FDP−OPT scale to arbitrary numbers of datasets and requests, while GDP is restricted to $n_{corp} \leq 5$ or $n_{req} \leq 10$.

5.2.4 *Join Unbiased estimator.*

Here, we compare the β error of the unbiased estimator proposed in Section 4.1 to the naive estimator over many-to-many joins. We first fix the number of tuples $n = 10000$ but vary join key domain size $d \in \{10, 50, 100, 200\}$, then fix $d = 100$ but vary $n \in \{100, 5K, 10K, 50K\}$, and report the results in Figure 10a and Figure 10b respectively. As d increases, the errors of the unbiased estimator converge to 0, while the biased estimator diverges as it fails to account for many-to-many join. When $n = 100$, the naive estimator achieves similar performance, as each join key has only one tuple (so one-to-one join without bias). However, increasing n introduces duplications and independence (for many-to-many join). The unbiased estimator reduces the noise and provides higher performance than the naive estimator.

6 RELATED WORKS

Dataset search. Traditional data discovery focus on augmentable (i.e., joinable or unionable) datasets [15, 27], whereas recent dataset search platforms [17, 41, 43, 49] are based on data augmentation *for ML tasks*. However, none address differential privacy concerns.

Differential Privacy for Databases. Differentially private databases can queries over multiple table [34, 38, 59] . They use GDP to query results over joins and unions of user and provider datasets. Among all database operations, join is notoriously challenging for DP due to the exponential growth in sensitivity along the join path. FDP may offer a solution: rather than adding noise to the entire join query, FDP decomposes the query into small statistics with bounded sensitivity, which are then combined to complete the query execution.

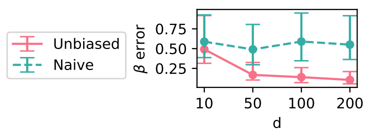
Federated ML. These methods let each untrusted party compute and privatize their local gradients for horizontal [52, 53, 58, 63] or vertical [32, 56] federated ML, which are then combined to train the final model. However, it focuses on training a single model, and the gradient is specific to that model. In contrast, data search repeatedly trains new models to evaluate candidate augmentations, which still requires splitting the privacy budget.

Differentially Private Sufficient Statistics. Previous research has used sufficient statistics [33] to compute differentially generalized linear models and then applied perturbations [39] to ensure privacy. Other studies [57] have comprehensively examined various GDP mechanisms for linear regression, including objective perturbation, posterior sampling, subsample-and-aggregate, noisy SGD, and sufficient statistics perturbation. Of these, sufficient statistics perturbation combined with regularization has the best performance. However, these studies only consider ML on a single dataset.

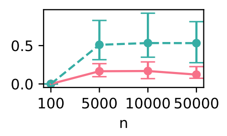
Factorized ML. Factorized ML decomposes ML models into semi-ring queries, designs algebraic operators to combine them and achieves asymptotically lower time complexity. They support popular ML models like ridge regression, decision stump [50], support vector machine [36], and factorization machine [50]. However, none of them support differentially private ML.

7 CONCLUSIONS

We present *Saibot*, a differentially private data search platform that help user identify useful datasets in a large data corpus to improve ML performance by augmentation. *Saibot* employs FDP, a novel mechanism for computing sufficient semi-ring statistics that are privatized once, and can be reused to train and evaluate models for unlimited join and union combinations. We further investigate the duplication and independence issues in many-to-many joins, suggest an unbiased estimator for it, explore parameter bounds for linear regression models, and create an optimization technique that redistributes noise to reduce its impact on the model. Our evaluation on a NYC dataset corpus with >300 datasets shows that FDP achieves an $r2$ score close (~50−90%) to that of a non-private search, outperforming other mechanisms (LDP, GDP, SDP) with a significantly lower $r2$ score <0.02.



(a) β error varying d.



(b) β error varying n.

Figure 10: β error for naive and unbiased estimators over join. The unbiased estimator dominates the naive one.

REFERENCES

[1] [n. d.]. 2013 - 2018 School ELA REsults. https://data.cityofnewyork.us/Education/2013-2018-School-ELA-REsults/qkpp-pbi8.

[2] [n. d.]. 2013 -2018 School Math Results. https://data.cityofnewyork.us/Education/2013-2018-School-Math-Results/m27t-ht3h.

[3] [n. d.]. 2013-16 School ELA Data Files By Grade - Gender. https://data.cityofnewyork.us/Education/2013-16-School-ELA-Data-Files-By-Grade-Gender/436j-ja87.

[4] [n. d.]. 2014-15 To 2016-17 School- Level NYC Regents Report For All Variables. https://data.cityofnewyork.us/Education/2014-15-To-2016-17-School-Level-NYC-Regents-Report/csps-2ne9/.

[5] [n. d.]. 2016-2017 Graduation Outcomes School. https://data.cityofnewyork.us/Education/2016-2017-Graduation-Outcomes-School/nb39-jx2v.

[6] [n. d.]. California Consumer Privacy Act. https://oag.ca.gov/privacy/ccpa.

[7] [n. d.]. The Family Educational Rights and Privacy Act (FERPA). https://studentprivacy.ed.gov/.

[8] [n. d.]. Health Insurance Portability and Accountability Act of 1996 (HIPAA). https://www.cdc.gov/phlp/publications/topic/hipaa.html.

[9] 2018. 2018 reform of EU data protection rules. https://ec.europa.eu/commission/sites/beta-political/files/data-protection-factsheet-changes_en.pdf.

[10] 2022. CMS Data. https://data.cms.gov/.

[11] 2022. NYC Open Data. https://opendata.cityofnewyork.us/.

[12] Mahmoud Abo Khamis, Hung Q Ngo, and Atri Rudra. 2016. FAQ: questions asked frequently. In *Proceedings of the 35th ACM SIGMOD-SIGACT-SIGAI Symposium on Principles of Database Systems*. 13–28.

[13] Daniel G. Alabi. 2022. *The Algorithmic Foundations of Private Computational Social Science*. Ph. D. Dissertation. Harvard University.

[14] Avrim Blum, Katrina Ligett, and Aaron Roth. 2013. A learning theory approach to noninteractive database privacy. *Journal of the ACM (JACM)* 60, 2 (2013), 1–25.

[15] Sonia Castelo, Rémi Rampin, Aécio Santos, Aline Bessa, Fernando Chirigati, and Juliana Freire. 2021. Auctus: a dataset search engine for data discovery and augmentation. *Proceedings of the VLDB Endowment* 14, 12 (2021), 2791–2794.

[16] Xiaojun Chen, Guowen Yuan, Feiping Nie, and Joshua Zhexue Huang. 2017. Semi-supervised Feature Selection via Rescaled Linear Regression.. In *IJCAI*, Vol. 2017. 1525–1531.

[17] Nadiia Chepurko, Ryan Marcus, Emanuel Zgraggen, Raul Castro Fernandez, Tim Kraska, and David Karger. 2020. ARDA: automatic relational data augmentation for machine learning. *arXiv preprint arXiv:2003.09758* (2020).

[18] Bolin Ding, Janardhan Kulkarni, and Sergey Yekhanin. 2017. Collecting telemetry data privately. *Advances in Neural Information Processing Systems* 30 (2017).

[19] Cynthia Dwork, Krishnaram Kenthapadi, Frank McSherry, Ilya Mironov, and Moni Naor. 2006. Our data, ourselves: Privacy via distributed noise generation. In *Annual international conference on the theory and applications of cryptographic techniques*. Springer, 486–503.

[20] Cynthia Dwork, Frank McSherry, Kobbi Nissim, and Adam Smith. 2006. Calibrating noise to sensitivity in private data analysis. In *Theory of Cryptography: Third Theory of Cryptography Conference, TCC 2006, New York, NY, USA, March 4-7, 2006. Proceedings 3*. Springer, 265–284.

