The Fast and the Private: Task-based Dataset Search

Zezhou Huang
zh2408@columbia.edu
Columbia University

Jiaxiang Liu
jl6235@columbia.edu
Columbia University

Haonan Wang
hw2983@columbia.edu
Columbia University

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

ABSTRACT

Recent platforms utilize ML task performance metrics, not metadata keywords, to search large data corpus. Requesters provide an initial dataset, and the platform searches for additional datasets that augment—join or union—requester's dataset to most improve the model (e.g., linear regression) performance. Although effective, current task-based data searches are stymied by (1) high latency which deters users, (2) privacy concerns for regulatory standards, and (3) low data quality which provides low utility. We introduce `Mileena`, a fast, private, and high-quality task-based dataset search platform. At its heart, `Mileena` is built on pre-computed semi-ring sketches for efficient ML training and evaluation. Based on semi-ring, we develop a novel *Factorized Privacy Mechanism* that makes the search differentially private and scales to arbitrary corpus sizes and numbers of requests without major quality degradation. We also demonstrate the early promise in using LLM-based agents for automatic data transformation and applying semi-rings to support causal discovery and treatment effect estimation.

1 INTRODUCTION

Existing relational data repositories [37, 40] offer the potential to augment and improve data-oriented tasks such as machine learning, and have motivated considerable work in both research [10, 17, 18] and industry [2, 3, 29]. The traditional approach uses keyword search [5] over metadata about datasets but requires the user to guess relevant keywords, manually integrate each returned dataset, and assess its utility. In response, recent work advocates for task-based data search [10, 24, 32, 39, 46], which takes as input an ML task (based on training and test datasets) and returns datasets in the data store that augment the training dataset in a way to improve the model quality. These augmentations can be any combination of joins and unions with other relations in the data store to add features, and with relations to add more samples.

In principle, such a system could enable a development cycle: as data users develop predictive or causal models over their local data, it searches for and automatically suggests or even integrates datasets that *concretely improves the user's model*. Existing approaches develop a data discovery index [17] to identify union and join candidate augmentations; but they laboriously evaluate each candidate by applying the augmentation, retraining, and cross-validating the model to assess *utility*. Other works cluster and prune the candidates [18], but a single search query still takes minutes.

We believe practical task-based data search remains stymied by three practical considerations:

- *Latency*: Machine learning is often performed as part of an iterative user-facing data analysis and development process [42]. As in web search, latency affects a user's willingness to use the system [7]. Unfortunately, existing data search systems take tens of minutes to hours because candidate assessment relies on costly model retraining and evaluation [10, 18]. These latencies pale in comparison to existing keyword-based dataset search services which, despite their disconnect from the user's data task, return results in seconds and dominate current deployments [3, 5].
- *Privacy*: Even within a single organization, data privacy, and access controls are major concerns. For instance, access and use of data containing Personally Identifiable Information (PII) are regulated by governments [1]. Differential privacy is a promising approach that is rapidly gaining adoption due to its well-defined mathematical guarantees [13]. It introduces noise to released statistics in a way that masks the presence or absence of an individual in the dataset, and was famously used to release fine-grained statistics for the 2020 US Census [30]. Unfortunately, existing applications of differential privacy have largely focused on federated machine learning use cases to train a single model. When applied to dataset search over even a handful of datasets, it requires so much noise that search is no better than random.
- *Data Quality*: It is now a common refrain that 80-90% of effort in data science goes into data preparation and cleaning, and this is similarly true for machine learning applications [42]. However, this challenge is exacerbated in a dataset search context, where data providers may upload hundreds or thousands of datasets and cannot be expected to prepare and clean each one. It's similarly unrealistic to expect the end-user to do the same for each candidate augmentation. Yet, preparation and cleaning may be necessary for search queries to return high-quality results that most improve the user's ML models. To this end, a scalable, extensible, and *fully-automated cleaning procedure* is necessary.

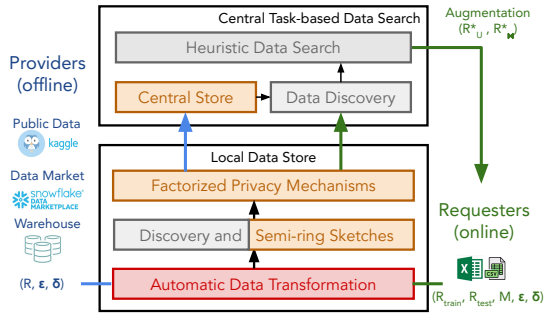

Figure 1: `Mileena` Architecture. Gray components are from previous works, orange represents current progress, and red indicates ongoing work. The blue workflow is offline for providers, while the green workflow is online for requesters.

This paper describes our current progress to develop a fast, private, and high-quality task-based dataset search platform called `Mileena`. At its heart, the system is built on the concept of factorized ML using semi-ring aggregation. Similar to data cubes which rely on aggregations that distribute across unions to pre-compute partial aggregates over partitions of a relation, semi-ring aggregates

distribute across unions *and* joins. Semi-rings have been designed for common statistical aggregation functions, as well as a wide range of machine learning models, including linear regression. This is a natural match with dataset search, where the goal is to join and union an initial training dataset with registered relations, and evaluate a data task over the result. For data tasks that can be formulated over semi-rings, such as training a linear model, the semi-ring computation can be pushed to the base relations and pre-computed.

This insight helps `Mileena` scale to thousands of datasets and return high-quality augmentations within a few seconds. Data providers pre-compute and upload semi-ring aggregates of each dataset to `Mileena`. When a user submits a search request, `Mileena` uses a semi-ring-based proxy model to find the most promising augmentations, and then trains a final model that is returned to the user. We also show that this approach is compatible with differential privacy, and develop a novel *Factorized Privacy Mechanism* that makes the entire search process differentially private, while scaling to arbitrary corpus sizes and numbers of requests without major degradation in the search result quality.

