# METAM: Goal-Oriented Data Discovery

Sainyam Galhotra, Yue Gong, Raul Castro Fernandez
Department of Computer Science, The University of Chicago
Email: {sainyam, yuegong, raulcf}@uchicago.edu

***Abstract*—Data is a central component of machine learning and causal inference tasks. The availability of large amounts of data from sources such as open data repositories, data lakes and data marketplaces creates an opportunity to augment data and boost those tasks' performance. However, augmentation techniques rely on a user manually discovering and shortlisting useful candidate augmentations. Existing solutions do not leverage the synergy between discovery and augmentation, thus underexploiting data.**

**In this paper, we introduce METAM, a novel goal-oriented framework that queries the downstream task with a candidate dataset, forming a feedback loop that automatically steers the discovery and augmentation process. To select candidates efficiently, METAM leverages properties of the: i) data, ii) utility function, and iii) solution set size. We show METAM's theoretical guarantees and demonstrate those empirically on a broad set of tasks. All in all, we demonstrate the promise of goal-oriented data discovery to modern data science applications.**

## I. INTRODUCTION

Augmenting a dataset by joining it with others can improve the utility of data-driven tasks such as causal inference and supervised machine learning. The abundance of tables in open repositories [1], lakes in organizations [2], and even data markets [3], [4] translates into an abundance of *augmentation candidates*. Identifying good augmentation candidates among many is a data discovery problem. To solve this problem, one could use a traditional data discovery system to identify what tables join with the input data, and then, separately, identify what joins increase the task's utility. This *discover-then-augment* approach works when the discovery system returns candidates relevant to the task. Unfortunately, in practice, it is hard to guarantee the discovery system identifies good augmentations because: i) relevant augmentations depend on the task; and ii) analysts may not know what properties make an augmentation relevant, e.g., what features augment the predictive power of a classifier. The disconnection between data discovery and augmentation presents a research opportunity.

We harness that opportunity with a new approach we call **goal-oriented data discovery**, where data discovery is not treated as a one-time process. Instead, it adapts to the task by performing interventions: a process to augment the initial dataset and validate its utility. *Interventional* queries help to identify augmentations that *cause* an increase in the task's utility, thus steering the discovery process to automatically maximize the task's utility. This achieves two objectives: first, analysts do not need to know what criteria make an augmentation good because the approach is automatic. Second, any downstream task with a utility function benefits from this discovery approach. The consequences of goal-oriented data discovery are significant; consider the following anecdote. We used goal-oriented data discovery to predict "housing prices" in a geographical area. Our technique identified some obvious datasets that a social scientist would have been able to identify using a discovery system, such as "income of people staying in the neighborhood" and "crime stats". But crucially, it also identified non-obvious datasets correlated with housing prices such as "presence of grocery stores" and "number of taxi trips" from those areas. Indeed, many sociologists and economists leverage external data to infer causal relationships between attributes of interest [5], [6], [7]. But they rely on domain knowledge and manual effort to identify those relationships. *Goal-oriented data discovery* paves the way to identify new causal relationships from large data repositories automatically.

A trivial but computationally prohibitive way of solving goal-oriented data discovery is to measure the utility of every augmentation candidate and choose the best. The key technical contributions of our paper focus on developing *interventional querying algorithms* for goal-oriented data discovery that exploit the structure of the data, the utility function, and the solution to adaptively prioritize the candidates for querying.

**Interventional Queries for Goal-Oriented Data Discovery.** Our technical contributions leverage the following properties to optimize the complexity of discovery without loss of quality.

Properties of the Data. A key insight is that *similar* augmentations perform similarly on the downstream task. We exploit this insight by clustering augmentations and judiciously choosing them from different clusters, thus skipping computation. To cluster augmentations, we represent each with a vector of *data profiles*, which are task-independent measures of data and include semantic similarity, correlation, and mutual information, among others. When a combination of data profiles is correlated with the task's utility, clustering narrows down the number of augmentation candidates.

Properties of the Utility function. A task's optimal utility is given by a set of augmentations. Enumerating all subsets is infeasible. Our insight is that when utility functions are monotonic, i.e., utility never decreases with new augmentations, it is possible to find augmentations efficiently, by considering them one by one. And we can make any function monotonic by ignoring augmentations that harm utility using a wrapper around the user-provided task implementation.

Properties of the Solution set. The optimal set of augmentations is a collection of join paths over different datasets. Most augmentations are irrelevant for a downstream task, and useful augmentations are a small subset of the candidate set. We leverage this property to prioritize small subsets of join

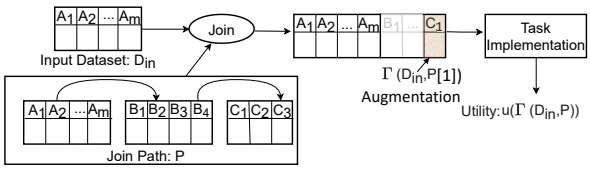


Fig. 1: Notation of a data augmentation pipeline.

paths over larger ones by using combinatorial testing.

METAM: Goal-oriented data discovery We combine the aforementioned properties into an anytime algorithm that finds augmentation candidates by adaptively querying the task. Although goal-oriented data discovery is NP-hard, the algorithm is guaranteed to identify an approximate solution under reasonable assumptions which hold in practice. We also show the efficiency guarantees of the algorithm, and implement it as part of METAM, an end-to-end data discovery approach.

**Evaluation Results.** METAM automatically finds useful augmentations among millions of them within minutes. We evaluate METAM on large repositories with data from cities in the United States and Kaggle, and for prescriptive analytics tasks such as *what-if* and *how-to* analysis in causal inference, classification and regression in supervised ML, as well as entity linking, and clustering.

**Outline.** We present notation and discuss the problem in Section II. We present insights for interventional queries in Section III, algorithm in Section IV, and it's analysis in Section V. We present evaluation in Section VI, and related work in Section VII.

## II. Goal-Oriented Data Discovery

In this section, we introduce notation and problem statement in Section II-A, and preliminaries in Section II-C.

### A. Problem Definition

Figure 1 presents a summary of the notation used in the paper. Let $\mathcal{R}(A_1, \ldots, A_m)$ denote a relation schema over $m$ attributes, where $A_i$ denotes the $i^{th}$ attribute. $D$ comprises a schema $\mathcal{R}(A_1, \ldots, A_m)$ and a list of tuples $T$ where each tuple $t \in T$ is a specific instance of the schema. Using this notation, we now define a noisy dataset and a data repository.

**Definition 1** (Noisy Structured Data)**.** *A noisy structured dataset $D$ is characterized by incomplete schema information $\mathcal{R}(A_1, \ldots, A_m)$ where $A_i = \phi$ denotes missing header values and a list of tuples $T$ which may contain less than $m$ values. Additionally, certain tuples may contain duplicate values.*

**Definition 2** (Data Repository)**.** *A data repository contains a set of datasets $\mathcal{D} = \{D_1, \ldots, D_n\}$, where different datasets $D_i, D_j$ may contain overlapping values. Additionally, the datasets may lack schema and contain missing values.*

The different datasets that are joined with the input dataset $D_{in}$ are denoted as a join path, defined below.

**Definition 3** (Join Path)**.** *A join path $P$ is defined as an ordered list of noisy datasets $P \equiv \{D_1, \ldots, D_t\}$ such that datasets $D_j$ and $D_{j+1}$ join for all $j < t$ with each other forming a chain of join operations. The dataset formed by joining these $t$ datasets is considered to be augmented with the original dataset $D_{in}$.*

The augmentation of a join path $P$ to a dataset $D_{in}$ in denoted by $\Gamma(D_{in}, P)$. Augmenting a join path $P$ with $D_{in}$ can augment multiple new columns, some of which may not be required by the downstream task. To distinguish between these different augmented columns, we define augmentation as the projection of the join consisting of a single column.

**Definition 4** (Augmentation)**.** *An augmentation $\Gamma(D_{in}, P[j])$ is defined as the $j^{th}$ column that is added after joining the path $P$ with the input dataset $D_{in}$.*

Note that $\Gamma(D_{in}, P) \equiv \{\Gamma(D_{in}, P[j]), \forall j \in \{1, 2, \ldots, l\}\}$, where $l$ is the number of columns after materializing the join-path $P$. $\Gamma$ considers materializing a join path to add a new attribute. Other types of data augmentation include unions (addition of additional rows), relational embedding based augmentations, spatio-temporal augmentations, etc. Empirically, we observed that join-path based addition of new attributes is commonly used for popular applications. However, our framework extends to any general set of augmentations. A task $t$ is a black box consisting of the analysis algorithm that takes a dataset ($D_{in}$ or an augmented version of $D_{in}$) as input and outputs the performance measure of the task. We call "query" the process of obtaining a task's utility on a dataset. We formally define the utility score of the task as follows.

**Definition 5** (Utility score)**.** *Given a task $t$ that operates on a dataset $D$, the utility score $u_t(D)$ is defined as the objective (or the evaluation metric) of the task when operated with $D$.*

Without loss of generality, we assume that the utility score is normalized $u_t(\cdot) \in [0, 1]$ and that higher utility score means better task quality. Different sets of augmentations may improve the task's utility. We seek a *minimal* set that contains only the augmentations that help to maintain high utility.

**Definition 6** (Minimal set of Augmentations)**.** *Given an initial dataset $D_{in}$, a mechanism to compute $u_t(\cdot)$, a set of augmentations $\mathcal{P} = \{P_1[i_1], \ldots, P_j[i_j]\}$ is considered minimal if removing any attribute from $\Gamma(D_{in}, \mathcal{P})$ reduces task utility.*

**Problem II.1** (Goal-oriented discovery)**.** *Given an initial dataset $D_{in}$, a task $t$ that has a mechanism to compute $u(D')$ for any table $D'$, the problem of identifying augmentations that optimize the task is to identify a minimum augmentation set $\mathcal{T}$ such that the augmented dataset has utility $u(\Gamma(D_{in}, \mathcal{T})) \geq \theta$.*

### B. Task implementation and utility calculation

A task takes a dataset as input and returns a utility score that depends, at least in part, on the input dataset. Some examples:

1) Predictive analytics: The task trains a machine learning model over an input training dataset. The training code could be simple, e.g., a random forest, or complex, such as an AutoML solution. The utility correspond to a model-relevant metric such as accuracy, F1 score and may also include fairness and robustness metrics.