[21] Cynthia Dwork, Frank McSherry, Kobbi Nissim, and Adam D. Smith. 2006. Calibrating Noise to Sensitivity in Private Data Analysis. In *Theory of Cryptography, Third Theory of Cryptography Conference, TCC 2006, New York, NY, USA, March 4-7, 2006, Proceedings*. 265–284.

[22] Cynthia Dwork, Guy N Rothblum, and Salil Vadhan. 2010. Boosting and differential privacy. In *2010 IEEE 51st Annual Symposium on Foundations of Computer Science*. IEEE, 51–60.

[23] Cynthia Dwork, Adam Smith, Thomas Steinke, and Jonathan Ullman. 2017. Exposed! a survey of attacks on private data. *Annual Review of Statistics and Its Application* 4 (2017), 61–84.

[24] Úlfar Erlingsson, Vitaly Feldman, Ilya Mironov, Ananth Raghunathan, Kunal Talwar, and Abhradeep Thakurta. 2019. Amplification by shuffling: From local to central differential privacy via anonymity. In *Proceedings of the Thirtieth Annual ACM-SIAM Symposium on Discrete Algorithms*. SIAM, 2468–2479.

[25] Úlfar Erlingsson, Vasyl Pihur, and Aleksandra Korolova. 2014. Rappor: Randomized aggregatable privacy-preserving ordinal response. In *Proceedings of the 2014 ACM SIGSAC conference on computer and communications security*. 1054–1067.

[26] Vitaly Feldman, Audra McMillan, and Kunal Talwar. 2022. Hiding among the clones: A simple and nearly optimal analysis of privacy amplification by shuffling. In *2021 IEEE 62nd Annual Symposium on Foundations of Computer Science (FOCS)*. IEEE, 954–964.

[27] Raul Castro Fernandez, Ziawasch Abedjan, Famien Koko, Gina Yuan, Samuel Madden, and Michael Stonebraker. 2018. Aurum: A data discovery system. In *2018 IEEE 34th International Conference on Data Engineering (ICDE)*. IEEE, 1001–1012.

[28] Richard Frank, Flavia Moser, and Martin Ester. 2007. A method for multi-relational classification using single and multi-feature aggregation functions. In *Knowledge Discovery in Databases: PKDD 2007: 11th European Conference on Principles and Practice of Knowledge Discovery in Databases, Warsaw, Poland, September 17-21, 2007. Proceedings 11*. Springer, 430–437.

[29] Todd J Green, Grigoris Karvounarakis, and Val Tannen. 2007. Provenance semirings. In *Proceedings of the twenty-sixth ACM SIGMOD-SIGACT-SIGART symposium on Principles of database systems*. 31–40.

[30] Hongyu Guo and Herna L Viktor. 2008. Multirelational classification: a multiple view approach. *Knowledge and Information Systems* 17 (2008), 287–312.

[31] Moritz Hardt and Guy N Rothblum. 2010. A multiplicative weights mechanism for privacy-preserving data analysis. In *2010 IEEE 51st annual symposium on foundations of computer science*. IEEE, 61–70.

[32] Stephen Hardy, Wilko Henecka, Hamish Ivey-Law, Richard Nock, Giorgio Patrini, Guillaume Smith, and Brian Thorne. 2017. Private federated learning on vertically partitioned data via entity resolution and additively homomorphic encryption. *arXiv preprint arXiv:1711.10677* (2017).

[33] Jonathan Huggins, Ryan P Adams, and Tamara Broderick. 2017. PASS-GLM: polynomial approximate sufficient statistics for scalable Bayesian GLM inference. *Advances in Neural Information Processing Systems* 30 (2017).

[34] Noah Johnson, Joseph P Near, and Dawn Song. 2018. Towards practical differential privacy for SQL queries. *Proceedings of the VLDB Endowment* 11, 5 (2018), 526–539.

[35] Peter Kairouz, Keith Bonawitz, and Daniel Ramage. 2016. Discrete distribution estimation under local privacy. In *International Conference on Machine Learning*. PMLR, 2436–2444.

[36] Mahmoud Abo Khamis, Ryan R Curtin, Benjamin Moseley, Hung Q Ngo, XuanLong Nguyen, Dan Olteanu, and Maximilian Schleich. 2020. Functional Aggregate Queries with Additive Inequalities. *ACM Transactions on Database Systems (TODS)* 45, 4 (2020), 1–41.

[37] Mahmoud Abo Khamis, Hung Q Ngo, XuanLong Nguyen, Dan Olteanu, and Maximilian Schleich. 2018. AC/DC: in-database learning thunderstruck. In *Proceedings of the second workshop on data management for end-to-end machine learning*. 1–10.

[38] Ios Kotsogiannis, Yuchao Tao, Xi He, Maryam Fanaeepour, Ashwin Machanavajjhala, Michael Hay, and Gerome Miklau. 2019. Privatesql: a differentially private sql query engine. *Proceedings of the VLDB Endowment* 12, 11 (2019), 1371–1384.

[39] Tejas Kulkarni, Joonas Jälkö, Antti Koskela, Samuel Kaski, and Antti Honkela. 2021. Differentially private bayesian inference for generalized linear models. In *International Conference on Machine Learning*. PMLR, 5838–5849.

[40] Jaewoo Lee and Chris Clifton. 2011. How much is enough? choosing ε for differential privacy. In *Information Security: 14th International Conference, ISC 2011, Xi'an, China, October 26-29, 2011. Proceedings 14*. Springer, 325–340.

[41] Yifan Li, Xiaohui Yu, and Nick Koudas. 2021. Data acquisition for improving machine learning models. *Proceedings of the VLDB Endowment* 14, 10 (2021), 1832–1844.

[42] Andrzej Maćkiewicz and Waldemar Ratajczak. 1993. Principal components analysis (PCA). *Computers & Geosciences* 19, 3 (1993), 303–342.

[43] Fatemeh Nargesian, Abolfazl Asudeh, and HV Jagadish. 2022. Responsible Data Integration: Next-generation Challenges. In *Proceedings of the 2022 International Conference on Management of Data*. 2458–2464.

[44] Joseph P Near, Xi He, et al. 2021. Differential Privacy for Databases. *Foundations and Trends® in Databases* 11, 2 (2021), 109–225.

[45] Dan Olteanu and Jakub Závodnỳ. 2015. Size bounds for factorised representations of query results. *ACM Transactions on Database Systems (TODS)* 40, 1 (2015), 1–44.

[46] Judea Pearl. 2022. Comment: understanding Simpson's paradox. In *Probabilistic and Causal Inference: The Works of Judea Pearl*. 399–412.

[47] Wahbeh Qardaji, Weining Yang, and Ninghui Li. 2014. Priview: practical differentially private release of marginal contingency tables. In *Proceedings of the 2014 ACM SIGMOD international conference on Management of data*. 1435–1446.

[48] Nithya Sambasivan, Shivani Kapania, Hannah Highfill, Diana Akrong, Praveen Paritosh, and Lora M Aroyo. 2021. "Everyone wants to do the model work, not the data work": Data Cascades in High-Stakes AI. In *proceedings of the 2021 CHI Conference on Human Factors in Computing Systems*. 1–15.

[49] Aécio Santos, Aline Bessa, Christopher Musco, and Juliana Freire. 2022. A sketch-based index for correlated dataset search. In *2022 IEEE 38th International Conference on Data Engineering (ICDE)*. IEEE, 2928–2941.