We also present our ongoing work that improves data quality and extends the semi-ring framework to causal inference. For data quality, we propose an agent-based framework to automatically transform and extract features from a provider's dataset before registration with `Mileena`. The key idea is to use agents that perform a range of exploratory data analysis and context-gathering tasks to distill the semantics of the dataset into a compact representation, which is then used to generate transformation and featurization functions custom to the dataset. To support causal inference, we are studying how semi-rings can be used for both causal discovery and estimating average treatment effects, and novel challenges that arise in the context of dataset search. By building on semi-rings, causal inference tasks are automatically made differentially private. Section 2 and 3 mainly summarizes our work in [21] and [24].

2 PROBLEM AND SOLUTION

In this section, we define the problem of differentially private task-based dataset searches. We start with the background of the data model, ML task, and differential privacy. Then, we lay out the trust model, privacy needs, and search problem. Finally, we walk through the architecture of `Mileena` to solve this problem. While non-private search has been studied [10, 32, 39, 46], we are the first to establish the trust and privacy requirements for practical dataset search.

2.1 Problem Definition

Data Model. We follow the standard relational data model. Relations are denoted as R, attributes as A, and domains as $dom(A)$. For clarity, the schema is included in square brackets $R[A_1, \cdots, A_n]$.

Data Task. We focus on ML task (M, R_{train}, R_{test}), which seeks to train a good model on a relation R containing features $X \subset S_R$ and target $Y \in S_R$. The function $M.Train(R_{train})$ returns a model m that predicts y from X. The function $M.Evaluate(m, R_{test})$ returns the *task utility* (typically cross-validation accuracy) on a test dataset. The goal of dataset search is to augment R_{train} with additional features (via joins) and samples (via unions) so that the *utility* over the augmented dataset is maximized. Beyond ML tasks, Section 4.2 describes our ongoing work to support causal inference (CI) tasks.

Trust Model. The data search platform may contain data that includes Personally Identifiable Information (PII). To comply with legal standards [1], organizations must protect data against misuse by untrusted entities. To model trust, on the one extreme, the local trust model used by Apple [11] and Google [15] assumes that individuals don't trust any aggregator. While this requires weak assumption, its mechanisms provide limited utility [52]. At the other extreme, others assume a global trust model [27], where individuals trust the central aggregator (central search). Mechanisms for this model yield high utility but the assumption is unrealistic. For dataset search, we introduce a two-level trust model (Figure 2) inspired by private federated ML [49]. Here, individuals trust direct 1^{st}-level aggregators (e.g., patients trust their healthcare providers) but not the 2^{nd}-level aggregator (data search platform) and other non-direct 1^{st}-level aggregators (e.g., other healthcare providers). Under this trust model, existing techniques in Trusted Execution Environments (TEEs) such as SGX [14, 41] are not applicable. This is because this method still requires a trusted 2^{nd}-level aggregator to store individuals' datasets.

Differential Privacy. For untrusted entities, rather than prohibit access outright, differential privacy (DP) [13] supports analysis of sensitive data while bounding the degree of privacy loss based on the budget (ϵ, δ) set by each aggregator. Each query on the dataset adds noise to the results, inversely proportional to the budget consumed; when $\epsilon = 0$, the dataset becomes inaccessible. Formally:

Definition 2.1 ((ϵ, δ)-DP). Let f be a randomized algorithm that takes a relation R as input. f is (ϵ, δ)-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, $Pr[f(R_1) \in S] \leq e^{\epsilon} Pr[f(R_2) \in S] + \delta$, 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.

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 central search platform for public benefit (providers) or improve ML accuracy for better (requesters). Following HIPPA guidance, they deidentify collected datasets through DP; these de-identified datasets are no longer considered PHI and can be disclosed.*

Differentially Private Task-based dataset search. Given a data corpus with datasets from different providers, a requester sends a request with datasets to augment a task (e.g., ML). The goal is to identify a set of augmentable (join/union) datasets that maximize task utility while satisfying trust and privacy requirements.

To formalize this, let $\mathcal{R} = \{R_1, R_2, ...\}$ be a data corpus collected from some providers. Providers and requesters hope to disclose datasets to the untrusted central search. Each sets a `DP` budget (ϵ_i, δ_i) for each dataset R_i, which is independent of other datasets and the central search. Requester sends a request with training and testing dataset (R_{train}, R_{test}), chooses a model M, and specifies (ϵ, δ). Requester's goal is to train model M on R_{train} and maximize its performance on R_{test}, which we call the task's *utility*.

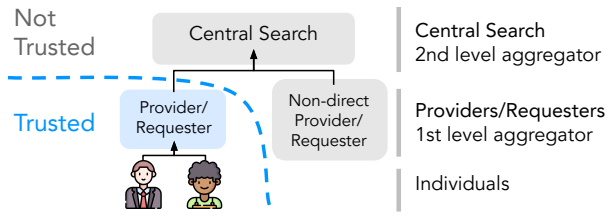

Figure 2: Mileena trust model: individuals only trust the direct 1^{st}-level aggregator, and not any others.