2) Prescriptive analytics involves hypothetical analysis (what-if and how-to queries), causal inference, and explainability as the key components [8], [9], [10], [11]. The utility corresponds to a metric that summarizes the quality of generated explanations, support of identified causal dependencies, and total causal effect of attributes.
3) Data preparation and visualization: These include entity resolution, entity linking, data cleaning and visualization. The utility often corresponds to F-score and accuracy.

Note that these are some examples of task implementation and any advances in feature engineering, data transformation and data analysis can be incorporated in the utility function.

### C. Preliminaries and Problem Discussion

We use previous data discovery techniques [12], [13], [14], [15] to obtain a set of augmentations from large data repositories; we use Aurum [12]. Although existing solutions may generate noisy candidates (due to the use of approximate techniques and semantic ambiguity), our approach works even when different sources of noise are present. We also leverage existing profiling techniques [16], [17], [18], [19] to describe the augmentation candidates $\Gamma(D_{in}, \mathcal{P})$. For example, a data profile $\langle Corr, C_1, C_2 \rangle$ refers to the correlation between columns $C_1$ and $C_2$. We use the term *data profile value* to denote the value of the characteristic/property. A data profile may refer to a single attribute (e.g. domain of an attribute) or multiple attributes (correlation between attributes), and these properties may be correlated with downstream utility.

**Definition 7** (Data Profile)**.** *A data profile $X$ is defined as the property of a dataset $D$ such that the tuples in $D$ satisfy $X$.*

We consider the following data profiles in our implementation.

• **Correlation and Mutual Information (MI)**: MI is often used to evaluate causal relationship between attributes [20]. Correlation and MI are also predictors of the quality of attributes in machine learning tasks. This profile estimates the Pearson correlation (MI) of the candidate augmentation $P$ and the attributes of $D_{in}$.

• **Semantic-embedding based distance** profile captures the semantic similarity between the considered datasets. This profile is computed as the cosine similarity between the embeddings of both datasets constructed from pre-trained models such as BERT [21]. The dataset embedding is constructed by averaging the embedding vectors of tokens present in the table.

• **Dataset metadata/attributes** profile calculates the similarity between datasets based on their source and attributes. Unlike the semantic embeddings, this profile captures syntactic similarity between attributes and dataset source, which is commonly used to estimate the quality of augmentations [22]

• **Dataset overlap** profile calculates the cardinality of the final dataset identified after augmentation.

**Extending to other data profiles.** Metam can be extended with new profiles to cater for new downstream tasks (e.g., anomaly detection [23], conditional independence for fairness [24]) or to leverage advances in profiling techniques [23], [25], [26], [27].

*Problem Discussion:* We discuss how various discovery-then-augment baselines fail to solve the problem efficiently:

• **Join Everything**: This technique joins $D_{in}$ with every join path identified. This approach may bring irrelevant attributes that deteriorate the utility. Furthermore, the approach is infeasible in large-scale scenarios with thousands of augmentations.

• **Uniform sampling**: This technique samples join paths out of the identified joinable datasets uniformly at random and uses them to augment $D_{in}$. This approach does not guarantee that the chosen sample will improve the utility score.

• **Join Path overlap ranking**: A common technique is to rank join paths based on the cardinality of the augmented datasets (used by S4 [14] and Ver [22]). This technique identifies datasets that contain fewer missing values, but does not guarantee to optimize the task.

• **Using data profiles for selection**: Analysts may use intuition to rank augmentations based on profiles that are expected to be useful. This approach may work if the profiles are accurate and the domain scientist's intuition is correct. But this approach will not work well otherwise, and it relies on accurate estimation of profiles, which are hard to compute in large-scale data repositories, where approximate techniques are often necessary for scalability.

The crux of the problem is that these baselines are unaware of the task, so there is no guarantee that the chosen augmentations will improve utility. Solving goal-oriented data discovery requires an interventional approach that seeks to understand what augmentations *cause* the task's utility to increase.

## III. Intervention-based Querying

In this section, we first describe the limitations of baseline techniques and then discuss how the properties of data, utility function, and solution let us design an efficient technique.

### A. Baseline Interventional Techniques

Goal-oriented data discovery can be solved by enumerating every subset of candidate augmentations, computing their utility, and choosing the minimal subset that achieves utility of at least $\theta$. With $n$ candidate augmentations, this process may require up to $O(2^n)$ queries (does not finish for $n > 30$), so this approach is unfeasible. We consider two improvements:

• **Utility-based selection:** Given a ranking of augmentations (e.g., based on some data profiles) this solution queries, iteratively, in ranking order until the utility obtained is higher than the desired threshold, $\theta$. This solution will be inefficient every time the data profile is not related to the task. And choosing a data profile related to the downstream task is non-obvious for many of the interesting data-driven tasks considered.

• **Prediction from expert advice:** To avoid selecting a profile a priori, different data profiles can be treated as experts and we can use expert selection techniques to solve the problem. For example, the multiplicative weights update method (MW) [28] estimates the ability of different experts to improve utility. MW guarantees the selection of the best expert in hindsight. In its simplest form, considering each profile as a different expert fails to identify combinations of profiles that best

rank augmentations. Enumerating combinations of profiles as experts introduces known combinatorial problems to decision making techniques, including MW [28].

These two techniques have two shortcomings: i) they do not guarantee finding a minimal set of augmentations that optimize the task's utility; ii) they require $O(n)$ queries, with $n$ indicating the number of candidate augmentations. Furthermore, no algorithm can find the optimal solution in less than $(2^n - 1)$ queries in the worst case (Theorem 2).

### B. Towards Efficient Interventional Querying

Even though no technique can be designed to optimize the worst case, we identify several properties of practical scenarios, which help to efficiently identify an optimal (or a close approximation) solution.

**P1 (Optimal solution often contains few augmentations).** Most augmentations are not useful for the task. The number of augmentations, $k$ in the optimal solution is much lower than the total number of augmentations, $k << n$. Therefore, considering smaller subsets of candidate augmentations has a high chance of identifying the optimal solution. This intuition has previously been studied in the combinatorial testing literature. We leverage this insight to prioritize the consideration of smaller subsets over large-sized subsets.

What if P1 does not hold? The scenario where $k$ is not much smaller than $n$ is not realistic, as augmenting thousands of new attributes blows up the space and worsens overall efficiency. Even in this case, the algorithm identifies the optimal solution as it will eventually consider large-sized subsets.

Empirical Validation. Evaluation over 10 different tasks considered in Section VI identified more than 5000 candidate augmentations ($n > 5000$) for each scenario. However, the best solution contained less than 5 augmentations in 8 of the cases and less than 25 in the rest. Therefore, less than $0.5\%$ of the candidate augmentations actually help to improve utility.

**P2 (*Similar Datasets are likely to have similar effect on the utility score*).** Two datasets that contain the same set of tuples have the same influence on the utility score. Even though the presence of duplicates is an extreme case, we observe that open datasets often contain duplicate information. We extend this observation to general datasets by considering their similarity. Augmenting two different datasets that have similar profile values are expected to have similar effect on the utility score with more than $0.5$ probability. This property motivates us to consider the dataset properties to cluster similar augmentations and holistically consider all intra-cluster augmentations for analysis. We use the data profiles introduced in Section II-C. Data profiles are not only useful to cluster the augmentations but also to score them (we refer to these as quality scores).

What if P2 does not hold? This property is used by METAM to prioritize augmentations for querying. In case there is no connection between dataset similarity and their utility score, the identified ordering of augmentations would be the same as a randomized ordering. This would increase the number of required queries to $O(nk)$ in the worst case, where $n$ is the total number of candidate augmentations and $k$ is the number of augmentations in the optimal solution.

Empirical Validation. We compared the difference in utility for augmentations with similarity within $[0.9, 1]$ and found that more than 85% of these have utility difference less than $0.02$. Further, the utility difference increases with reducing similarity. Therefore, highly similar datasets are generally expected to have similar utility.

**P3 (*Monotonicity of the utility function*).** The utility function often satisfies monotonicity with respect to the augmentations, i.e., augmenting new columns to a dataset never worsens the task's utility. For example, causal inference tasks often estimate the total causal impact of identified attributes, which is monotonic. Accuracy and F-score of a Bayes-optimal classifier [29] is also monotonic (as Bayes-optimal classifier ignores the newly added feature if it does not help with prediction). However, certain utility metrics may not be monotonic due to varied reasons e.g. missing values or noise in the newly added attribute. Monotonicity can still be ensured by wrapping the task with a mechanism that ignores an augmentation if it worsens utility. A MONOTONICITY CERTIFICATION component (Figure 2) enforces monotonicity in our framework.

What if P3 does not hold? The monotonicity certification component ignores augmentations that worsen utility. This additional check to verify if adding a new augmentation worsens utility may require additional queries.

Empirical Validation. We evaluated monotonicity of the utility for the considered scenarios and identified that causal inference tasks (what-if and how-to) are always monotonic, and classification tasks are monotonic for more than $60\%$ of the queries. In the remaining $40\%$ of the cases, monotonicity certification component asks additional $\approx 20$ queries and ignores the augmentation that worsens utility to ensures monotonicity.

## IV. GOAL-ORIENTED DISCOVERY ALGORITHM

In this section, we give an overview of METAM in Section IV-A, and discuss the subroutines in Section IV-B. Figure 2 presents the different components of METAM.

### A. METAM Algorithm Overview

METAM takes as input the initial dataset $D_{in}$, a task $t$ and a collection of datasets $\mathcal{D}$ and outputs a minimal set of augmentations that ensure the task utility of the augmented dataset is at least $\theta$ (see Algorithm 1). The algorithm has two main components: i) *candidate generation and likelihood estimation*; ii) *adaptive querying strategy*. The first component identifies the candidate augmentations and computes the vector of data profiles. The second component alternates between two complementary mechanisms that exploit properties P1–P3 to query the task. Alternating between two procedures allows METAM to leverage the best of both techniques and find a solution without assuming anything about the utility function.