[50] Maximilian Schleich, Dan Olteanu, Mahmoud Abo Khamis, Hung Q Ngo, and XuanLong Nguyen. 2019. A layered aggregate engine for analytics workloads. In *Proceedings of the 2019 International Conference on Management of Data*. 1642–1659.

[51] Maximilian Schleich, Dan Olteanu, and Radu Ciucanu. 2016. Learning linear regression models over factorized joins. In *Proceedings of the 2016 International Conference on Management of Data*. 3–18.

[52] Reza Shokri and Vitaly Shmatikov. 2015. Privacy-preserving deep learning. In *Proceedings of the 22nd ACM SIGSAC conference on computer and communications security*. 1310–1321.

[53] Stacey Truex, Ling Liu, Ka-Ho Chow, Mehmet Emre Gursoy, and Wenqi Wei. 2020. LDP-Fed: Federated learning with local differential privacy. In *Proceedings of the Third ACM International Workshop on Edge Systems, Analytics and Networking*. 61–66.

[54] Haleh Vafaie, Ibrahim F Imam, et al. 1994. Feature selection methods: genetic algorithms vs. greedy-like search. In *Proceedings of the international conference on fuzzy and intelligent control systems*, Vol. 51. 28.

[55] Roman Vershynin. 2018. *High-dimensional probability: An introduction with applications in data science*. Vol. 47. Cambridge university press.

[56] Chang Wang, Jian Liang, Mingkai Huang, Bing Bai, Kun Bai, and Hao Li. 2020. Hybrid differentially private federated learning on vertically partitioned data. *arXiv preprint arXiv:2009.02763* (2020).

[57] Yu-Xiang Wang. 2018. Revisiting differentially private linear regression: optimal and adaptive prediction & estimation in unbounded domain. *arXiv preprint arXiv:1803.02596* (2018).

[58] Kang Wei, Jun Li, Ming Ding, Chuan Ma, Howard H Yang, Farhad Farokhi, Shi Jin, Tony QS Quek, and H Vincent Poor. 2020. Federated learning with differential privacy: Algorithms and performance analysis. *IEEE Transactions on Information Forensics and Security* 15 (2020), 3454–3469.

[59] Royce J Wilson, Celia Yuxin Zhang, William Lam, Damien Desfontaines, Daniel Simmons-Marengo, and Bryant Gipson. 2019. Differentially private SQL with bounded user contribution. *arXiv preprint arXiv:1909.01917* (2019).

[60] Genqiang Wu, Xianyao Xia, and Yeping He. 2017. Achieving Dalenius' Goal of Data Privacy with Practical Assumptions. *arXiv preprint arXiv:1703.07474* (2017).

[61] Jia Xu, Zhenjie Zhang, Xiaokui Xiao, Yin Yang, Ge Yu, and Marianne Winslett. 2013. Differentially private histogram publication. *The VLDB journal* 22 (2013), 797–822.

[62] Raul Castro Fernandez Eugene Wu Zezhou Huang, Pranav Subramaniam. 2022. (Technical Report) Kitana: Efficient Data Augmentation Search for AutoML. https://www.dropbox.com/s/u6d8tvg5e4eatd6/kitana.pdf?dl=0.

[63] Yang Zhao, Jun Zhao, Mengmeng Yang, Teng Wang, Ning Wang, Lingjuan Lyu, Dusit Niyato, and Kwok-Yan Lam. 2020. Local differential privacy-based federated learning for internet of things. *IEEE Internet of Things Journal* 8, 11 (2020), 8836–8853.

A FDP SENSITIVITY

For union, query $q_u : \mathcal{D}^n \longrightarrow s$ returns a vector s containing the sum of i-order polynomial semiring across all tuples, for $i \in \{0, \ldots, k\}$. Let $t_1, t_2 \in \mathcal{D}^n$ and t_1^i, t_2^i be vectors of i-order polynomial with respect to $t_1[f_1], \ldots, t_1[f_n]$ and $t_2[f_1], \ldots, t_2[f_n]$, respectively. Let $\sigma(k)$ denote the set of series $[k_1, \ldots, k_d]$ such that $k_i \in \mathbb{N}$ and $\sum k_i = k$. The squared distance between t_1^k and t_2^k can be computed as

$$\begin{aligned}
&\sum_{\substack{[k_1,\ldots,k_d]\\ \in\sigma(k)}} (\prod_{i=1}^{d} t_1[f_i]^{k_i} - \prod_{i=1}^{d} t_2[f_i]^{k_i})^2 \\
\le &\sum_{\substack{[k_1,\ldots,k_d]\\ \in\sigma(k)}} \binom{k}{k_1, \ldots, k_d} (\prod_{i=1}^{d} t_1[f_i]^{k_i} - \prod_{i=1}^{d} t_2[f_i]^{k_i})^2 \\
= &\sum_{\substack{[k_1,\ldots,k_d]\\ \in\sigma(k)}} \binom{k}{k_1, \ldots, k_d} (\prod_{i=1}^{d} t_1[f_i]^{2k_i} + \prod_{i=1}^{d} t_2[f_i]^{2k_i} - 2\prod_{i=1}^{d} (t_1[f_i]t_2[f_i])^{k_i})
\end{aligned}$$

By multinomial theorem, we may rewrite the last equation as

$$\begin{aligned}
&(\sum_{i=1}^{d} t_1[f_i]^2)^k + (\sum_{i=1}^{d} t_2[f_i]^2)^k - 2(\sum_{i=1}^{d} t_1[f_i]t_2[f_i])^k \\
&\le 2B^{2k} - 2(\sum_{i=1}^{d} t_1[f_i]t_2[f_i])^k
\end{aligned}$$

That is, when k is even, the latter term is strictly positive. Hence $\|t_1^k - t_2^k\|^2 \le 2B^{2k}$. Let $D_1, D_2 \in \mathcal{D}^n$ be two neighbouring datasets differ in one tuple, t_1 and t_2. The sensitivity of $q(\cdot)$ can be computed as

$$\Delta_{q_u} = \max \|q_u(D_1) - q_u(D_2)\| \le \sqrt{\sum_{i=1}^{k} 1_{2\mathbb{Z}+1}(i)4B^{2i} + 1_{2\mathbb{Z}}(i)2B^{2i}}$$

For join, let there be ℓ join keys, query $q_j : \mathcal{D}^n \longrightarrow s$ concatenates vectors returned by q_u on each partition of tuples for each join key. Consider two cases: 1, t^a and t^b are in the same bin 2, t^a and t^b are in different bins. For the former case,

$$\begin{aligned}
\Delta_q &= \max \|q_j(D_1) - q_j(D_2)\|_2 \\
&\le \max \|q_j(t_1) - q_j(t_2)\|_2 \\
&= \sqrt{\sum_{i=1}^{k} 1_{2\mathbb{Z}+1}(i)4B^{2i} + 1_{2\mathbb{Z}}(i)2B^{2i}}
\end{aligned}$$

For the later case,

$$\begin{aligned}
\Delta_q &= \max \|q_j(D_1) - q_j(D_2)\|_2 \\
&= \max \sqrt{\sum_{i}^{k} \|t_1^i\|^2 + \sum_{i}^{k} \|t_2^i\|^2} \\
&\le \sqrt{\max \sum_{i}^{k} \|t_1^i\|^2 + \max \sum_{i}^{k} \|t_2^i\|^2} \\
&\le \sqrt{2 \sum_{i=0}^{k} B^{2i}}
\end{aligned}$$

Hence, $\Delta_q = \max(\sqrt{\sum_{i=1}^{k} 1_{2\mathbb{Z}+1}(i)4B^{2i} + 1_{2\mathbb{Z}}(i)2B^{2i}}, \sqrt{2\sum_{i=0}^{k} B^{2i}})$