To maximize the *utility*, the requester aims to find a set of provider datasets in $\mathcal{R}$ to augment data. The function $Discover(R, augType)$ finds datasets in $\mathcal{R}$ that can be joined or unioned with R, given $augType \in \{\bowtie, \cup\}$. Putting everything together:

Problem 1 (Task-Based dataset search.). *For each request* $(R_{train}, R_{test}, M, \epsilon, \delta)$*, find the set of datasets* $\mathbf{R}^*_\cup, \mathbf{R}^*_\bowtie \subseteq \mathcal{R}$ *such that*

$$\mathbf{R}^*_\cup, \mathbf{R}^*_\bowtie = \underset{\mathbf{R}_\cup, \mathbf{R}_\bowtie}{\text{argmax}}\, M.Evaluate(M.Train(R_{trainAug}), R_{testAug})$$

$$s.t. \quad \mathbf{R}_\cup \subseteq Discover(R, \cup), \mathbf{R}_\bowtie \subseteq Discover(R, \bowtie),$$

$$R_{trainAug} = (R_{train} \cup_{R_1 \in \mathbf{R}_\cup} R_1) \bowtie_{R_2 \in \mathbf{R}_\bowtie} R_2$$

$$R_{testAug} = R_{test} \bowtie_{R \in \mathbf{R}_\bowtie} R$$

$$\text{The search over } (R_{train}, R_{test}) \text{ is } (\epsilon, \delta) - DP$$

$$\text{The search over } R_i \text{ is } (\epsilon_i, \delta_i) - DP, \forall R_i \in \mathcal{R}$$

2.2 Mileena Walkthrough

In this section, we walk through Mileena's architecture (Figure 1) to solve Problem 1. Mileena stands on prior data discovery and search systems [9, 10, 17]. However, these systems rely on slow join/union operations and fail to meet privacy requirements. To improve upon this, we use (1) a proxy model that can quickly estimate the benefits of a candidate augmentation using pre-computed semi-ring sketches (Section 3.2), (2) a factorized privacy mechanism to ensure DP (Section 3.3), and (3) an agent-based automated data transformation framework to improve data quality (Section 4.1).

2.2.1 Local Data Store. This locally manages each provider's/requester's raw data. It transforms and pre-processes each dataset, and generates privatized sketches that will be uploaded to the centralized search platform. Providers specify a DP budget for each relation they register; requesters upload training (and possibly testing) datasets with their DP budgets, and a task M.

Automatic Data Transformation. Raw datasets are noisy and require parsing and transformations to derive predictive features. We propose a fully automated approach based on LLM agents (Section 4.1), so the transformed dataset is most useful to search tasks. These costs can scale to the provider and requester's willingness to bear them. An enterprise may allocate resources to datasets in a data lake to improve searchability, while data sellers already clean and prepare data that they provide in existing data markets [26].

Discovery and Semi-ring Sketches. Data discovery indexes datasets based on their schemas and column features to quickly find join and union candidates (that need to be further assessed against the data task). We currently Aurum [17] (Section 2.2.2). For utility assessment, we propose novel semi-ring sketches, which are used for efficient ML training and evaluation[1].

Factorized Privacy Mechanism. Semi-ring sketches are aggregates over raw data, and can be privatized by adding appropriate noise. Compared to standard privacy mechanisms for data discovery index [16, 51], we introduce a novel Factorized Privacy Mechanism for semi-ring sketches (Section 3.3). Once privatized, these sketches can be repeatedly used across searches without any privacy cost.

2.2.2 Central Task-based dataset search. For each request, the central task-based dataset search component solves Problem 1 online.

Central data store. By default, all provider and requester data is privatized before upload to the central data store, and all search algorithms are over these privatized sketches.

Data Discovery. For each request, we employ Aurum [17] to discover augmentable data; Aurum finds union- and join-compatible datasets based on the column Jaccard similarity (minhash sketches) and cosine similarity (TF-IDF sketches).

Search algorithm. Given the candidate augmentations, the search algorithm greedily finds a sequence of vertical and horizontal augmentations that maximizes expected task utility. The basic algorithm (Algorithm 1) iterates over each augmentation, materializes the augmented dataset, trains the model, evaluates its training accuracy (or other quality measure), selects the augmentation that most improves the utility, and repeats. Augmentation, training, and evaluation (L5) are so expensive that existing works [9, 10, 18] primarily focus on aggressively pruning the set of augmentation to evaluate. In addition to these pruning methods, we use a semi-ring-compatible proxy model (e.g., linear regression) to directly derive the augmented model parameters and compute the model's utility in time independent of the relation sizes. This allows us to evaluate candidates in milliseconds. When used to power an AutoML service, Mileena improves final model accuracy, and reduces both query latency and monetary cost by orders of magnitude as compared to existing dataset search platforms and AutoML services (Section 3.2.3).

Algorithm 1 Search Algorithm

```
1:  Input: R, ML model
2:  R_U, R_⋈ ← {}, {}
3:  for k times do
4:      for all R_aug ∈ R_corpus do
5:          Augment R with R_aug, train and evaluate ML
6:      end for
7:      Greedily add the next best R_aug to R_U, R_⋈
8:      R ← Augment R with R_aug
9:  end for
10: Return: (R_U, R_⋈)
```

System output. Upon the completion of Mileena's search operation, the requester will be presented with an augmentation plan (query plan that first unions with $\mathbf{R}_\cup$ then joins with $\mathbf{R}_\bowtie$) as shown

[1]Semi-aggregates can express complex models like linear regression [47], gradient boosting and random forests [23], and approximate generalized linear models [25].

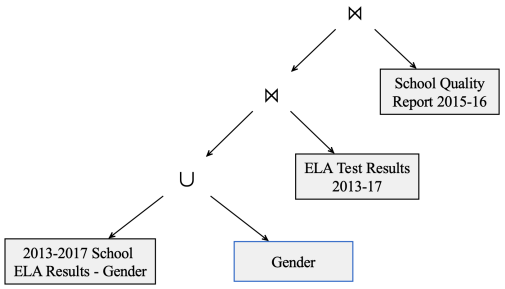

Figure 3: Augmentation plan generated by `Mileena`.

in fig. 3, detailing the discovered tables. Specifically, for datasets classified as public, a download link will be available. In cases involving sensitive data, `Mileena` will leverage established methodologies such as [33] to facilitate secure data sharing in data markets.