**Candidate Generation and likelihood estimation.** First, METAM identifies $\mathcal{P}$, the candidate augmentations for $D_{in}$ (line 1) provided by traditional discovery techniques (Section II-C). Each augmentation is processed to compute its

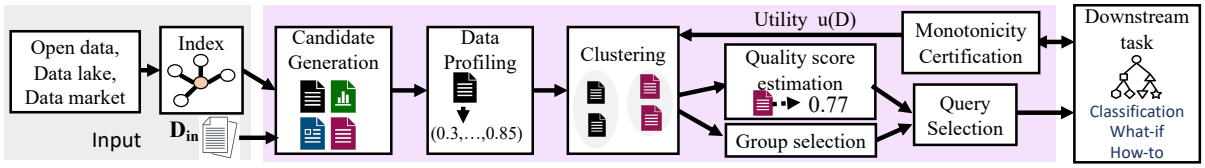



Fig. 2: Overview of METAM's architecture.

**Algorithm 1:** METAM

**Input :** Dataset $D_{in}$, Data Repository $\mathcal{D}$, utility threshold $\theta$
**Output:** $\mathcal{T}$: list of augmentations

1 $\mathcal{P} \leftarrow$ GENERATE-CANDIDATES $(D_{in}, \mathcal{D})$
2 EVALUATE-PROFILE$(\mathcal{P}, D_{in})$
3 $\mathcal{T}^*, \mathcal{T}_c^* \leftarrow \phi$, $t \leftarrow 1$, $D \leftarrow D_{in}$
4 JPSCORE $\leftarrow$ ESTIMATE-QUALITY-SCORES$(\mathcal{P}, D_{in})$
5 $\mathcal{C} \leftarrow$ CLUSTER-PARTITION$(\mathcal{P}, D_{in}, \epsilon)$
6 **while** $u(\Gamma(D, \mathcal{T}^*)) < \theta$ *and* $u(\Gamma(D, \mathcal{T}_c^*)) < \theta$ **do**
7  $\mathcal{X}, Q \leftarrow \phi, i \leftarrow 0$
8  **while** $i < \tau$ *or* $\max_{P \in Q} u'[P] \leq u(D)$ **do**
9  $P_{max} \leftarrow \max_{P_i \in \mathcal{P} \setminus (\mathcal{X} \cup \mathcal{T}^*)}$ JPSCORE$(P_i)$
10  $u'[P_{max}] \leftarrow u(\Gamma(D, \{P_{max}\}))$
11  $\mathcal{X} \leftarrow \mathcal{X} \cup$ CLUSTER$\{P_{max}\}$
12  UPDATE-QUALITY-SCORES$(\mathcal{P}, P_{max}, u')$
13  $\mathcal{P} \leftarrow$ IDENTIFY-GROUP$(\mathcal{C}, t)$
14  **if** $u(\Gamma(D_{in}, \mathcal{P})) > u(\Gamma(D_{in}, \mathcal{T}_c^*))$ **then**
15  $\mathcal{T}_c^* \leftarrow \mathcal{P}$, $u'[\mathcal{T}_c^*] \leftarrow u(\Gamma(D_{in}, \mathcal{T}_c^*))$
16  $i \leftarrow i + 1, Q \leftarrow Q \cup \{P_{max}\}$
17  $P'_{max} \leftarrow \arg\max_{P \in Q \cup \{\mathcal{T}_c^*\}} u'[P]$
18  **if** $u(\Gamma(D, \{P'_{max}\})) > u(D)$ **then**
19  $\mathcal{T}^* \leftarrow \mathcal{T}^* \cup \{P'_{max}\}$
20  $D \leftarrow \Gamma(D, \{P'_{max}\})$
21  **if** CHECK-STOP-CRITERION() **then**
22  break
23 $\mathcal{T}^* \leftarrow \arg\max_{T \in \{\mathcal{T}^*, \mathcal{T}_c^*\}} u(\Gamma(D_{in}, T))$
24 $\mathcal{T} \leftarrow$ IDENTIFY-MINIMAL$(\mathcal{T}^*, \theta)$
25 **return** $\mathcal{T}$

vector of data profiles (line 2). These profiles are first used to cluster candidate augmentations based on the augmentation similarity; augmentations in the same cluster are expected to impact the utility score similarly (Property P2). This property is used by the querying strategy to choose representative augmentations from each cluster and thereby reduce queries to the utility function. The data profiles are also used to calculate quality score of augmentations to estimate their contributions towards task utility. Intuitively, the quality score is a likelihood estimate that helps to query the relevant augmentations earlier. We explain the scoring technique in the implementation details. The clusters and quality scores are used to iteratively query the task to evaluate the utility of the augmentation.

**Adaptive Querying strategy.** In each iteration (lines 8–16), the algorithm interleaves *sequential* and *group querying* to evaluate the utility score of the most promising queries. The sequential mechanism ( highlighted in blue ) estimates the quality score of candidate augmentations and chooses the best augmentation by sequentially querying the task. This procedure processes the augmentations by diversifying the queries across clusters. Specifically, the quality score of an augmentation is the weighted average of profile scores, where the importance of profiles is used as weights. Profile importance is estimated by evaluating the likelihood that higher profile value indeed achieves higher utility. The quality scores are sorted in a non-increasing order to query the task, with a constraint that at most one augmentation is considered from

**Algorithm 2:** CLUSTER-PARTITION

**Input :** Augmentations $\mathcal{P}$, $D_{in}$, cluster radius $\epsilon$
**Output:** $\mathcal{C}$: Clusters of Augmentations

1 $c_1 \leftarrow$ CHOOSE-RANDOM$(\mathcal{P})$
2 $S \leftarrow \{c_1\}, \mathcal{C} \leftarrow \{\mathcal{P}\}$
3 **while** DISTANCE$(\mathcal{C}, S) > \epsilon$ **do**
4  $c \leftarrow$ CHOOSE-FARTHEST$(S)$
5  $S \leftarrow S \cup \{c\}, \mathcal{C} \leftarrow$ ASSIGN$(\mathcal{C}, S)$
6 **return** $\mathcal{C}$

each cluster. After every query of an augmentation $P$, the identified utility score $u(\Gamma(D, \{P\}))$ is used to update the importance of data profiles and the corresponding scores of other augmentations. The number of augmentations that are queried before choosing the augmentation with maximum utility is controlled by a parameter $\tau$. After $\tau$ queries, the augmentation with maximum utility gain ($P'_{max}$) is added to the solution set $\mathcal{T}^*$ and $D$ is updated to $\Gamma(D, \{P'_{max}\})$ for subsequent iterations. The group mechanism ( highlighted in red ) considers subsets of size $t$ (initialized as $t = 1$ in line 3) and evaluates its utility. This approach prioritizes useful clusters over less relevant clusters by using the Thompson sampling [30], [31] mechanism to construct the subsets. We describe this mechanism in Section IV-B. The value of $t$ is increased when all sets of size less than $t$ have been queried.

METAM continues the querying procedure until the utility of augmented dataset is not less than $\theta$ or all augmentations are queried and none of them improve task utility.

**Minimality check.** In the last stage, the best solution among $\mathcal{T}^*$ and $\mathcal{T}_c^*$ is chosen and the identified augmentations are post-processed to identify a minimal set that achieves a utility score of at least $\theta$. This component iteratively removes one augmentation at a time from the solution and evaluates the utility score $u(\Gamma(D_{in}, \mathcal{T}^* \setminus \{T\}))$. If this modified dataset has utility $\theta$ or higher, then $T$ is dropped from the solution set.

**Stopping Criterion.** Algorithm 1 is an anytime algorithm that stops when the task's utility achieves the threshold $\theta$. If $\theta$ is not provided, the algorithm continues until: i) the search space is explored; or ii) the user finds a good solution.

We specify a CHECK-STOP-CRITERION subroutine that tests all user-specified stopping requirements such as time constraint, query budget, solution size constraint, etc.

### *B.* METAM *Subroutines and Generalizations*

We now delve into METAM's details as used by Algorithm 1 to cluster augmentations, estimate quality scores, and propagate the output of a query to other augmentations.

**CLUSTER-PARTITION.** We use a distance based clustering technique to ensure that augmentations within the same cluster have all profile values within a distance of $\epsilon$ from their representative. In other words, Algorithm 2 partitions the space

of augmentations into $2\epsilon$ width cubes (of $l$-dimension, where $l$ is the number of considered profiles) where the cluster representatives form the centers. This procedure is often referred to as an $\epsilon$-cover of the augmentations based on the embedding constructed from their profiles. The parameter $\epsilon$ introduces a trade-off between the number of clusters (which impacts the query complexity) and their quality. Algorithm 2 presents the pseudocode of the clustering algorithm, which adapts the greedy k-center clustering algorithm [32] to increase $k$ until all clusters have radius less than $\epsilon$. The distance between augmentations $P_1$ and $P_2$ is calculated as $d(P_1, P_2) = \max_{i \in R} d(r_1^i, r_2^i)$, where $R$ denotes the set of profiles.

Given a set of candidate augmentations $\mathcal{P}$ and the initial dataset $D_{in}$ as input, the algorithm chooses cluster representatives (also known as centers) to initialize each cluster and then assigns all augmentations to the respective cluster centers. $S$ denotes the centers and $\mathcal{C}$ denotes a partitioning of $\mathcal{P}$ into the respective clusters. The algorithm initializes the first center by randomly choosing an augmentation and assigning all candidate augmentations to this center (line 1). The subsequent centers are identified by choosing the augmentation that has the maximum distance from its center (lines 3–5). After identifying a new center, all augmentations are re-assigned to the updated set of centers. This farthest identification and reassignment procedure continue until the farthest augmentation is at most $\epsilon$ away from its center.