B ERROR ANALYSIS

For a single buyer, with relation R_0, along with m sellers, $R_1, \ldots, R_m$, where $m \le n_{seller}$ and i be any integer from 1 to k. LDP computes $[f, f^2, \ldots, f^k]$ for each tuple and adds noise to each of them. By similar analysis to that of FDP, LDP's sensitivity is the same as Δ in FDP for both union and join cases. Hence, for each tuple, t, from any dataset in $R_0, \ldots, R_m$, $t[\widetilde{f^i}] \sim t[f^i] + \mathcal{N}(0, \sqrt{2\ln(1.25/\delta)}\Delta/\epsilon)$ The empirical expectation of f^i over the union of datasets can be estimated as summing on all tuples among $m + 1$ datasets. That is,

$$\widetilde{f^i_{\text{LDP}}} = \frac{1}{n}\left(\sum_{j=0}^{m} \sum_{t\in R_j} t[f^i]\right) + X_i \quad X_i \sim \mathcal{N}(0, \sqrt{2\ln(1.25/\delta)}\Delta/\sqrt{n}\epsilon)$$

Putting everything together, and by assumption that k is small, we have

$$\begin{aligned}
E[\|s'_{\text{LDP}} - \hat{s}\|] &= E\left[\sqrt{\sum_{i=1}^{k}\left(\widetilde{f^i_{\text{LDP}}} - \frac{1}{n}\left(\sum_{j=0}^{m}\sum_{t\in R_j} t[f^i]\right)\right)^2}\right] \\
&\le \sqrt{E\left[\sum_{i=1}^{k}\left(\widetilde{f^i_{\text{LDP}}} - \frac{1}{n}\left(\sum_{j=0}^{m}\sum_{t\in R_j} t[f^i]\right)\right)^2\right]} \\
&= \sqrt{\sum_{i=1}^{k} E[X_i^2]} \\
&= O(\Delta/\sqrt{n}\epsilon)
\end{aligned}$$

For FDP, the only difference is that

$$\widetilde{f^i_{\text{FDP}}} \sim \frac{1}{n}\left(\sum_{j=0}^{m} \sum_{t\in R_j} t[f^i]\right) + X_i \quad X_i \sim \mathcal{N}(0, \sqrt{2\ln(1.25/\delta)}\Delta/n\epsilon)$$

Following the same line of derivation,

$$E[\|s'_{\text{FDP}} - \hat{s}\|] = O(\Delta/n\epsilon)$$

However, GDP needs to account for any possible combination of a single buyer and a subset of sellers, where each party's privacy needs to be protected. Specifically, each buyer appears in $2^{n_{seller}} - 1$ combinations, since each buyer requires at least one seller. On the other hand, for a fixed buyer, each seller is involved in $2^{n_{seller}-1}$ combinations. Hence, each seller will appear in $n_{buyer}2^{n_{seller}-1}$

combinations in total. Because each seller and buyer have privacy budget (ϵ, δ), in order to provide privacy guarantees for each party in any combination, the amount of privacy budget spent on perturbing pre-normalized s is $\epsilon' = \min(\epsilon/(2^{n_{seller}}-1), \epsilon/n_{buyer}2^{n_{seller}-1})$ and $\delta' = \min(\delta/(2^{n_{seller}}-1), \delta/n_{buyer}2^{n_{seller}-1})$

$$\widetilde{f^i_{\texttt{GDP}}} = \frac{1}{n}\left(\sum_{j=0}^{m}\sum_{t\in R_j} t[f^i]\right) + X_i \quad X_i \sim \mathcal{N}(0, \sqrt{2\ln(1.25/\delta')}\Delta/n\epsilon')$$

Based on the same line of analysis above

$$E[\|s'_{\texttt{GDP}} - \hat{s}\|] = O(n_{buyer}2^{n_{seller}-1}\Delta/n\epsilon)$$

Now we consider SDP-1, based on [24], it suffice to guarantee $\epsilon/\sqrt{n}$-DP for local responses to achieve (ϵ,δ)-DP from the central's perspective, where each tuple t in $R_0, \ldots, R_m$ satisfy that $t[\widetilde{f^i}] \sim t[f^i] + \text{Lap}(\Delta/\sqrt{n}\epsilon)$. Then we have

$$\widetilde{f^i_{\texttt{SDP-1}}} = \frac{1}{n}\sum_{j=0}^{m}\sum_{t\in R_j}(t[f^i] + X_{j,t}) \quad X_{j,t} \sim \text{Lap}(\Delta/\sqrt{n}\epsilon)$$

Then, we have

$$E[\|s'_{\texttt{SDP-1}} - \hat{s}\|] = E\left[\sqrt{\sum_{i=1}^{k}\left(\widetilde{f^i_{\texttt{SDP-1}}} - \frac{1}{n}\left(\sum_{j=0}^{m}\sum_{t\in R_j} t[f^i]\right)\right)^2}\right]$$
$$\leq \sqrt{\sum_{i=1}^{k} E\left[\left(\frac{1}{n}\sum_{j=0}^{m}\sum_{t\in R_j} X_{j,t}\right)^2\right]}$$

Since $E\left[\left(\frac{1}{n}\sum_{j=0}^{m}\sum_{t\in R_j} X_{j,t}\right)\right] = 0$, it follows that

$$E\left[\left(\frac{1}{n}\sum_{j=0}^{m}\sum_{t\in R_j} X_{j,t}\right)^2\right] = Var\left(\frac{1}{n}\sum_{j=0}^{m}\sum_{t\in R_j} X_{j,t}\right) = \frac{\Delta^2}{n^2\epsilon^2}$$

Substituting back to the equation, and based on assumption that k is small, we have

$$E[\|s'_{\texttt{SDP-1}} - \hat{s}\|] = O(\Delta/n\epsilon)$$

For SDP-2, just like GDP, it also needs to account for all possible combination of a single buyer and any subsets of sellers in the centralized shuffler. However, the differences are that SDP-2 allows each combination's privacy guarantee to be amplified by an amount of $O(\sqrt{n})$, and that SDP-2 draw random noises from Laplace distribution instead of Gaussian distribution. That is,

$$\widetilde{f^i_{\texttt{SDP-2}}} = \frac{1}{n}\sum_{j=0}^{m}\sum_{t\in R_j}(t[f^i] + X_{j,t}) \quad X_{j,t} \sim \text{Lap}(\Delta/\sqrt{n}\epsilon')$$

Hence, the expected utility can be computed followint the same line of derivation of SDP-1. That is

$$E[\|s'_{\texttt{SDP-2}} - \hat{s}\|] = O(\Delta/n\epsilon') = O(n_{buyer}2^{n_{seller}-1}\Delta/n\epsilon)$$

C UNBIASED PROOF

We make the simplifying assumption that J is uniformly distributed: if $d = |dom(J)|$, then each $j \in J$ appears n/d times in R.

Proposition 8 (Expected s over $R_{\bowtie}$). *Assume that $R_{\bowtie}$ is the population. For any other 1,2-order polynomial p,*

$$E[p] = s[p]/n$$

where c is the count (0-order polynomial). Then $E[p]$ is the expected s over $R_{\bowtie}$.

Proposition 9 (Unbiased Estimator of s over R).

$$\widehat{E[p]} = \begin{cases} f_1f_2 = \frac{1-n}{1-d}\frac{s[f_1f_2]}{s[c]} + \frac{n-d}{1-d}\frac{s[f_1]}{s[c]}\frac{s[f_2]}{s[c]} \\ \qquad\qquad\qquad \text{for } f_1 \in F_1, f_2 \in F_2 \\ p = s[p]/s[c] \quad \text{for any other polynomial } p \end{cases}$$

$\hat{s}$ is an unbiased estimator of s.