3 PRIVATE SEMI-RING SKETCHES

In this section, we delve into Semi-ring Sketches, exploring how they can efficiently train and evaluate the proxy model (linear regression), as well as the associated differential privacy mechanism.

3.1 Semi-ring Aggregates Primer

Augmentations are composed of joins and unions ($\mathbf{R}^*_\cup, \mathbf{R}^*_\bowtie$ in Problem 1) with relevant datasets. For efficient heuristic data search, it is necessary to reevaluate the data task (e.g., retraining and evaluating an ML model) after augmenting (joining or unioning) the training data in time independent of the relation sizes. Our main observation is that semi-ring aggregations are a natural fit for this use case.

Annotated Relations and Semi-ring Aggregates. The annotated relational [6] model maps $t \in R$ to a commutative semi-ring $(D, \oplus, \otimes, 0, 1)$, where D is a set, $\oplus$ and $\otimes$ are commutative binary operators closed over D, and $0/1$ are zero/unit elements. An annotation for $t \in R$ is denoted as $R(t)$, and $R(t) = 0$ for $t \notin R$.

Aggregation Query with Semi-ring. Some aggregation queries can be reformulated using annotations of relations.

- $(\gamma_\mathbf{A} R)(t) = \sum\{R(t_1) | \ t_1 \in R, \pi_\mathbf{A}(t) = \pi_\mathbf{A}(t_1)\}$.
- $(R_1 \cup R_2)(t) = \ R_1(t) \oplus R_2(t)$.
- $(R_1 \bowtie R_2)(t) = \ R_1(\pi_{S_{R_1}}(t)) \otimes R_2(\pi_{S_{R_2}}(t))$.

(1) The annotation for tuple t after a sum aggregation group-by $\mathbf{A}$ on R is the sum ($\oplus$) of the annotations across all join keys corresponding to t. (2) The annotation for union $R_1 \cup R_2$ is the sum of annotations in R_1 and R_2. (3) The annotation for join $R_1 \bowtie R_2$ is the product of contributing annotations.

Aggregation Pushdown. Semi-rings allow for aggregations' distribution (pushdown) through joins and unions [6], akin to projection pushdown before joins. Consider the query

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

Rather than apply γ after the join (which is $O(n^3)$ where n is relation size), γ can be performed on R_1 (and similarly on R_2, R_3) before joining with R_2, in $O(n)$:

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

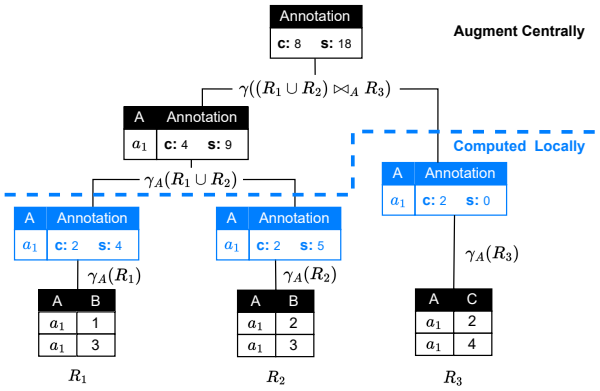

Figure 4: $\gamma((R_1 \cup R_2) \bowtie_A R_3)$ computes linear regression; aggregations are pushed before joins, and are pre-computed locally to accelerate central data search.

Associativity of additions can be exploited for union:

$$\gamma_A(R_1[A,B] \cup R_2[A,B]) = \gamma_A(R_1[A,B]) \cup \gamma_A(R_2[A,B])$$

Application to Linear Regression. Given the training data $\mathbf{X} \in \mathbb{R}^{n \times m}$, and the target variable $\mathbf{y} \in \mathbb{R}^{n \times 1}$, linear regression [47] computes $\theta^* = (\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T\mathbf{y}$. By considering $\mathbf{y}$ as a special feature, we find that $\mathbf{X}^T\mathbf{X} \in \mathbb{R}^{m \times m}$ is the core sufficient statistics to compute, which contains the count, sum of features and sum of products between features [47]. Thus, computing $\mathbf{X}^T\mathbf{X}$ over joins and unions boils down to an aggregation query over these operations, which is supported by semi-ring operations through annotations, as we have just discussed. We provide an example for sum:

Example 2. *Given R_1, R_2, R_3 in fig. 4, we aim to compute the sum of feature B over $(R_1 \cup R_2) \bowtie_A R_3$. Naively, we will first materialize the union and join results and then aggregate B. On the other hand, we design a simplified version of the covariance matrix semi-ring as a pair $(c, s) \in (\mathbb{Z}, \mathbb{R})$, which contains the count and sum, respectively. $\oplus$ and $\otimes$ are defined as $(c_1, s_1) \oplus (c_2, s_2) = (c_1 + c_2, s_1 + s_2)$ and $(c_1, s_1) \otimes (c_2, s_2) = (c_1 c_2, c_1 s_2 + c_2 s_1)$. Concretely, each tuple will be augmented with 2 additional columns: one is initialized with a value of 1 representing c, and the other with the value of B representing s (or 0 if B is absent). Subsequently, aggregation is prioritized before any union or join operations; we pre-aggregate annotations for each join key as highlighted in blue in fig. 4. Next, unioning and joining are translated into corresponding $\oplus$ and $\otimes$ operators over the semi-ring annotations. Finally, an aggregation is performed to compute the sum of B over $\mathbf{A}$, yielding $s = 18$.*

3.2 Pre-computed Semi-ring Sketches

Although aggregation pushdown optimizes training and evaluation for a single augmentation, the whole search process still requires the recomputation of semi-ring aggregates across all augmentations. To optimize this, we aggressively pre-compute aggregate as sketches locally. Online, the evaluation for horizontal augmentation reduces $O(n)$ to $O(1)$ and $O(d)$ for vertical augmentation, with n being the relation size and d the join key cardinality. Typically, $d << n$.