**QUALITY-SCORE Estimation.** An augmentation's quality score ranks it according to the expectation of improving task's utility. This score has two components. First, a *profile-based score*, which is a weighted average of their profile values. This score is equivalent to a prior that is estimated from dataset properties. The pipeline is initialized by assigning equal weight to all profiles and these weights are improved with increasing number of queries. Formally, the importance weight of a profile $p$ is defined as the feature importance of $p$ when used to predict the utility of an augmentation. Second, a *utility-based score*, which indicates the gain in utility score on augmenting a join path. If an augmentation $P$ has been queried previously, the gain in utility is considered as its utility score. If $P$ has not been queried but another join path from the same cluster (say $P'$) has been queried, then, the utility score of $P$ is calculated as the $(1-d(P, P'))$ times the score of $P'$, where $d(P, P')$ is the distance between the augmentations.

The quality score of an augmentation is defined as the sum of profile-based score and utility-based score. This score depends on the importance weights of profiles and the utility score of other augmentations in the same cluster. The UPDATE-QUALITY-SCORES mechanism performs this update.

**IDENTIFY-GROUP.** At a high level, each cluster is characterized by a probability which denotes the likelihood of achieving higher utility on augmentation. Without queries, we do not have an accurate estimate of these probabilities. Querying augmentations from a cluster gives an accurate estimate of the probability but the challenge is to choose between querying a single cluster to get an accurate estimate or diversifying across clusters to explore other options. This follows the traditional explore-exploit dilemma in bernoulli-bandits [31], where the likelihood can be modelled as posterior probability.

We model each cluster as a bandit, where pulling an arm is equivalent to sampling an augmentation. The increase in task utility is the *reward* for a given cluster. We initialize the probability of each arm to $1/|\mathcal{C}|$ and maintain the number of times we get a reward to evaluate the posterior. Each element of the k-sized subset is sampled randomly from the clusters with this probability.

**Generalization: What to do when profiles are not useful?** The clustering algorithm identifies groups of augmentations that have similar profile values which impacts Algorithm 1's query selection. We now analyze the effect of two different types of noises in data profiles on METAM. First, if two augmentations that have similar utility do not have similar data profiles, then these augmentations would be placed in separate clusters and would be independently queried to calculate utility. This case is similar to an approach where clustering is not used. It would not affect METAM's quality, but may lead to increased query complexity. Second, when dissimilar datasets have similar data profiles, two augmentations that have similar profiles may not yield similar utility. To handle such settings, we propose a strategy that adapts based on the quality of profiles. Instead of a single query per cluster, METAM queries $\log |C|$ randomly chosen augmentations from a cluster $C$ to get an accurate estimate of homogeneity with a high probability, *i.e.* it checks if the majority of the sampled augmentations have utility within $(1+\epsilon)$-approximation of the average utility of queried augmentations. If the homogeneity condition does not hold, the quality score estimation ignores the utility score component and considers each element as an independent cluster for subsequent iterations of the algorithm.

**Choosing profiles.** Developers include any profiles that *may* be correlated with the downstream task. They may not know whether such correlation exists and whether it's strong; they "cast a wide net" expecting that some combination of profiles show a correlation. If that is the case, METAM will identify it during the search process. How to balance the number and types of profiles included is beyond the scope of this paper; by default, we include profiles that are effective for machine learning and causal inference tasks. Further, sampling techniques allow to cheapen the computation of profiles.

**Impact of $\tau$.** $\tau$ determines the number of clusters considered before choosing the augmentation with maximum utility gain. Larger values of this parameter guarantees that the augmentation with maximum utility gain is selected before any augmentation with lower utility gain. Intuitively, this approach identifies a small-sized minimal set of augmentations as compared to *any minimal set*. By default, we choose $\tau = |\mathcal{C}|$ to ensure that at least one augmentation from each cluster has been queried. Choosing a smaller value of $\tau$ is the same as relying on the quality score to pick the clusters with maximum quality score. In the extreme case, $\tau = 1$ is equivalent to querying augmentations in non-increasing order of quality

score and selecting any augmentation that improves task utility in the solution set (irrespective of its gain). These settings of $\tau$ should be considered when solution set size need not be optimized. We demonstrate the effect of $\tau$ in Section VI-A.

**Note.** The augmentations that do not help to improve utility may be either erroneous (e.g. incorrect join due to incorrect key) or correct but uninformative for the downstream task. METAM is robust to handle erroneous augmentations in $\mathcal{P}$.

**Risk and Vulnerability.** METAM optimizes to choose attributes that maximize utility. Whenever the utility function does not capture the application requirements, e.g., if the outcome is used by a critical decision-making software that cannot be allowed to be fully automated, data scientist can manually verify the augmentations returned by METAM and flag the augmentations that do not satisfy their requirements. After flagging these augmentations, METAM may have to be rerun to re-optimize for another minimal set of augmentations.

## V. METAM'S PERFORMANCE GUARANTEES

In this section, we show that goal-oriented data discovery is NP-hard and no algorithm can identify the optimal solution in less than $2^n-1$ queries. Because the worst case is highly contrived, we analyze METAM's efficiency in practical settings. We show that METAM identifies a constant-approximation of the optimal solution in $O(\log n)$ queries. We extend the discussion to noisy settings where the discussed properties (Section III-B) may not hold.

To prove the problem is NP-hard, we consider a decision version of the problem and show a reduction from the set-cover problem (formal proof in technical report [33]).

**Theorem 1.** In the worst case, every algorithm identifies an arbitrarily worse solution unless it performs $O(2^n)$ queries.
*Proof.* Let $\mathcal{T}$ denote the set of augmentations such that any of the $2^n-1$ subsets of $\mathcal{T}$ is a valid augmentation. Consider an adversarial utility that outputs $u(\Gamma(D_{in},T))=u(D_{in})$ for all queries until the algorithm has queried $2^n-2$ subsets of $\mathcal{T}$. For the last query it outputs $u(\Gamma(D_{in},T^*))=\theta$. This utility will require $2^n-1$ queries to identify the optimal solution.□

### A. Quality Guarantees

We first show that METAM guarantees finding the optimal solution even in the worst case when allowed to run for $n^k$ queries. Then, we show that METAM identifies a constant-approximation in $O(\log n)$ queries for the practical scenarios discussed in Section III-B.

**Theorem 2.** If $\exists\ \mathcal{T}^*$ such that $u(\Gamma(D_{in},\mathcal{T}^*))\geq\theta$, then METAM's output $\mathcal{T}$ satisfies $u(\Gamma(D_{in},\mathcal{T}))\geq\theta$.
*Proof.* The combinatorial testing component of METAM's querying strategy explores all possible subsets of augmentations until a valid solution set is identified. Therefore, the output of METAM is guaranteed to achieve utility of at least $\theta$. Since the approach considers all subsets of size atmost $k$ and incrementally increases $k$, it identifies the solution in $O(n^k)$ queries where $k$ is the size of the optimal solution. □

**Approximation guarantees.** We first show that METAM identifies a constant approximation of the optimal solution within $O(\log n)$ queries. We assume that the optimal solution contains $k$ augmentations, where $k$ is a constant. To prove this result, we first consider the scenario where properties P2 and P3 hold. Later, we relax these assumptions and show that even when similar datasets yield different utility (P2 does not hold) and the utility is non-monotonic (P3 has to be ensured with a wrapper), METAM identifies an approximate solution.
**Notation.** $\mathcal{C}=\{C_1,\ldots,C_t\}$ denotes the set of cluster centers, $\mathcal{T}^*=\{T_1^*,\ldots,T_k^*\}$ denote the optimal solution, $\mathcal{C}^*$ denote the optimal solution over the cluster centers, and the solution returned by METAM is $\mathcal{T}=\{T_1,\ldots,T_k\}$.

**Theorem 3.** METAM's querying approach identifies a solution with $1/\alpha(1-e^{-\alpha\gamma})-k\epsilon$ approximation in $O(\log n)$ queries.

*Proof.* First, we show that the optimal solution over the set of representatives (cluster centers) is $(1-k\epsilon)$ approximation of the optimal solution over all augmentations $\mathcal{P}$ (Using Lemma 1). Second, we prove that the solution identified by METAM is $1/\alpha(1-e^{-\alpha\gamma})$- approximation of the optimal solution over the centers. Combining these two proofs, we get the desired result. Formally,

$$
\begin{aligned}
&u(\Gamma(D_{in},\mathcal{T})\geq 1/\alpha(1-e^{-\alpha\gamma})u(\Gamma(D_{in},\mathcal{C}^*))\\
&\geq 1/\alpha(1-e^{-\alpha\gamma})(1-k\epsilon)u(\Gamma(D_{in},\mathcal{T}^*)\\
&\geq \left(1/\alpha(1-e^{-\alpha\gamma})-k\epsilon\right)u(\Gamma(D_{in},\mathcal{T}^*)=\left(1/\alpha(1-e^{-\alpha\gamma})-k\epsilon\right)OPT
\end{aligned}
$$

This shows that METAM identifies the approximate solution after $k$ rounds of finding the best augmentation. Additionally, METAM tests property P2 (the composition of each cluster) by querying a random sample of $O(\log n)$ queries from each cluster. This step has an added complexity of $O(|\mathcal{C}|\log n)$ to test homogeneity of the clusters. Lemma 2 shows that $|\mathcal{C}|=O(1/\epsilon^l)$, giving an overall complexity of $O(\log n)$. □

**Lemma 1.** Optimal solution over representatives $\mathcal{C}^*$, $u(\Gamma(D_{in},\mathcal{C}^*))\geq(1-k\epsilon)OPT$.
*Proof.* We apply property of the cluster that the center has utility within a factor of $(1+\epsilon)$ of augmentations in the cluster.

$$
\begin{aligned}
u(\Gamma(D_{in},\mathcal{T}_i^*)) &= u(\Gamma(D_{in},\mathcal{T}_{i-1}^*\cup\{P_i\}))\\
&\quad \text{Cluster property of adding a new augmentation}\\
&\leq (1+\epsilon)u(\Gamma(D_{in},\mathcal{T}_{i-1}^*\cup\{C_i^*\}))\\
&\leq (1+\epsilon)u(\Gamma(D_{in},(\mathcal{T}_{i-2}^*\cup\{C_i^*\})\cup P_{i-1}))\\
&\quad \text{Recursively applying the cluster property}\\
&\leq (1+\epsilon)^k u(\Gamma(D_{in},\mathcal{C}_i^*))\\
&\approx (1-k\epsilon)OPT
\end{aligned}
$$
□

We now bound the number of clusters generated by CLUSTER-PARTITION component of METAM to show that the number of queries grows linearly in the size of solution set.