Proof. We demonstrate that, for any 1,2-order polynomial where features are from the same relation, $E[s[p]/s[c]] = E[p]$.

$$s[c] = n \cdot n/d$$
$$E[s[f]/s[c]] = E[(\sum_{t\in R} t[f]\cdot n/d)/(n\cdot n/d)] = \sum_{t\in R} E[t[f]]/n = E[f]$$
$$E[s[f_1f_2]/s[c]] = E[(\sum_{t\in R} t[f_1]\cdot t[f_2]\cdot n/d)/(n\cdot n/d)]$$
$$= \sum_{t\in R} E[t[f_1]\cdot t[f_2]]/n = E[f_1f_2]$$

The first equality is because for each join key, the cartesian product is computed, leading to duplication of tuples with the same join key in both tables by n/d times. The count is also increased by n/d, thus resulting in the equality $s[p]/s[c] = E[p]$.

However, this equality does not hold for the f_1f_2, where f_1 and f_2 are from different relations. In this case, f_1 from R_1 is paired with all f_2 from R_2 with the same join key, but the information about which f_2 is paired with f_1 in original R is lost. Nonetheless, we can still estimate $E[f_1f_2]$ by exploiting the covariance across groups.

We first analyze $E[f_1f_2]$ for a single join key value j. We use notation s^j to denote the polynomial semi-ring for the join key value k. Consider random variable of the average:

$$s_1^j = s^j[f_1]/s^j[c] = \left(\sum_{t\in\sigma_j(R)} t[f_1]\cdot n/d\right)/(n/d)^2 = \sum_{t\in\sigma_j(R)}\frac{t[f_1]}{n/d}$$
$$s_2^j = s^j[f_2]/s^j[c] = \left(\sum_{t\in\sigma_j(R)} t[f_2]\cdot n/d\right)/(n/d)^2 = \sum_{t\in\sigma_j(R)}\frac{t[f_2]}{n/d}$$

s_1^j and s_2^j can be understood as the mean of f_1 and f_2 from the sample $\sigma_j(R)$. It is obvious that $E[s_1^j] = E[f_1]$ and $E[s_2^j] = E[f_2]$.

From the definition of covariance, we have:

$$E[s_1^js_2^j] = cov(s_1^j, s_2^j) + E[s_1^j]E[s_2^j]$$
$$= cov\left(\sum_{t\in\sigma_j(R)}\frac{t[f_1]}{n/d}, \sum_{t\in\sigma_j(R)}\frac{t[f_2]}{n/d}\right) + E[f_1]E[f_2]$$

We next compute the *cov*:

$$\begin{aligned} cov\left(\sum_{t\in\sigma_j(R)} \frac{t[f_1]}{n/d}, \sum_{t\in\sigma_j(R)} \frac{t[f_2]}{n/d}\right) &= \frac{d^2}{n^2} \sum_{\substack{t_1\in\sigma_j(R)\\ t_2\in\sigma_j(R)}} cov(t_1[f_1], t_2[f_2]) \\ &= \frac{d^2}{n^2} \sum_{t\in\sigma_j(R)} cov(t[f_1], t[f_2]) \\ &= \frac{d}{n} cov(f_1, f_2) \end{aligned}$$

The first line is by the property of covariance and the second line is by the independence between tuples. Therefore,

$$E[s_1^j s_2^j] = \frac{d}{n} cov(f_1, f_2) + E[f_1]E[f_2]$$

Next, consider the random variables across join keys:

$$\begin{aligned} s_1 &= s[f_1]/s[c] = \sum_{t\in R} t[f_1]/n \\ s_2 &= s[f_2]/s[c] = \sum_{t\in R} t[f_2]/n \\ s_{1,2} &= s[f_1 f_2]/s[c] = \sum_{j\in dom(J)} s_1^j \cdot s_2^j/d \end{aligned}$$

where s_1 and s_2 are the average across join keys. $s_{1,2}$ is the average products across join keys. It is obvious that $E[s_1] = E[f_1], E[s_2] = E[f_2]$. We next study $E[s_1 s_2]$ and $E[s_{1,2}]$:

$$\begin{aligned} E[s_1 s_2] &= cov(s_1, s_2) + E[s_1^j]E[s_2^j] \\ &= cov\left(\sum_{t\in R} t[f_1]/n, \sum_{t\in R} t[f_2]/n\right) + E[f_1]E[f_2] \end{aligned}$$

Similar as before,

$$\begin{aligned} cov\left(\sum_{t\in R} t[f_1]/n, \sum_{t\in R} t[f_2]/n\right) &= \frac{1}{n^2} \sum_{\substack{t_1\in R\\ t_2\in R}} cov(t_1[f_1], t_2[f_2]) \\ &= \frac{1}{n^2} \sum_{t\in R} cov(t[f_1], t[f_2]) = \frac{1}{n} cov(f_1, f_2) \end{aligned}$$

Therefore:

$$E[s_1 s_2] = \frac{1}{n} cov(f_1, f_2) + E[f_1]E[f_2]$$

Finally,

$$E[s_{1,2}] = \sum_{j\in dom(J)} E[s_1^j \cdot s_2^j]/d = \frac{d}{n} cov(f_1, f_2) + E[f_1]E[f_2]$$

Putting everything together, we show that $\frac{1-n}{1-d} s_{1,2} + \frac{n-d}{1-d} s_1 \cdot s_2$ is an unbiased estimator of $E[f_1 f_2]$:

$$\begin{aligned} E[\frac{1-n}{1-d} s_{1,2} + \frac{n-d}{1-d} s_1 \cdot s_2] =& \frac{1-n}{1-d} E[s_{1,2}] + \frac{n-d}{1-d} E[s_1 s_2] \\ =& \frac{1-n}{1-d}(\frac{d}{n} cov(f_1, f_2) + E[f_1]E[f_2]) + \\ & \frac{n-d}{1-d}(\frac{1}{n} cov(f_1, f_2) + E[f_1]E[f_2]) \\ =& cov(f_1, f_2) + E[f_1]E[f_2] = E[f_1 f_2] \end{aligned}$$

The first line is by the linearity of expectation, and the last line is by the definition of covariance.

Table 1: Notation

Notation	Description
R_i	relations of providers/requesters.
n	size of each relation.
$J, Dom(J), d$	join key, domain of join key, domain size.
B	the ℓ_2 distance upper bound of each tuple in each relation.
$\widehat{\sigma^2_{\mathrm{x}}}, \widehat{\sigma^2_{\mathrm{xy}}}$	the empirical estimation of the variance and covariance.
$\widetilde{\sigma^2_{\mathrm{x}}}, \widetilde{\sigma^2_{\mathrm{xy}}}$	the privatized empirical estimation of the variance and covariance.
B_1, B_2	$1-p$ confidence bound on $\lvert\widehat{\sigma^2_{\mathrm{x}}} - \widetilde{\sigma^2_{\mathrm{x}}}\rvert$ and $\lvert\widehat{\sigma^2_{\mathrm{xy}}} - \widetilde{\sigma^2_{\mathrm{xy}}}\rvert$.