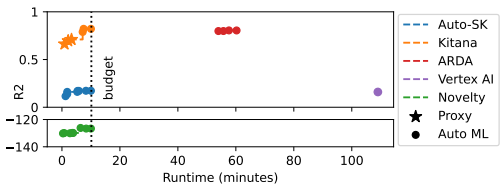

Figure 5: Task utility (testing $R2$) with 10 minutes time budget (dotted line). Mileena searches a corpus of 517 datasets with linear regression and reaches $R2 =\sim 0.7$; Mileena then sends the augmented dataset to AutoML and further improves $R2$ to 0.82. Other baselines are either slow or have low $R2$.

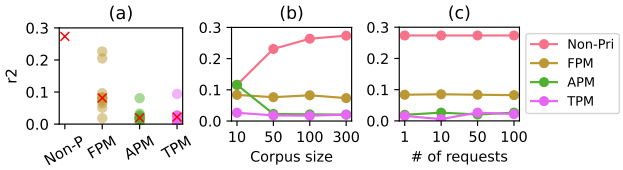

Figure 6: Task utility (non-private $r2$) for ML over augmented dataset from different private searches (a) across 10 runs with the median as a red cross, (b) varying the corpus size, and (c) varying the number of requests. FPMnotably outperforms even with a high volume of corpus datasets and requests.

3.2.1 Horizontal Augmentation. Horizontal augmentation A^h unions training data R. As discussed earlier, the aggregation can be pushed before unioning. The key optimization is to pre-compute aggregations for R and A^h as $\gamma(R)$ and $\gamma(A^h)$, when data is uploaded to the local data store. $\gamma(R)$ is shared across all candidate horizontal augmentations, and $\gamma(A^h)$ is shared across user requests. Now, horizontal augmentation adds pre-computed aggregates in near-constant time.

3.2.2 Vertical Augmentation. Vertical augmentation is more complex because pushing aggregation through the join needs to consider join keys. Consider A^v, which joins the training data R using join key j, and similarly, aggregations for R and A^v can be pre-computed as $\gamma_j(R)$ and $\gamma_j(A^v)$, $\gamma_j(R)$ is shared among all vertical augmentation candidates with join key j. Thus, we pre-compute $\gamma_{j'}(R)$ for all of its valid join keys j'. We also pre-compute $\gamma_{j'}(A^v)$ for all of its valid join keys, and share them across all requests where A^v is a vertical candidate.

3.2.3 Experiments. We use Mileena to power an AutoML service that uses up to 10 minutes for data search. Then, we materialize the augmented dataset and use the remaining time to run an AutoML library or service [24]. We compared it with existing ML-based dataset search systems (ARDA [10], Novelty [32]), AutoML libraries (Auto-sklearn), and Google's Vertex AI on 517 datasets from NYC Open Data [40]. Figure 5 reports $R2$ from Mileena's proxy model (stars) and the AutoML models (circle). Note that ARDA and Vertex AI don't enforce the time budgets. Pure AutoML approaches perform poorly because the dataset lacks predictive features; ARDA eventually finds a good model after ≈50min that is slightly worse than Mileena. Novelty assesses augmentations based on how "novel" the data is compared to the training data, but is uncorrelated with model utility and actually degrades the final model. Mileena returns a high-quality model almost immediately, and converges to the highest quality model within the budget.

3.3 Factorized Privacy Mechanism

Private task-based data search is particularly challenging: even a single request requires model retraining across a vast number of augmented datasets as join/union results. How can we ensure a scalable data search with large number of datasets and requests, without depleting the privacy budgets of both requesters and providers?

Our key insight is that the semi-ring sketches, once privatized, are composable (through semi-ring operators) and reusable (as post-processing without additional DP cost), making them ideal for data search that trains ML across joins and unions from different sets of relations. We name our mechanism *Factorized Privacy Mechanism* (FPM), which applies Gaussian mechanism [12] to these sketches (blue in Figure 4) locally before transferring to the central data corpus. We further develop novel budget allocations that optimize the proxy model's accuracy [21].

3.3.1 Experiments. Using NYC Open Data and regression models (utility is $R2$), Figure 6 shows how FPMscales far beyond existing mechanisms in terms of corpus size and number of search requests as compared to **APM** [50], which applies DP mechanism to aggregates after computing the join/union results under a global trust model, and **TPM** [53], applies DP mechanism to individual tuples. **Non-P** reports results without any privacy; although FPMachieves up to ~40–90% of the Non-P utility, the gap can be further reduced by clustering and smoothing join groups [28] in the future works.

4 ONGOING WORK

We now describe two directions of ongoing work.

4.1 Hand-Free Data Transformation

Data transformation is pivotal for ML [45], but Mileena uses private sketches rather than transformable raw data during the search. Is it possible to transform datasets locally prior to sketch computation in ways that benefit a variety of data tasks?