**Lemma 2.** The number of clusters generated by METAM is $O(1/\epsilon^l)$, where $l$ is the number of considered profiles.
*Proof.* Each join path is represented by $l$ profiles and each profile value is within $[0,1]$. The clustering algorithm chooses a new center whenever the cluster radius is more than $\epsilon$. The l-dimensional space of profiles has unit volume. Therefore, the space can be covered by $O(2^l/\epsilon^l)$ spheres of radius $\epsilon$. □

**Lemma 3.** $u(\Gamma(D_{in}, \mathcal{T})) \geq \frac{1}{\alpha}(1 - e^{-\alpha\gamma})u(\Gamma(D_{in}, \mathcal{T}^*))$, where $\alpha$ and $\gamma$ denote the curvature of $u$ and submodularity ratio, respectively.

*Proof.* Here, we analyze a special case where $\gamma = 1$, i.e. the function is always submodular and present the insights of the general result after the proposition.

$$OPT = u(\Gamma(D_{in}, \mathcal{T}^*)) \leq u(\Gamma(D_{in}, \mathcal{T}^* \cup \mathcal{T}_i)) \text{ (using monotonicity)}$$
$$\leq u(\Gamma(D_{in}, \mathcal{T}_i)) + \sum_{j=1}^{k} \Big(u(\Gamma(D_{in}, \mathcal{T}_i \cup \{T_j^*\})) - u(\Gamma(D_{in}, \mathcal{T}_i))\Big)$$
Since $T_{i+1}$ has the maximum marginal gain
$$\leq u(\Gamma(D_{in}, \mathcal{T}_i)) + \sum_{j=1}^{k} \left(u(\Gamma(D_{in}, \mathcal{T}_i \cup \{T_{i+1}\})) - u(\Gamma(D_{in}, \mathcal{T}_i))\right)$$
$$= u(\Gamma(D_{in}, \mathcal{T}_i)) + k\left(u(\Gamma(D_{in}, \mathcal{T}_i \cup \{T_{i+1}\})) - u(\Gamma(D_{in}, \mathcal{T}_i))\right)$$

Therefore, marginal gain of $T_{i+1}$ is atleast $\frac{1}{k}(OPT - u(\Gamma(D_{in}, \mathcal{T}_i)))$. Let $\delta_i = OPT - u(\Gamma(D_{in}, \mathcal{T}_i))$. Therefore, marginal gain of $T_{i+1} = \delta_i - \delta_{i+1}$. Using this in the above equation, $\delta_i - \delta_{i+1} \geq \frac{1}{k}\delta_i$. Therefore, $\delta_{i+1} \leq (1 - 1/k)\delta_i$. Applying this recursively, we get $\delta_k \leq (1-1/k)^k\delta_0 \leq \frac{1}{e}\delta_0 = \frac{1}{e}OPT$. Therefore, $u(\Gamma(D_{in}, \mathcal{T}_k)) = (1 - 1/e)OPT$. □

This proof is extended from the traditional result for submodular optimization [34]. In case of general utility functions, the above proof extends using the techniques in [35].

**Dependence on $\alpha$ and $\gamma$.** METAM does not use $\alpha$ and $\gamma$ as input and these notions are used only to characterize the class of utility functions that can be analyzed. The approximation ratio of this result is dependent on the curvature and submodularity ratio of the utility function, which capture the likelihood that the utility function satisfies submodularity. Recent work has studied estimation of these parameters for several real-world functions like linear regression, sparse feature selection, Bayesian A-optimality, determinantal functions, and linear programs with combinatorial constraints [36], [35]. However, it is an open problem to bound the ratios for general functions.

**Noisy Clusters.** In case where the identified clusters are not homogenous (property P2 does not hold), METAM ignores all clusters (considers each augmentation as its own cluster). In this case, we show that METAM's quality score estimate is accurate to order the candidate augmentations after $O(\log n)$ queries (proof in the technical report [33]). Therefore, METAM identifies useful augmentations within $O(\log n)$ queries. Further, in the worst case when the estimated quality scores are also inaccurate due to noise in data, METAM requires $O(n)$ queries to choose one augmentation, which is repeated $k$ times, yielding an overall complexity of $O(nk)$.

## VI. EXPERIMENTAL EVALUATION

In this section, we answer the following research questions.

- **RQ1:** Does METAM identify augmentations that improve the utility of the downstream task? (Section VI-A).
- **RQ2:** Does METAM scale with the number of augmentations and data profiles? (Section VI-B).
- **RQ3:** How sensitive is METAM to profile informativeness and parameter choice? (Section VI-C)

| Dataset | #Tables | #Columns | #Joinable Columns | Size |
|---|---|---|---|---|
| Open-Data | 69K | 29.5M | 28.6M | 119G |
| Kaggle | 1950 | 91231 | 6.7M | 18G |

TABLE I: Characteristics of Datasets

**Datasets.** We consider a diverse collection of datasets (schools, taxi, grocery, pharmacy, crime, housing prices, etc.) from two real-world data repositories.

- **Open Data** around 69K datasets comprising datasets from Open Data Portal Watch, which catalogs 262 open data portals such as NYC Open data, finances.worldbank.org, etc.
- **Kaggle** [37] contains around 1950 datasets identified by crawling different competitions.

**Settings.** We implement METAM in Python and run all experiment on a server with 187GB RAM. The join path index was computed offline using Aurum [12] with the default parameter settings. Unless specified, $\epsilon$ is set to 0.05 and $\tau$ is set to the number of identified clusters. We study the effect of these parameters in Section VI-C. We generate all data profiles (Section II-C) on a random sample of 100 records.

### A. Modern Data Science Toolkit

Predictive (machine learning) and prescriptive (causal inference) analytics form the core of modern data analysis. We evaluate METAM on both types.

- **Machine Learning**: We consider: i) two classification tasks; ii) a fair-ML task that aims to ensure fairness while maintaining high accuracy; and iii) a regression task.
- **Causal Inference**: We consider *what-if* and *how-to* analysis. What-if analysis answers hypothetical reasoning questions, e.g., "What will be affected if the average revenue increases by 20%? ". How-to analysis answers questions such as"How can I achieve revenue more than 20%?". These tasks form the core of prescriptive analytics [8], [9], [10], [11].

First, we measure METAM's performance on these tasks and compare it to other baselines. Second, we demonstrate METAM generalizes to other tasks by implementing entity linking, clustering, and fair ML. We demonstrate METAM augments the quality of all the above tasks and outperforms baselines without human intervention.

**Baselines.** No prior work studies goal-oriented data discovery so we adapt prior techniques (see Section III-A) as baselines.

- Prediction from expert advice (`MW`): We use the randomized version of the multiplicative weights update algorithm that chooses an expert proportionally to its weight.
- Overlap ranking-based approach (`Overlap`) queries the augmentations in non-increasing order of overlap with $D_{in}$, a mechanism commonly used in prior approaches [14].
- Uniform sampling approach (`Uniform`) samples queries from the candidate set uniformly at random.

We consider ARDA [38] as a task-specific baseline for classification and regression tasks.

*1) Focus on minimizing the size of solution set:* We configure METAM to find the smallest possible augmentation that boosts the task's utility. Smaller solutions are easier to interpret and thus are highly desirable. Figure 3 shows that METAM

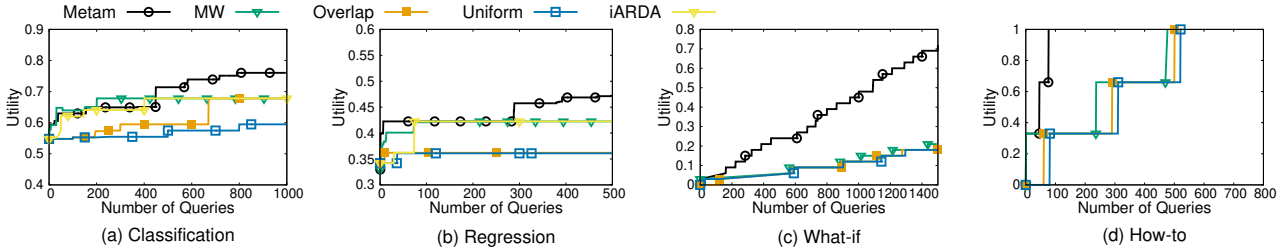



Fig. 3: Comparison of METAM with baselines (iARDA is run for supervised ML only) for four tasks on open data repository.

*outperforms all baselines across all four tasks*. It requires the lowest number of queries yet achieves the highest utility.

**Classification (Price).** We predict house prices across different regions in New York, California and Illinois. We downloaded around 1000 records by searching for randomly chosen zipcodes in these states on Redfin [39]. The task learned a random forest classifier to predict if the house price was low or high. In around 270 queries, the utility achieved by METAM rises from 0.69 to 0.84. The three most promising augmentations identified by METAM include i) presence of Walmart in the same zipcode; ii) number of taxi trips; and iii) crime information (especially for prediction within Chicago). Some of the identified augmentations such as "presence of a grocery store nearby" are quite intriguing and we found many studies [5], [40] that have shown their importance for predicting house prices. Most importantly, this results were obtained without human intervention beyond pointing METAM to the datasets and giving it a task.

**Classification (Schools).** The goal is to predict school performance on a standardized test [38] consisting of around 1800 records. The task trains a random forest classifier and computes the F-score on a validation dataset as the utility score. The join path index identified more than 9000 augmentations from the data repository. We sampled 100 candidate augmentations and identified that 60% of them are erroneous, e.g. joining datasets with incorrect keys. In addition to prior baselines, we adapted ARDA [38], a prior augmentation technique for ML to the interventional setting (denoted by iARDA) where augmentations are queried in decreasing order of ranking returned by [38].