□

D CONFIDENCE BOUND OF LINEAR REGRESSION

Let $\sigma = \sqrt{2\ln(1.25/\delta)}\Delta/\epsilon$ where $\Delta = O(B^2)$. We are interested in $n, B \longrightarrow \infty$ and $\epsilon, \delta, p \longrightarrow 0$ in our analysis. Hence $\sigma = O\left(\frac{B^2\sqrt{\ln(1/\delta)}}{\epsilon}\right)$

the privatized empirical expectation of the moments are defined as $\widetilde{E[X]} = \widehat{E[X]} + e_1, \widetilde{E[X^2]} = \widehat{E[X^2]} + e_2, \widetilde{E[XY]} = \widehat{E[XY]} + e_3$ and $\widetilde{E[Y]} = \widehat{E[Y]} + e_4$. Then, we have $e_1, e_2, e_3, e_4 \sim \mathcal{N}(0, \sigma^2/n^2)$

Lemma D.1 (High-probability bound on $\widetilde{\sigma_x}$). *Given* $\widehat{\sigma^2_{\mathrm{x}}} = \widehat{E[X^2]} - \widehat{E[X]}^2$ *and* $\widetilde{\sigma^2_{\mathrm{x}}} = \widetilde{E[X^2]} - \widetilde{E[X]}^2$, *with probability at least* $1-p$, $\lvert\widehat{\sigma^2_{\mathrm{x}}} - \widetilde{\sigma^2_{\mathrm{x}}}\rvert = O(B_1)$ *where*

$$B_1 = \frac{B^4 \ln(1/\delta)\ln(1/p)}{\epsilon^2 n}$$

Proof. By assumption that each tuple's ℓ_2 norm is bounded by B, each feature must also be bounded by B. Based on Gaussian tail bound, with probability at least $1-p/4$, $|e_i| \leq \sigma\sqrt{2\ln(8/p)}/n$.

$$\begin{aligned} \lvert\widehat{\sigma^2_{\mathrm{x}}} - \widetilde{\sigma^2_{\mathrm{x}}}\rvert &= |e_1 - 2e_2 \sum x/n - e_2^2| \\ &\leq |e_1| + |2e_2 \sum x/n| + |e_2^2| \\ &\leq \frac{\sigma\sqrt{2\ln(8/p)}}{n}\left(1 + 2B + \frac{\sigma\sqrt{2\ln(8/p)}}{n}\right) \\ &= O\left(\frac{B^2\sqrt{\ln(1/\delta)\ln(1/p)}}{\epsilon n} + \frac{B^2\ln(1/\delta)\ln(1/p)}{\epsilon^2 n^2}\right) \\ &= O\left(\frac{B^4\ln(1/\delta)\ln(1/p)}{\epsilon^2 n}\right) \end{aligned}$$

□

Similarly, D.1 can be used to derive the high probability bound on $\widetilde{\sigma^2_{xy}}$, that is

$$\lvert\widehat{\sigma^2_{\mathrm{xy}}} - \widetilde{\sigma^2_{\mathrm{xy}}}\rvert \leq B_2 = O\left(\frac{B^4\ln(1/\delta)\ln(1/p)}{\epsilon^2 n}\right)$$

Since the condition to satisfy both bounds coincide, with probability at least $1-p$, $\lvert\widehat{\sigma^2_x} - \widetilde{\sigma^2_x}\rvert \leq B_1$ and $\lvert\widehat{\sigma^2_{xy}} - \widetilde{\sigma^2_{xy}}\rvert \leq B_2$.

Let

$$\tau_1 = B_1/\widehat{\sigma_x^2} = O\left(\frac{B^4 \ln(1/\delta)\ln(1/p)}{\epsilon^2 n\widehat{\sigma_{\mathrm{x}}^2}}\right) = \tau_2$$

When $\tau_1, \tau_2 < 1$ and with probability at least $1 - p$, we may prove 4.1 as

Proof.

$$\begin{aligned}
|\hat{\beta}_x - \tilde{\beta}_x| &= \left|\frac{\widehat{\sigma_{\mathrm{xy}}^2}}{\widehat{\sigma_{\mathrm{x}}^2}} - \frac{\widetilde{\sigma_{\mathrm{xy}}^2}}{\widetilde{\sigma_{\mathrm{x}}^2}}\right| = \left|\frac{\widehat{\sigma_{\mathrm{xy}}^2}}{\widehat{\sigma_{\mathrm{x}}^2}} - \frac{\widetilde{\sigma_{\mathrm{xy}}^2}}{\widehat{\sigma_{\mathrm{x}}^2}} + \frac{\widetilde{\sigma_{\mathrm{xy}}^2}}{\widehat{\sigma_{\mathrm{x}}^2}} - \frac{\widetilde{\sigma_{\mathrm{xy}}^2}}{\widetilde{\sigma_{\mathrm{x}}^2}}\right| \\
&\le \frac{|\widetilde{\sigma_{\mathrm{xy}}^2} - \widehat{\sigma_{\mathrm{xy}}^2}|}{\widehat{\sigma_{\mathrm{x}}^2}} + \widetilde{\sigma_{\mathrm{xy}}^2} \cdot \left|\widetilde{\sigma_{\mathrm{x}}^2}^{-1} - \widehat{\sigma_{\mathrm{x}}^2}^{-1}\right| \\
&= \frac{|\widetilde{\sigma_{\mathrm{xy}}^2} - \widehat{\sigma_{\mathrm{xy}}^2}|}{\widehat{\sigma_{\mathrm{x}}^2}} + \widetilde{\sigma_{\mathrm{xy}}^2} \cdot \frac{|\widetilde{\sigma_{\mathrm{x}}^2} - \widehat{\sigma_{\mathrm{x}}^2}|}{\widetilde{\sigma_{\mathrm{x}}^2}\widehat{\sigma_{\mathrm{x}}^2}} \\
&\le \tau_2 + (\widehat{\sigma_{\mathrm{xy}}^2} + \tau_2 \cdot \widehat{\sigma_{\mathrm{x}}^2}) \frac{\tau_1 \widehat{\sigma_{\mathrm{x}}^2}}{(1 - \tau_1)(\widehat{\sigma_{\mathrm{x}}^2})^2} \\
&= \tau_2 + \frac{\tau_1}{1 - \tau_1}\left(\frac{\widehat{\sigma_{\mathrm{xy}}^2}}{\widehat{\sigma_{\mathrm{x}}^2}} + \tau_2\right) = \tau_2 + \frac{\tau_1}{1 - \tau_1}\left(\hat{\beta}_x + \tau_2\right)
\end{aligned}$$

□

Extension to Factorizd ML. The confidence bounds can be extended for factorized ML. The difference boils down to B_1, and the rest are the same. For union, let $R = R_1 \cup R_2 ... \cup R_k$ where $|R_i| = n$ and $k \longrightarrow \infty$. Then, for $e_i \sim \mathcal{N}(0, \sigma^2)$

$$\widetilde{E[x^2]} = \frac{\sum_i^k (\sum x^2 + e_i)}{kn} \sim \widehat{E[X^2]} + \mathcal{N}(0, \sigma^2/kn^2)$$

Therefore, with probability at least $1 - p/4$, $|\widetilde{E[X^2]} - \widehat{E[X^2]}| \le \sigma\sqrt{2\ln(8/p)}/\sqrt{k}n = O\left(\frac{B^2\sqrt{\ln(1/\delta)\ln(1/p)}}{\epsilon\sqrt{k}n}\right)$ (same for all other 3 moments $E[X], E[Y], E[XY]$), by minor changes in D.1, yielding new bounds on τ_1 and τ_2 as

$$\begin{aligned}
\tau_1 = B_1/\widehat{\sigma_x^2} &= O\left(\frac{B^2\sqrt{\ln(1/\delta)\ln(1/p)}}{\epsilon\sqrt{k}n\widehat{\sigma_{\mathrm{x}}^2}} + \frac{B^4\ln(1/\delta)\ln(1/p)}{\epsilon^2 kn^2\widehat{\sigma_{\mathrm{x}}^2}}\right) \\
&= O\left(\frac{B^4\ln(1/\delta)\ln(1/p)}{\epsilon^2\sqrt{k}n\widehat{\sigma_{\mathrm{x}}^2}}\right) = \tau_2
\end{aligned}$$