Recent data transformation approaches rely on deep-learning [20, 36], including Language Learning Models (LLMs) [38]. Although powerful and promising, LLMs need to serialize datasets into a textual form to include in the prompt context. In contrast, LLMs have limited context lengths (GPT-4 supports 8K input tokens) and are very costly (GPT-4 costs $\$0.03/1K$ tokens). Long contexts also lead to unreliably attention and hallucinations [34].

Our intuition is that developers and data scientists do not design features in one-shot from the raw data either. Instead, they perform exploratory data analysis (EDA), and understand the problem context—both to reduce the amount of information to keep in their human memory, identify the salient semantics relevant to the problem, and then use the knowledge to synthesize features that incorporate the problem semantics. To this end, we propose an agent-based framework for data transformation. Each agent specializes in a particular task and summarizes the information

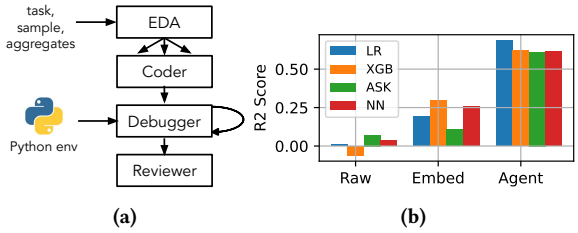

Figure 7: (a) Architecture for Agent-based Data Transformation. (b) Agent-based Transformation shows Model Performance (R2) across different ML models.

in a form consumable by an LLM or another agent. The design is illustrated in Figure 7a, with the following agents:

- **EDA**: This agent explores data and related docs to suggest transformations. Our implementation inputs the ML task contexts, a sample of ten rows, and column aggregates (min, max, median), and let this agent output a list of data transformations in NL.
- **Coder**: Each suggested transformation by EDA is designated to one **Coder**, which also inputs the related column samples and outputs a Python function to implement the transformation.
- **Debugger**: This agent inputs the function, accesses a Python environment, and ensures that the function can run. Following [43], Debugger iteratively modifies the function based on error messages. By default, the debugging is retried up to 10 times; if it still fails, that transformation is ignored.
- **Reviewer**: This agent evaluates the outputs from Debugger to ensure transformations meet EDA's requirements. It reviews the sample transformed data, and confirms if it aligns with the NL description by EDA to finalize the transformation.

4.1.1 Experiments. We evaluated linear regression, XGBoost, Auto-Sklearn, and TabNet [8] (SOTA DNN for tabular data) on the Kaggle Airbnb data [4], and report model R2. We compare no transformations, transformations using GPT-4 agents (us), and transformations using ada-002 embeddings (which create high-dimensional features for string columns). Figure 7b shows that agent-based transformation trumps model complexity and transformation approaches. The agents suggested diverse and useful transformations, from standard one-hot-encoding to more complex ones, such as string extraction and calculating stay duration from date strings. The most exciting result is that, **with agent-based transformations, linear regression (which is easy to maintain, and fast to train & predict as compared to NN models) out-performed all others**.

Our ongoing work moves toward a data store that continuously evolves and improves the transformations. This introduces novel challenges, such as how to efficiently update the sketches under DP [22], incorporate new information about datasets over time (e.g., crawl the web for data documentation), manage a library of diverse agents, and interact with the central data store.

4.2 Causal Inference

Causal Inference (CI) seeks to answer questions such as "What is smoking's impact on cancer?" It is distinguished from standard ML, which focuses on learning correlations (e.g., "How correlated is smoking to cancer?") because relationships are asymmetric, and the causal direction matters. CI queries rely on an accurate causal model, represented as a directed acyclic graph (DAG). Without key confounders [54], its intervention estimates can be arbitrarily incorrect. Dataset search offers the potential to *find* these missing variables and *discover* a sufficient causal model.

Factorized Causal Discovery. Existing causal discovery algorithms [19] only infer an equivalence class of DAGs given observational data alone. For example, conditional independence tests may determine that smoking and cancer are dependent, but cannot infer smoking causes cancer. Fortunately, the causal direction can be determined assuming sufficiency (no unobserved confounders), non-Gaussian noise, and linear relationships [48]. Consider, X and Y with $X \sim U(0, 10)$ and $Y = 2X + \epsilon \sim U(0, 10)$. A linear regressor using X to predict Y yields residuals $res_y \perp\!\!\!\perp X$ because $\epsilon \perp\!\!\!\perp X$, but the residuals when using Y to predict X yields $res_x \not\perp\!\!\!\perp Y$.

However, the assumptions are not realistic in practice. Fortunately, 1-N and N-N relationships between relations create colliders, on a lifted representation of variables, discoverable by conditional independence tests [35] that relax the linearity and non-Gaussian noise assumptions. Our ongoing work focuses on using semi-rings to integrate these ideas into `Mileena`'s fast and DP framework.

Differentially Private Treatment Effects. Given a causal diagram, existing techniques [44] that evaluate treatment effects require the distribution of treatment, target, and adjustment variables, potentially from different relations. This requires joining privatized relations, which may amplify DP noise so much that the resultant join distribution is ineffective. However, [31] shows that treatment effects are computable from marginal distributions.