METAM outperforms all baselines. It obtains 0.65 and 0.75 utility in 200 and 700 queries, respectively. This is in contrast to `MW` and `iARDA` that require each more than 1200 queries. In this dataset, correlation and mutual information of the augmented column with the target attribute are identified as the most informative data profiles by METAM. In each iteration, `MW` considers a single data profile to choose the query, while METAM combines all profiles to estimate a quality score. Therefore, METAM considers augmentations that have higher mutual information and correlation values early in the discovery process.

Classification with AutoML: We considered the schools performance dataset and used three AutoML libraries (Autosklearn [41], TPOT [42], and PyCaret [43]) to implement the classifier. Figure 4 (a) presents the result for TPOT library. The performance of METAM improves the utility of the learned classifier from 0.57 to 0.75 in less than 1000 queries while all other baselines achieve utility of 0.70. We observe similar results for other AutoML task implementations.

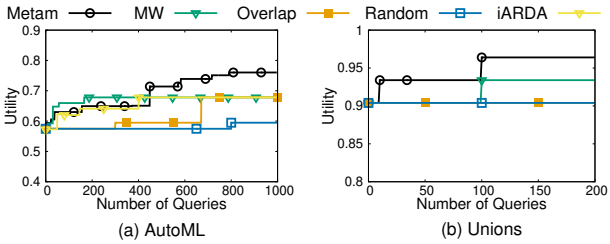



Fig. 4: Comparison of METAM and other baselines for (a) classification with AutoML and (b) Unions (addition of records).

Adding additional records (Unions): Prior experiments considered addition of new attributes. In this experiment, we considered adding additional records and attributes. Addition of records is often known as unions in data discovery literature. We considered a house rent prediction task for houses in NYC. We used the approach from [15] to create additional augmentation candidates for unions. Figure 4 (b) compares the utility of the trained classifier when the base classifier is augmented with additional rows. METAM improves utility from 0.90 to 0.96 in around 100 queries while all other baselines reach 0.93 utility. On further augmenting new attributes to this augmented dataset, utility improved by 1% to 0.974.

Optimizing the classifier with an expert: We tried to optimize the classifier by manually performing feature engineering (using trial and error) on the initial dataset for 3hrs. We were able to improve the utility of base classifier from 0.57 (achieved by AutoML) to 0.62. However, augmenting external data over this optimized improved its utility to more than 0.75. This shows that augmenting external data can bring in new attributes that may be highly predictive. Optimizing feature engineering and model learning may not achieve similar gains when the initial set of features are not extremely predictive.

**Regression.** The goal is to predict number of collisions in NYC using data such as number of daily taxi trips [38] over a dataset containing 350 records.. The task uses a random forest regressor and computes the mean absolute error (MAE), returning $1-MAE$ as utility. METAM outperforms all baselines. With only 300 queries, METAM reduces MAE from 0.66 to 0.55. Other techniques require three times more queries to achieve similar MAE. Notably, the utility achieved on these tasks is comparable to the values reported in prior work that use these datasets [38], demonstrating effectiveness of METAM to achieve competitive quality while generalizing to a wide variety of tasks.

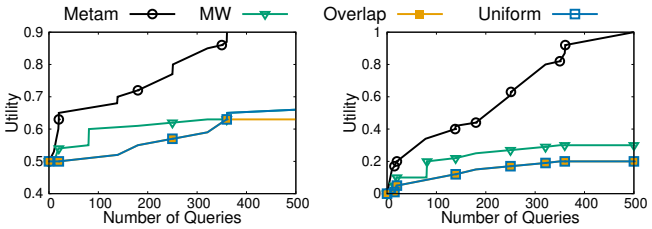



Fig. 5: Average result for a semi-synthetic evaluation

**What-if analysis.** The task takes an input dataset along with a hypothetical update, and calculates the causal impact of the update query on other attributes. We consider an initial table containing SAT scores of 450 students [44] and ask what attributes would be causally affected if "critical reading score" of students is updated. Understanding the attributes sheds light on what affects students' reading score, paving the way for the implementation of interventions. The task implements a causal discovery algorithm [45] and its quality is measured as the fraction of correctly identified attributes (p-value $\leq 0.05$).

Figure 3(c) shows that METAM reaches a utility of 0.70 in less than 1500 queries while all other baselines achieve less than 0.25 utility score. METAM reaches utility score of 1 in around 2000 queries while MW requires more than 8000 queries. One of the main reasons for improved performance of METAM is the clustering procedure, which allows METAM to prune out queries that are expected to have similar utility as that of previously queried augmentations (Property P2).

**How-to analysis**. We want to identify what attributes to update to maximize students' SAT score of 450 students. The task computes causal dependencies and returns the fraction of correctly identified attributes. The candidate set contains 240 augmentations. The optimal solution contains three attributes that have a strong causal impact on the reading score and METAM identifies this set of attributes in less than 100 queries, while all other baselines require more than 500 queries.

*2) Allow any size of the solution:* We relax the requirement of finding a minimal set of augmentations. The algorithm is allowed to choose the first augmentation that improves task utility rather than choosing the best augmentation. METAM consistently outperforms baselines for any stopping criterion. MW requires fewer queries with $\tau = 1$ than when optimizing the set of the solution set, but METAM still outperforms this baseline. In this relaxed setting, the solution set contains $\approx 9$ augmentations as compared to 2 augmentations in the experiment that minimizes the solution set size.

*3) Results are Significant:* In this experiment, we randomly sampled five different augmentations for a randomly chosen dataset in the repository (say $D$) and synthesized a new column in $D$ using these augmentations as i) the prediction attribute for a classification task and ii) the outcome variable for how-to analysis task. This modified version of $D$ was considered as the input table to test the presented techniques. We considered 100 different instantiations to present average statistics. Figure 5 shows that METAM consistently outperforms other techniques and is highly effective in quickly discovering useful augmentations. MW is better than other baselines as it is able to identify a useful augmentation whenever it is ranked as one of the highest by any of the data profiles.

| Dataset | METAM | MW | Overlap | Uniform |
|---|---|---|---|---|
| Schools (C) | **0.80** | 0.20 | 0.0 | 0.20 |
| Taxi (C) | **1** | 0.5 | 0.5 | 0.5 |
| Crime (C) | **0.90** | 0.20 | 0.1 | 0.1 |
| Housing prices (C) | **0.75** | 0.25 | 0.0 | 0.25 |
| Pharmacy | **0.95** | 0.43 | 0.43 | 0.25 |
| Grocery stores | **0.92** | 0.37 | 0.10 | 0.17 |

TABLE II: Utility of METAM and other baselines in less than 1000 queries. Datasets labelled with (C) perform a causal analysis task and others perform data analytics.

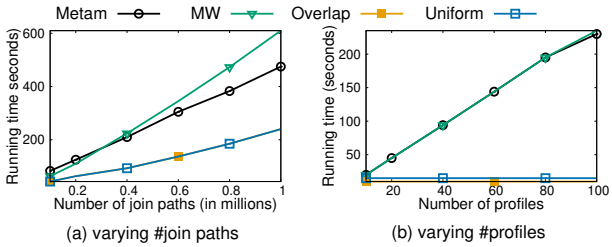
Table II shows the utility achieved by METAM and other baselines for a diverse set of datasets chosen by shortlisting the most accessed datasets on respective data sources. For example, crime data is the most accessed in Chicago's open data, Pharmacy information is one of the most accessed in Pennsylvania's open data portal. Across all datasets, METAM is capable of identifying augmentations that achieve the highest utility score. Among baselines, MW vastly underperforms METAM even after using considerably more queries.

*4) Generalization to other tasks:* **Entity Linking** aims to link entities [46] in the input dataset to their corresponding entities in knowledge graphs such as Wikidata or DBPedia [47].We consider a CDC dataset [37] containing statistics about different cities across the US. Each record contains city name, state abbreviation, *e.g.* AL for Alaska. The task is to link all city names to their corresponding entities and evaluate accuracy as the utility metric. The task searches for every token on Wikidata and chooses the corresponding entity if it has a unique match. However, city names like Birmingham have multiple Wikidata entities as it is a city in the UK and in the state of Alabama. Augmenting state names to this dataset provides context to the entity linking approach for better linking. We consider the kaggle data repository and identified around 185 augmentations. METAM identified useful augmentations in 4 queries while MW required 10 queries and all other baselines required more than 40 queries.

**Clustering.** This evaluation considers a list of different raw materials and their categories (vegetable, fruit, spices, etc) collected from a health blogging website. The downstream task clusters the products based on their satiety score and returns the additive inverse of the largest cluster radius as the utility. The dataset identifies 8 augmentations in the repository, one of which augments Optimal nutrient intake (ONI) score of each ingredient. This score is highly correlated with the ground-truth clusters and therefore help to improve clustering quality. Given the small set of candidates, all techniques have a similar performance on this dataset, requiring $\approx 4$ queries.

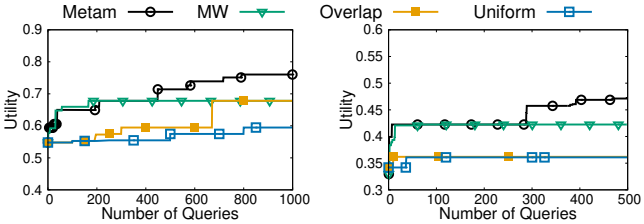
### B. Efficiency and Scalability

The number of queries considered in the above experiments are a proxy to evaluate the time taken to run the pipeline. METAM is efficient and identifies the set of augmentations in less than 10 min across all datasets. Most time is spent




(a) varying #join paths (b) varying #profiles

Fig. 6: Execution of METAM and other baselines for (a) varying the set of joinable datasets and (b) data profiles.




(a) Classification (b) Regression

Fig. 7: Comparison of METAM with other techniques using task-specific profiles from [38].