For join, consider $R[x, y, J] = R_1[x, J] \bowtie R_2[y, J]$ and $d = |dom(J)|$ where $d \longrightarrow n$. In contrast to union, there is additional noise added to the zero-th moment of each join key. i.e. the count of tuples within each join key. To avoid the scenario where this number is non-positive, an additional assumption is required [57] that the noise is bounded by $o(n/d)$. Then, for $e_{i,1}, e_{i,2}, e_{i,3} \sim \mathcal{N}(0, \sigma^2)$ defined as the Gaussian noise added to $\sum_{t\in R_1.i} x^2, \sum_{t\in R_1.i} x, \sum_{t\in R_2.i} y$ for each join key $i \in J$, with probability at least $1 - p/4$, $\sum_{i\in J} e_{i,j} \sim \mathcal{N}(0, d\sigma^2)$ and $\sum_{i\in J} e_{i,j} \le \sigma\sqrt{2d\ln(8/p)} = O\left(\frac{B^2\sqrt{d\ln(1/\delta)\ln(1/p)}}{\epsilon}\right)$ for $j = \{1, 2, 3\}$

$$\begin{aligned}
\widehat{E[X^2]} &= \frac{\sum_{i\in J}(\sum_{t\in R_1.i} x^2)n/d}{n^2/d} = \frac{\sum x^2}{n} \\
\widetilde{E[X^2]} &= \frac{\sum_{i\in J}\left((\sum_{t\in R_1.i} x^2) + e_{i,1}\right)(n/d + o(n/d))}{\sum_{j\in J}(n/d + o(n/d))(n/d + o(n/d))}
\end{aligned}$$

By expanding $\widetilde{E[X^2]}$, we have

$$\begin{aligned}
\widetilde{E[X^2]} &= \frac{(n/d)\sum x^2 + (n/d)\cdot\sum_{i\in J} e_{i,1} + o(n/d)\sum x^2 + o(n/d)\sum_{i\in J} e_{i,1}}{n^2/d + 2n\cdot o(n/d) + d\cdot o(n^2/d^2)} \\
&= \frac{(\frac{\sum x^2}{n} + (\sum_{i\in J} e_{i,1})/n)(1 + o(n/d)(d/n))}{1 + 2(d/n)\cdot o(n/d) + (d^2/n^2)\cdot o(n^2/d^2)} \\
&= \frac{\frac{\sum x^2}{n} + (\sum_{i\in J} e_{i,1})/n + o(1)(\frac{\sum x^2}{n} + (\sum_{i\in J} e_{i,1})/n)}{1 + o(1)} \\
&= (1 + o(1))\left(\frac{\sum x^2}{n} + (\sum_{i\in J} e_{i,1})/n + o(1)(\frac{\sum x^2}{n} + (\sum_{i\in J} e_{i,1})/n)\right)
\end{aligned}$$

Hence

$$\begin{aligned}
|\widetilde{E[X^2]} - \widehat{E[X^2]}| &= O\left(\frac{\sum_{i\in J} e_{i,1}}{n}\right) \\
&= O\left(\frac{B^2\sqrt{d\ln(1/\delta)\ln(1/p)}}{\epsilon n}\right)
\end{aligned}$$

Similarly, and based on $d \longrightarrow n$

$$\begin{aligned}
|\widetilde{E[X]}^2 - \widehat{E[X]}^2| &= O\left(2\left(\frac{\sum x}{n}\right)\left(\frac{\sum_{i\in J} e_{i,1}}{n}\right) + \left(\frac{\sum_{i\in J} e_{i,1}}{n}\right)^2\right) \\
&= O\left(\frac{B^3\sqrt{d\ln(1/\delta)\ln(1/p)}}{\epsilon n} + \frac{B^4 d\ln(1/\delta)\ln(1/p)}{\epsilon^2 n^2}\right) \\
&= O\left(\frac{B^4\sqrt{d}\ln(1/\delta)\ln(1/p)}{\epsilon^2 n}\right)
\end{aligned}$$

By triangle inequality, we have

$$\begin{aligned}
|\widetilde{\sigma_{\mathrm{x}}^2} - \widehat{\sigma_{\mathrm{x}}^2}| &\le |\widetilde{E[X^2]} - \widehat{E[X^2]}| + |\widetilde{E[X]}^2 - \widehat{E[X]}^2| \\
&= O\left(\frac{B^4\sqrt{d}\ln(1/\delta)\ln(1/p)}{\epsilon^2 n}\right)
\end{aligned}$$

and

$$\tau_1 = O\left(\frac{B^4\sqrt{d}\ln(1/\delta)\ln(1/p)}{\epsilon^2 n\widehat{\sigma_{\mathrm{x}}^2}}\right)$$

Similarly, based on the unbiased estimation, we have

$$\widehat{E[XY]} = \frac{1-n}{1-d}\frac{\sum_{i\in J}\left(\sum_{t\in R_1.i} x\right)\left(\sum_{t\in R_2.i} y\right)}{n^2/d} + \frac{n-d}{1-d}\cdot\frac{\sum_{i\in J}\left(\sum_{t\in R_1.i} x\right)}{n^2/d}\cdot\frac{\sum_{i\in J}\left(\sum_{t\in R_2.i} y\right)}{n^2/d}$$

$$\begin{aligned}\widetilde{E[XY]} &= \frac{1-n}{1-d}\frac{\sum_{i\in J}\left(\left(\sum_{t\in R_1.i} x\right)+e_{i,2}\right)\left(\left(\sum_{t\in R_2.i} y\right)+e_{i,3}\right)}{\sum_{j\in J}(n/d+o(n/d))(n/d+o(n/d))} \\ &+ \frac{n-d}{1-d}\cdot\frac{\sum_{i\in J}\left(\left(\sum_{t\in R_1.i} x\right)+e_{i,2}\right)\sum_{i\in J}\left(\left(\sum_{t\in R_2.i} y\right)+e_{i,3}\right)}{\left(\sum_{j\in J}(n/d+o(n/d))(n/d+o(n/d))\right)^2} \\ &= \frac{1-n}{1-d}\cdot\frac{d}{n^2}\frac{\sum_{i\in J}\left(\left(\sum_{t\in R_1.i} x\right)+e_{i,2}\right)\left(\left(\sum_{t\in R_2.i} y\right)+e_{i,3}\right)}{1+o(1)} \\ &+ \frac{n-d}{1-d}\cdot\frac{d^2}{n^4}\frac{\sum_{i\in J}\left(\left(\sum_{t\in R_1.i} x\right)+e_{i,2}\right)\sum_{i\in J}\left(\left(\sum_{t\in R_2.i} y\right)+e_{i,3}\right)}{(1+o(1))^2}\end{aligned}$$

Based on the same flow of logic as $|\widetilde{E[X^2]}-\widehat{E[X^2]}|$, we would like to bound $\sum_{i\in J}\left(\left(\sum_{t\in R_1.i} x\right)e_{i,3}+\left(\sum_{t\in R_2.i} y\right)e_{i,2}+e_{i,2}e_{i,3}\right)$. Note that

$$\sum_{i\in J}\left(\sum_{t\in R_2.i} y\right)e_{i,2} \le \frac{nB}{d}\sum_{i\in J} e_{i,2} = O\left(\frac{nB^3\sqrt{\ln(1/\delta)\ln(1/p)}}{\sqrt{d}\epsilon}\right)$$

$$\sum_{i\in J} e_{i,2}e_{i,3} \le \sigma\sqrt{\ln(8/p)}\sum_{i\in J} e_{i,2} = O\left(\frac{B^4\sqrt{d}\ln(1/p)\ln(1/\delta)}{\epsilon^2}\right)$$