We ran a synthetic experiment with 3 relations $R_1(T, Y), R_2(T, G), R_3(P, A, Y)$; with binary attributes student qualification (T), overall score (Y), gender (G), student participation (P), and assignment completion (A), DP budgets $\epsilon = 1$ and $\delta = 10^{-6}$, and 1-to-1 relationships. The causal diagram is $T \rightarrow P \rightarrow A \rightarrow Y$ and D is a confounder between T and Y ($T \leftarrow D \rightarrow Y$). We intervene on T and estimate its expected effect on Y, $E[Y \mid do(T = 1)]$ (ATE = $E[Y \mid do(T = 1)] - E[Y \mid do(T = 0)]$). We compare relative error using (1) backdoor adjustment by estimating $P(X, Y, G)$ from privatized R_1 and R_2, then $R_1 \bowtie R_2$. (2) $\sum_y y \sum_a P(a \mid t) \sum_p P(y \mid a, p)P(p)$ by estimating $P(A, T)$ from privatized R_1 and R_3, then $R_1 \bowtie R_3$ along with a noisy histogram of R_3. Their respective relative errors were 10.25% and 0.21%. Surprisingly, splitting the privacy budget between R_3 and its histogram greatly improves estimate accuracy. Our ongoing work designs intermediates from each dataset based on semi-ring aggregations to (1) represent the marginal distribution of the dataset, and (2) support computation of the joint distribution through joins.

Acknowledgements: This material is based upon work supported by the National Science Foundation under Grant No. 1845638, 1740305, 2008295, 2106197, 2103794, 2312991, Amazon, Google, Adobe, and CAIT.

REFERENCES

[1] 2018. California Consumer Privacy Act. https://oag.ca.gov/privacy/ccpa.
[2] 2022. Amazon Marketplace. https://aws.amazon.com/marketplace.
[3] 2022. Snowflake Marketplace. https://www.snowflake.com/data-marketplace/.
[4] 2023. Airbnb. https://www.kaggle.com/datasets/joyshil0599/airbnb-listing-data-for-data-science.