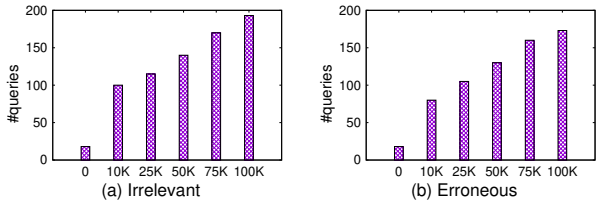
identifying a candidate set of augmentations and their data profiles. Note that the profiles have varied complexity, e.g. correlation and mutual information are more complex than dataset overlap. Out of 10 minutes, roughly 4 minutes are spent on generating data profiles for the candidate augmentations. For a fixed number of queries, `MW` requires the same amount of time than METAM. The other baselines run slightly faster, taking 7 minutes.

To evaluate scalability, we vary the number of candidate augmentations and data profiles and measure the time needed to run 1000 queries. Figure 6 (a) shows that all techniques scale linearly with the number of joinable datasets in the repository. The time taken by `MW` increases faster than METAM due to $O(n \log n)$ time taken to sort augmentations. In contrast, METAM clusters augmentations in $O(n)$ time and uses identified clusters for efficient computation. Although `Overlap` and `Uniform` are faster, they vastly underperform the other baselines, as previously shown. Figure 6 (b) shows that METAM and `MW` scale linearly with the number of profiles, while the time taken by `Overlap` and `Uniform` does not change as they do not use any of the input profiles. In summary, METAM processes $1M$ augmentations in less than 10 minutes and scales linearly with the number of augmentations.

### C. Effect of Profile Informativeness and Parameters

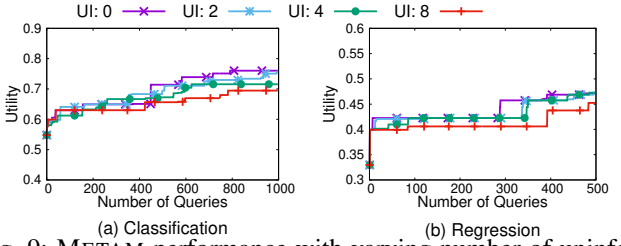
In this section, we first evaluate the impact of incorrect candidate augmentations (false positives) on overall performance of METAM. We then evaluate the impact of (a) adding informative profiles (b) uninformative ones, and (c) removing informative profiles on METAM's ability to improve downstream utility and overall efficiency. We then evaluate an extreme scenario where all profiles are uninformative. Lastly, we perform an ablation analysis to understand the impact of different components of METAM and their parameters.

**Do incorrect augmentations impact METAM's performance?** In this experiment, we considered a single ground-truth augmentation for classification and regression datasets. Figure 8(a) fixes erroneous augmentations (incorrect join path identified by candidate generate algorithm) to 100 and varies correct but irrelevant augmentations (do not help task utility) and Figure 8(b) fixes the irrelevant augmentations but varies erroneous augmentations. We observed that METAM identified the ground truth augmentation and discards all incorrect and irrelevant augmentations within 200 queries. The number of required queries increases with increasing number of irrelevant or erroneous augmentations.




(a) Irrelevant (b) Erroneous

Fig. 8: Number of queries required to identify ground truth with varying irrelevant and erroneous augmentations.




(a) Classification (b) Regression

Fig. 9: METAM performance with varying number of uninformative profiles. The number of informative profiles is fixed to be 5 and uninformative are denoted by UI.

**Adding informative profiles.** In this experiment, we add informative task-specific data profiles using Arda [38]. Figure 7 shows that METAM requires fewer queries than `MW` and other baselines. In fact, METAM achieves utility of 0.68 in less than 200 queries, as opposed to more than 400 queries when these specialized data profiles are not used. Informative profiles boost METAM's performance.

**Adding uninformative profiles.** This experiment considers the initial set of profiles (as described in Section IV) and generates additional uninformative profiles. Figure 9 shows that METAM achieves similar utility gains across all settings and adding noisy profiles does not affect the quality of the solution because METAM learns to ignore them at the cost of running a few more queries.

**Removing profiles.** We consider the classification and regression tasks with 10 data profiles, 5 of which are uninformative. First, the uninformative profiles are removed, followed by removal of other informative profiles. Figure 10 shows that removing irrelevant features (until the number of removed features is less than 5) improves the progressive growth in task utility with respect to the number of queries. On further removing any of the relevant profiles, the number of required queries to reach a similar utility increases.

**What if all profiles are uninformative?** In this experiment, we considered the schools dataset and generated all profiles randomly. In this case, the quality score estimated by METAM

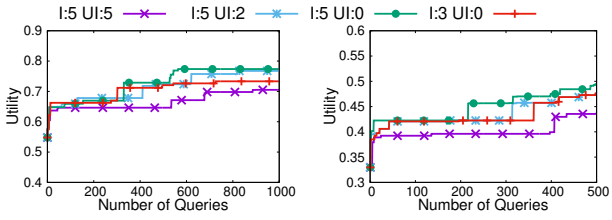



Fig. 10: Effect of removing profiles on METAM quality vs query tradeoff. I denotes the number of informative profiles.

is similar to a random ordering. However, METAM identifies the optimal set of augmentations, but the number of queries is similar to that of uniform baseline.

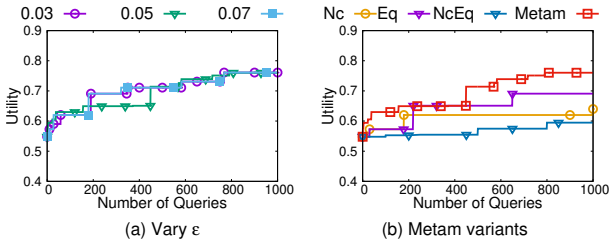



Fig. 11: Ablation analysis $\epsilon$, and METAM variants.

**METAM and its variants.** Figure 11 (b) compares METAM with three variants, (i) Eq: This variant is equivalent to ignoring Thompson sampling procedure and ranks each cluster with equal importance. (ii) Nc: considers each augmentation as a different cluster, which is equivalent to ignoring property P2. (iii) NcEq: considers each augmentation equally important and ignores clustering. METAM outperforms all variants because Eq and NcEq process the candidate set of augmentations in a random order while Nc wastes queries on redundant augmentations that are not expected to help with the downstream utility (which are clustered together by METAM). Therefore, considering clustering and ranking mechanisms together allows METAM to prioritize important augmentations and efficiently ignore redundant ones.

**Impact of $\epsilon$.** Figure 11 (a) measures the quality of METAM when varying $\epsilon$ for clustering. Increasing the value of $\epsilon$ reduces the number of clusters in the dataset and increases the distance between intra-cluster augmentations. Therefore, the property P2 may be violated in such a case and METAM may ignore the cluster structure for quality score estimation. On the contrary, choosing a smaller value of $\epsilon$ means that all augmentations that are present in the same cluster are very similar with respect to the considered augmentations and therefore, all augmentations within a cluster may have similar effect on the utility of the downstream task. Figure 11 shows that METAM the number of queries do not change drastically on varying $\epsilon$.

## VII. Related Work

**Data Discovery Approaches.** Data discovery systems and libraries [22], [48], [12], [49], [17], [18], [16], [50], [51], [52] compute join paths from data repositories automatically. Hence, given a query dataset, these systems tell users what other datasets join with the query dataset. Many modern techniques allow users to interactively search useful joinable datasets [53] by specifying requirements in the form of data profiles or summaries and explanations [54]. However, without the information of user's goal, it is difficult to identify which join path is useful to the downstream task, as we have demonstrated in the evaluation section.

**Human-in-the-Loop Data Integration.** Human-in-the-loop approaches [55], [56], [57] utilize user feedback to steer the search process. These techniques have been applied to problems ranging entity resolution [58], [59], [60], [61], join selection, and entity alignment, among others [62]. These approaches' focus is to reduce the effort necessary to find solutions. In contrast, METAM focuses on automatic data discovery and augmentation that does not require user intervention.

**Data and Machine Learning (ML).** There are data augmentation techniques [38], [63], [64], [65], [66], [67] designed for ML tasks that either select attributes and data points that are useful for prediction or learn relational embeddings to embed data into a vector space. These techniques are not interventional, and inherently rank available datasets based on their importance, which can be used as data profiles for METAM. Therefore, the contributions of this work are orthogonal to those of prior techniques that rank features or augmented datasets based on their importance. Additionally, METAM is not restricted to machine learning tasks. Instead, it implements *goal-oriented data discovery* and consequently can steer data discovery for any task for which developers can provide a function that computes the utility score. Furthermore, METAM dynamically changes the weights of different profiles, making it more robust to data quality issues in join paths and profiles.

**Query-Driven Tasks.** Query-Driven approaches [68], [69] focus on reducing the calculation of tasks like entity resolution based on the requirements of a concrete query. These approaches solve the efficiency problem in an online setting and require a well-defined query a priori. However, it is impossible to have a well-defined query in data discovery problems, i.e. usually analysts do not know clearly which portion of the data collection is useful for their task.

## VIII. Conclusions

We proposed a novel approach to perform goal-oriented data discovery that runs interventional queries to efficiently identify useful augmentations. Our approach leveraged properties of the data, utility function and the solution set to optimize the querying strategy. We analyzed the optimality and the query complexity of our approach. Empirical analysis on a diverse set of datasets from different application scenarios and comparison with baselines demonstrate versatility and effectiveness to efficiently select useful augmentations.

## IX. Acknowledgements

We thank the Chameleon cloud platform for providing the resources necessary to conduct this research. This work was partially supported by the NSF Convergence Accelerator (Award Number #2040718), NSF grant #2030859 and the Griffin Applied Economics Incubator at UChicago.

## REFERENCES


[1] "Data.gov https://www.data.gov." [Online]. Available: https://www.data.gov

[2] A. Y. Halevy, F. Korn, N. F. Noy, C. Olston, N. Polyzotis, S. Roy, and S. E. Whang, "Managing google's data lake: an overview of the goods system." *IEEE Data Eng. Bull.*, vol. 39, no. 3, pp. 5–14, 2016.

[3] "Dawex, https://www.dawex.com/en/."

[4] R. C. Fernandez, P. Subramaniam, and M. J. Franklin, "Data market platforms: trading data assets to solve data problems," *PVLDB*, vol. 13, no. 12, pp. 1933–1947, 2020.