Hence the first two terms are bounded by $O\left(\frac{B^4\ln(1/\delta)\ln(1/p)}{\sqrt{d}\epsilon^2}\right)$. For the last term, we may also bound as

$$\left(\sum_{i\in J}\sum_{t\in R_1.i} x\right)\sum_{i\in J} e_{i,3} = O\left(\frac{nB^3\sqrt{d\ln(1/\delta)\ln(1/p)}}{\epsilon}\right)$$

$$\sum_{i\in J} e_{i,3}\sum_{i\in J} e_{i,2} = O\left(\frac{B^4 d\ln(1/\delta)\ln(1/p)}{\epsilon^2}\right)$$

So the last term is $O\left(\frac{nB^3 d\sqrt{d\ln(1/\delta)\ln(1/p)}}{\epsilon n^3}+\frac{B^4 d^2\ln(1/\delta)\ln(1/p)}{\epsilon^2 n^3}\right)$, which can be combined as $O\left(\frac{B^4 d^2\ln(1/\delta)\ln(1/p)}{\epsilon^2 n^3}\right)$. Therefore

$$|\widehat{E[XY]}-\widetilde{E[XY]}| = O\left(\frac{B^4\ln(1/\delta)\ln(1/p)}{\sqrt{d}\epsilon^2}\right)$$

Based on the similar analysis as $|\widetilde{E[X]}^2-\widehat{E[X]}^2|$, we have $|\widetilde{E[X]}\widetilde{E[Y]}-\widehat{E[X]}\widehat{E[Y]}| = O\left(\frac{B^4\sqrt{d}\ln(1/\delta)\ln(1/p)}{\epsilon^2 n}\right)$. This yields

$$|\widehat{\sigma^2_{\text{xy}}}-\widetilde{\sigma^2_{\text{xy}}}| = O\left(\frac{B^4\ln(1/\delta)\ln(1/p)}{\sqrt{d}\epsilon^2}\right)$$

With an extra assumption that X and Y are 0-centered and each tuple within R_1 and R_2 is independent and the join key is uncorrelated with X and Y. By the Chernoff-Hoeffding's inequality, with probability at least $1-p/4$, we have

$$\left|\sum_{t\in R_1.i} x\right|, \left|\sum_{t\in R_1.i} y\right| \le B\sqrt{2\ln(16d/p)n/d} \quad \forall i\in J$$

This yields

$$\begin{aligned}\sum_{i\in J}\left(\sum_{t\in R_2.i} y\right)e_{i,2} &\le B\sqrt{2\ln(16d/p)n/d}\sum_{i\in J} e_{i,2} \\ &= O\left(\frac{B^3\sqrt{n\ln(d/p)\ln(1/\delta)\ln(1/p)}}{\epsilon}\right)\end{aligned}$$

Giving a bound that scale with the size of the relation

$$|\widehat{E[XY]}-\widetilde{E[XY]}| = O\left(\frac{B^4\ln(1/p)\ln(1/\delta)\sqrt{d\ln(d/p)}}{\epsilon^2\sqrt{n}}\right)$$

Putting everything together, with probability at least $1-p$, we have

$$\tau_2 = O\left(\frac{B^4\ln(1/p)\ln(1/\delta)\sqrt{d\ln(d/p)}}{\epsilon^2\sqrt{n\widehat{\sigma^2_{\text{x}}}}}\right)$$

Extension to multi-features. The extension of our analysis to multi-dimensional features involves two modifications. Firstly, the bounds B_1 and B_2 are determined by matrix norm bounds through random matrix theory [55] instead of the absolute value of single random variable . Secondly, the bound of the inverse of $\widetilde{\sigma^2_{\text{x}}}$ is required, where $\widetilde{\sigma^2_{\text{x}}}$ was scalar but now is a matrix; the inverse of $\widetilde{\sigma^2_{\text{x}}}$ may become unboundedly large if its minimum eigenvalue is close to 0. To address this, Wang [57] makes an additional assumption that the noises to $\widehat{\sigma^2_{\text{x}}}$ has a minimum eigenvalue λ_{min} of $o(|\widehat{\sigma^2_{\text{x}}}|)$.

E ALLOCATION OF NOISES

We analyze the implication of dynamic allocation of privacy budget for moments on linear regression confidence bound appendix D. For union, it is possible to impute noise directly to $(\widehat{\sigma^2_{\text{x}}})_i$, $(\widehat{\sigma^2_{\text{xy}}})_i$, empirical variance and covariance for each dataset R_i. Each of $(\widehat{\sigma^2_{\text{x}}})_i$, $(\widehat{\sigma^2_{\text{xy}}})_i$ has sensitivity $\Delta' = O(B^2/n)$. Thus, let $\sigma' = \sqrt{2\ln(1.25/\delta)}\Delta'/\epsilon$ and

$$\widetilde{\sigma^2_{\text{x}}} \sim \frac{\sum_i^k\left((\widehat{\sigma^2_{\text{x}}})_i+\mathcal{N}(0,\sigma'^2)\right)}{k}$$

Applying gaussian tail bound and the independency assumption yields $|\widetilde{\sigma^2_{\text{x}}}-\widehat{\sigma^2_{\text{x}}}| = O(B^2\sqrt{\ln(1/\delta)\ln(1/p)}/\epsilon\sqrt{k}n)$, and $|\widetilde{\sigma^2_{\text{xy}}}-\widehat{\sigma^2_{\text{xy}}}| = O(B^2\sqrt{\ln(1/\delta)\ln(1/p)}/\epsilon\sqrt{k}n)$. This reduces the bound on τ_1 and τ_2 by a factor of $O(B^2\sqrt{\ln(1/\delta)\ln(1/p)}/\epsilon)$. Based on appendix A, consider the query $q_{j,i}: \mathcal{D}^n \longrightarrow s_i$ returns the a vector s_i containing the sum of the i-order polynomial semiring across each join key.

$$\Delta_{q_{j,i}} = 1_{2\mathbb{Z}+1}(i)4B^{2i} + 1_{2\mathbb{Z}}(i)2B^{2i}$$

For linear regression, it is feasible to decomposite q_j into 3 sequential queries, $q_{j,0}$, $q_{j,1}$ and $q_{j,2}$, each with privacy budget $(\epsilon/3, \delta_3)$. Inheriting notations from appendix D, $\Delta = \Delta_{q_j}$, $O(B^2\Delta_{q_{j,0}}) = O(B\Delta_{q_{j,1}}) = O(\Delta_{q_{j,2}}) = O(\Delta)$, note that although there is less privacy budget on releasing the count of tuples within each join key, the sensitivity is also reduced by a magnitude of B^2, i.e. from Δ to $\Delta_{q_{j0}}$. Hence, we believe it is still reasonable to assume that the noise on this number is small, and bounded by $o(n/d)$. The main implication is that $e_{i,2}, e_{i,3} = \mathcal{N}(0, \sqrt{2\ln(1.25/(\delta/3))}\Delta_{q_{j,1}}/(\epsilon/3))$, and $e_{i,1} = \mathcal{N}(0, \sqrt{2\ln(1.25/(\delta/3))}\Delta_{q_{j,2}}/(\epsilon/3))$, and no more change to the analysis is required. Following the computations in appendix D, we have

$$\tau_1 = O(\frac{B^2\sqrt{d}\ln(1/\delta)\ln(1/p)}{\epsilon^2 n\widehat{\sigma_{\mathrm{x}}^2}}), \tau_2 = O(\frac{B^2\ln(1/\delta)\ln(1/p)}{\sqrt{d}\widehat{\sigma_x^2}})$$