[5] 2023. Google Dataset Search. https://datasetsearch.research.google.com/.
[6] Mahmoud Abo Khamis, Hung Q Ngo, and Atri Rudra. 2016. Faq: Questions Asked Frequently. In *SIGMOD*.
[7] Ioannis Arapakis, Xiao Bai, and B Barla Cambazoglu. 2014. Impact Of Response Latency On User Behavior In Web Search. In *SIGIR*.
[8] Sercan Ö Arik and Tomas Pfister. 2021. Tabnet: Attentive Interpretable Tabular Learning. In *AIII*.
[9] Sonia Castelo, Rémi Rampin, Aécio Santos, Aline Bessa, Fernando Chirigati, and Juliana Freire. 2021. Auctus: a Dataset Search Engine for Data Discovery and Augmentation. In *PVLDB*.
[10] Nadiia Chepurko, Ryan Marcus, Emanuel Zgraggen, Raul Castro Fernandez, Tim Kraska, and David Karger. 2020. Arda: Automatic Relational Data Augmentation for Machine Learning. In *arXiv*.
[11] Graham Cormode, Somesh Jha, Tejas Kulkarni, Ninghui Li, Divesh Srivastava, and Tianhao Wang. 2018. Privacy At Scale: Local Differential Privacy In Practice. In *SIGMOD*.
[12] Cynthia Dwork, Krishnaram Kenthapadi, Frank McSherry, Ilya Mironov, and Moni Naor. 2006. Our Data, Ourselves: Privacy Via Distributed Noise Generation. In *EUROCRYPT*.
[13] Cynthia Dwork, Frank McSherry, Kobbi Nissim, and Adam Smith. 2006. Calibrating Noise to Sensitivity In Private Data Analysis. In *TCC*.
[14] Muhammad El-Hindi, Tobias Ziegler, Matthias Heinrich, Adrian Lutsch, Zheguang Zhao, and Carsten Binnig. 2022. Benchmarking the second generation of intel SGX hardware. In *Proceedings of the 18th International Workshop on Data Management on New Hardware*. 1–8.
[15] Úlfar Erlingsson, Vasyl Pihur, and Aleksandra Korolova. 2014. Rappor: Randomized Aggregatable Privacy-Preserving Ordinal Response. In *SIGSAC*.
[16] Natasha Fernandes, Yusuke Kawamoto, and Takao Murakami. 2021. Locality Sensitive Hashing With Extended Differential Privacy. In *ESORICS*.
[17] Raul Castro Fernandez, Ziawasch Abedjan, Famien Koko, Gina Yuan, Samuel Madden, and Michael Stonebraker. 2018. Aurum: A Data Discovery System. In *ICDE*.
[18] Sainyam Galhotra, Yue Gong, and Raul Castro Fernandez. 2023. Metam: Goal-Oriented Data Discovery. In *arXiv*.
[19] Clark Glymour, Kun Zhang, and Peter Spirtes. 2019. Review Of Causal Discovery Methods Based On Graphical Models. In *Frontiers in genetics*.
[20] Alireza Heidari, Joshua McGrath, Ihab F Ilyas, and Theodoros Rekatsinas. 2019. Holodetect: Few-Shot Learning for Error Detection. In *SIGMOD*.
[21] Zezhou Huang, Jiaxiang Liu, Daniel Alabi, Raul Castro Fernandez, and Eugene Wu. 2023. Saibot: A Differentially Private Data Search Platform. In *PVLDB*.
[22] Ziyue Huang, Yuan Qiu, Ke Yi, and Graham Cormode. 2021. Frequency Estimation Under Multiparty Differential Privacy: One-Shot and Streaming. In *arXiv*.
[23] Zezhou Huang, Rathijit Sen, Jiaxiang Liu, and Eugene Wu. 2023. Joinboost: Grow trees over normalized data using only SQL. In *PVLDB*.
[24] Zezhou Huang, Pranav Subramaniam, Raul Castro Fernandez, and Eugene Wu. 2023. Kitana: Efficient Data Augmentation Search for AutoML. In *arXiv*.
[25] Jonathan Huggins, Ryan P Adams, and Tamara Broderick. 2017. Pass-Glm: Polynomial Approximate Sufficient Statistics for Scalable Bayesian Glm Inference. In *NeurIPS*.
[26] Alex Izydorczyk. 2023. https://magis.substack.com/p/snowflake-summit-2023.
[27] Noah Johnson, Joseph P Near, and Dawn Song. 2018. Towards Practical Differential Privacy for Sql Queries. In *PVLDB*. VLDB Endowment.
[28] Georgios Kellaris and Stavros Papadopoulos. 2013. Practical differential privacy via grouping and smoothing. *Proceedings of the VLDB Endowment* (2013).
[29] Javen Kennedy, Pranav Subramaniam, Sainyam Galhotra, and Raul Castro Fernandez. 2022. Revisiting Online Data Markets In 2022: A Seller and Buyer Perspective. In *SIGMOD*.
[30] Christopher T Kenny, Shiro Kuriwaki, Cory McCartan, Evan TR Rosenman, Tyler Simko, and Kosuke Imai. 2021. The Use Of Differential Privacy for Census Data and Its Impact On Redistricting: The Case Of the 2020 Us Census. In *Science advances*. American Association for the Advancement of Science.
[31] Sanghack Lee and Elias Bareinboim. 2020. Causal Effect Identifiability Under Partial-Observability. In *ICML*.
[32] Yifan Li, Xiaohui Yu, and Nick Koudas. 2021. Data Acquisition for Improving Machine Learning Models. In *PVLDB*. VLDB Endowment.
[33] Jinfei Liu. 2020. Dealer: end-to-end data marketplace with model-based pricing. *arXiv preprint arXiv:2003.13103* (2020).
[34] Nelson F Liu, Kevin Lin, John Hewitt, Ashwin Paranjape, Michele Bevilacqua, Fabio Petroni, and Percy Liang. 2023. Lost In the Middle: How Language Models Use Long Contexts. In *arXiv*.
[35] Marc Maier, Katerina Marazopoulou, David Arbour, and David Jensen. 2013. A Sound and Complete Algorithm for Learning Causal Models From Relational Data. In *arXiv*.
[36] Yinan Mei, Shaoxu Song, Chenguang Fang, Haifeng Yang, Jingyun Fang, and Jiang Long. 2021. Capturing Semantics for Imputation With Pre-Trained Language Models. In *ICDE*.
[37] Natalia Miloslavskaya and Alexander Tolstoy. 2016. Big Data, Fast Data and Data Lake Concepts. In *Procedia Computer Science*. Elsevier.
[38] Avanika Narayan, Ines Chami, Laurel Orr, Simran Arora, and Christopher Ré. 2022. Can Foundation Models Wrangle Your Data?. In *arXiv*.
[39] Fatemeh Nargesian, Abolfazl Asudeh, and HV Jagadish. 2022. Responsible Data Integration: Next-Generation Challenges. In *SIGMOD*.
[40] nycopen 2022. Nyc Open Data. https://opendata.cityofnewyork.us/.
[41] Christian Priebe, Kapil Vaswani, and Manuel Costa. 2018. EnclaveDB: A secure database using SGX. In *2018 IEEE Symposium on Security and Privacy (SP)*. IEEE, 264–278.
[42] Fotis Psallidas, Yiwen Zhu, Bojan Karlas, Jordan Henkel, Matteo Interlandi, Subru Krishnan, Brian Kroth, Venkatesh Emani, Wentao Wu, Ce Zhang, et al. 2022. Data Science Through the Looking Glass: Analysis Of Millions Of Github Notebooks and Ml. Net Pipelines. In *SIGMOD Record*. ACM New York, NY, USA.
[43] Fardin Ahsan Sakib, Saadat Hasan Khan, and AHM Karim. 2023. Extending the Frontier Of Chatgpt: Code Generation and Debugging. In *arXiv*.
[44] Babak Salimi, Harsh Parikh, Moe Kayali, Lise Getoor, Sudeepa Roy, and Dan Suciu. 2020. Causal Relational Learning. In *SIGMOD*.
[45] 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 *CHI*.
[46] Aécio Santos, Aline Bessa, Christopher Musco, and Juliana Freire. 2022. A Sketch-Based Index for Correlated Dataset Search. In *ICDE*.
[47] Maximilian Schleich, Dan Olteanu, and Radu Ciucanu. 2016. Learning Linear Regression Models Over Factorized Joins. In *SIGMOD*.
[48] Shohei Shimizu, Takanori Inazumi, Yasuhiro Sogawa, Aapo Hyvarinen, Yoshinobu Kawahara, Takashi Washio, Patrik O Hoyer, Kenneth Bollen, and Patrik Hoyer. 2011. Directlingam: A Direct Method for Learning a Linear Non-Gaussian Structural Equation Model. In *JMLR*.
[49] Chang Wang, Jian Liang, Mingkai Huang, Bing Bai, Kun Bai, and Hao Li. 2020. Hybrid Differentially Private Federated Learning On Vertically Partitioned Data. In *arXiv*.
[50] Yu-Xiang Wang. 2018. Revisiting Differentially Private Linear Regression: Optimal and Adaptive Prediction & Estimation In Unbounded Domain. In *arXiv*.
[51] Benjamin Weggenmann and Florian Kerschbaum. 2018. Syntf: Synthetic and differentially private term frequency vectors for privacy-preserving text mining. In *SIGIR*.
[52] 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. In *IEEE TIFS*. IEEE.
[53] Mengmeng Yang, Lingjuan Lyu, Jun Zhao, Tianqing Zhu, and Kwok-Yan Lam. 2020. Local Differential Privacy and Its Applications: A Comprehensive Survey. In *arXiv*.
[54] Brit Youngmann, Michael Cafarella, Babak Salimi, and Anna Zeng. 2023. Causal Data Integration. In *arXiv*.