[5] D. G. Pope and J. C. Pope, "When walmart comes to town: Always low housing prices? always?" *Journal of Urban Economics*, vol. 87, pp. 1–13, 2015.

[6] "Economists find walmart stores increase nearby home-prices https://abcnews.go.com/Business/economists-find-walmart-stores-increase-nearby-home-prices/story?id=16560451."

[7] "Grocery stores affect home-values https://www.homes.com/blog/2020/08/how-do-grocery-stores-affect-home-values/."

[8] A. Meliou and D. Suciu, "Tiresias: the database oracle for how-to queries," in *Proceedings of the 2012 ACM SIGMOD International Conference on Management of Data*, 2012, pp. 337–348.

[9] T. Software, "Tableau https://www.tableau.com/." [Online]. Available: https://www.tableau.com/

[10] "Microsoft power bi https://powerbi.microsoft.com/en-us/." [Online]. Available: https://powerbi.microsoft.com/en-us/

[11] "Domo https://www.domo.com/." [Online]. Available: https://www.domo.com/

[12] R. C. Fernandez, Z. Abedjan, F. Koko, G. Yuan, S. Madden, and M. Stonebraker, "Aurum: A data discovery system," in *34th IEEE International Conference on Data Engineering, ICDE 2018, Paris, France, April 16-19, 2018*. IEEE Computer Society, 2018, pp. 1001–1012. [Online]. Available: https://doi.org/10.1109/ICDE.2018.00094

[13] E. Zhu, D. Deng, F. Nargesian, and R. J. Miller, "Josie: Overlap set similarity search for finding joinable tables in data lakes," in *Proceedings of the 2019 International Conference on Management of Data*, ser. SIGMOD '19. New York, NY, USA: Association for Computing Machinery, 2019, p. 847–864.

[14] F. Psallidas, B. Ding, K. Chakrabarti, and S. Chaudhuri, "S4: top-k spreadsheet-style search for query discovery," in *Proceedings of the 2015 ACM SIGMOD International Conference on Management of Data, Melbourne, Victoria, Australia, May 31 - June 4, 2015*, T. K. Sellis, S. B. Davidson, and Z. G. Ives, Eds. ACM, 2015, pp. 2001–2016. [Online]. Available: https://doi.org/10.1145/2723372.2749452

[15] F. Nargesian, E. Zhu, K. Q. Pu, and R. J. Miller, "Table union search on open data," *Proceedings of the VLDB Endowment*, vol. 11, no. 7, pp. 813–825, 2018.

[16] A. Santos, A. Bessa, F. Chirigati, C. Musco, and J. Freire, "Correlation sketches for approximate join-correlation queries," *Proceedings of the 2021 International Conference on Management of Data*, Jun 2021. [Online]. Available: http://dx.doi.org/10.1145/3448016.3458456

[17] R. C. Fernandez, J. Min, D. Nava, and S. Madden, "Lazo: A cardinality-based method for coupled estimation of jaccard similarity and containment," in *35th IEEE International Conference on Data Engineering, ICDE 2019, Macao, China, April 8-11, 2019*. IEEE, 2019, pp. 1190–1201. [Online]. Available: https://doi.org/10.1109/ICDE.2019.00109

[18] E. Zhu, F. Nargesian, K. Q. Pu, and R. J. Miller, "LSH ensemble: Internet-scale domain search," *Proc. VLDB Endow.*, vol. 9, no. 12, pp. 1185–1196, 2016. [Online]. Available: http://www.vldb.org/pvldb/vol9/p1185-zhu.pdf

[19] I. Trummer, "Can deep neural networks predict data correlations from column names?" *CoRR*, vol. abs/2107.04553, 2021. [Online]. Available: https://arxiv.org/abs/2107.04553

[20] K. Hlaváčková-Schindler, M. Paluš, M. Vejmelka, and J. Bhattacharya, "Causality detection based on information-theoretic approaches in time series analysis," *Physics Reports*, vol. 441, no. 1, pp. 1–46, 2007.

[21] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "Bert: Pre-training of deep bidirectional transformers for language understanding," *arXiv preprint arXiv:1810.04805*, 2018.

[22] Y. Gong, Z. Zhu, S. Galhotra, and R. C. Fernandez, "Ver: View discovery in the wild," *CoRR*, vol. abs/2106.01543, 2021. [Online]. Available: https://arxiv.org/abs/2106.01543

[23] S. Brin, R. Motwani, and C. Silverstein, "Beyond market baskets: Generalizing association rules to correlations," in *Proceedings of the 1997 ACM SIGMOD international conference on Management of data*, 1997, pp. 265–276.

[24] B. Salimi, L. Rodriguez, B. Howe, and D. Suciu, "Interventional fairness: Causal database repair for algorithmic fairness," in *Proceedings of the 2019 International Conference on Management of Data*, 2019, pp. 793–810.

[25] C. Chen, B. Golshan, A. Y. Halevy, W.-C. Tan, and A. Doan, "Biggorilla: An open-source ecosystem for data preparation and integration." *IEEE Data Eng. Bull.*, vol. 41, no. 2, pp. 10–22, 2018.

[26] J. d. J. Flores Herrera, S. Nadal Francesch, and Ó. Romero Moral, "Towards scalable data discovery," in *Advances in Database Technology: EDBT 2021, 24th International Conference on Extending Database Technology: Nicosia, Cyprus, March 23-26, 2021: proceedings*. OpenProceedings, 2021, pp. 433–438.

[27] Z. Abedjan, L. Golab, and F. Naumann, "Profiling relational data: a survey," *The VLDB Journal*, vol. 24, no. 4, pp. 557–581, 2015.

[28] S. Arora, E. Hazan, and S. Kale, "The multiplicative weights update method: a meta-algorithm and applications," *Theory of Computing*, vol. 8, no. 1, pp. 121–164, 2012.

[29] R. O. Duda, P. E. Hart, and D. G. Stork, *Pattern classification and scene analysis*. Wiley New York, 1973, vol. 3.

[30] W. R. Thompson, "On the likelihood that one unknown probability exceeds another in view of the evidence of two samples," *Biometrika*, vol. 25, no. 3-4, pp. 285–294, 1933.

[31] D. J. Russo, B. Van Roy, A. Kazerouni, I. Osband, Z. Wen *et al.*, "A tutorial on thompson sampling," *Foundations and Trends® in Machine Learning*, vol. 11, no. 1, pp. 1–96, 2018.

[32] T. F. Gonzalez, "Clustering to minimize the maximum intercluster distance," *Theoretical computer science*, vol. 38, pp. 293–306, 1985.

[33] https://github.com/sainyamgalhotra/sainyamgalhotra.github.io/blob/main/metam_techreport.pdf, "Metam: Goal-oriented data discover."

[34] G. L. Nemhauser, L. A. Wolsey, and M. L. Fisher, "An analysis of approximations for maximizing submodular set functions—i," *Mathematical programming*, vol. 14, no. 1, pp. 265–294, 1978.

[35] A. A. Bian, J. M. Buhmann, A. Krause, and S. Tschiatschek, "Guarantees for greedy maximization of non-submodular functions with applications," in *International conference on machine learning*. PMLR, 2017, pp. 498–507.

[36] A. Das and D. Kempe, "Approximate submodularity and its applications: Subset selection, sparse approximation and dictionary selection," *The Journal of Machine Learning Research*, vol. 19, no. 1, pp. 74–107, 2018.

[37] "kaggle, https://www.kaggle.com/."

[38] N. Chepurko, R. Marcus, E. Zgraggen, R. C. Fernandez, T. Kraska, and D. Karger, "ARDA: automatic relational data augmentation for machine learning," *Proc. VLDB Endow.*, vol. 13, no. 9, pp. 1373–1387, 2020. [Online]. Available: http://www.vldb.org/pvldb/vol13/p1373-chepurko.pdf

[39] "Redfin https://www.redfin.com/."

[40] P. Caporaso, "Taxi activity as a predictor of residential rent in new york city," Ph.D. dissertation, Massachusetts Institute of Technology, 2019.

[41] "Auto-sklearn automl https://automl.github.io/auto-sklearn/master/."

[42] "Tpot automl http://epistasislab.github.io/tpot/using/."

[43] "Pycaret https://pycaret.org/."

[44] "Nyc open data, https://opendata.cityofnewyork.us/."

[45] "causal-learn https://causal-learn.readthedocs.io/en/latest/."

[46] S. Galhotra and U. Khurana, "Semantic search over structured data," in *Proceedings of the 29th ACM International Conference on Information & Knowledge Management*, 2020, pp. 3381–3384.

[47] "Dbpedia, https://www.dbpedia.org/."

[48] M. Yakout, K. Ganjam, K. Chakrabarti, and S. Chaudhuri, "Infogather: Entity augmentation and attribute discovery by holistic matching with web tables," ser. SIGMOD '12, 2012, p. 97–108.

[49] S. Castelo, R. Rampin, A. S. R. Santos, A. Bessa, F. Chirigati, and J. Freire, "Auctus: A dataset search engine for data discovery and augmentation," *Proc. VLDB Endow.*, vol. 14, no. 12, pp. 2791–2794, 2021. [Online]. Available: http://www.vldb.org/pvldb/vol14/p2791-castelo.pdf

[50] M. Esmailoghli, J. Quiané-Ruiz, and Z. Abedjan, "COCOA: correlation coefficient-aware data augmentation," in *Proceedings of the 24th International Conference on Extending Database Technology, EDBT 2021, Nicosia, Cyprus, March 23 - 26, 2021*, Y. Velegrakis, D. Zeinalipour-Yazti, P. K. Chrysanthis, and F. Guerra, Eds.

OpenProceedings.org, 2021, pp. 331–336. [Online]. Available: https://doi.org/10.5441/002/edbt.2021.30

[51] F. Nargesian, K. Q. Pu, B. G. Bashardoost, E. Zhu, and R. J. Miller, "Data lake organization," *arXiv preprint arXiv:1812.07024*, 2018.

[52] F. Nargesian, K. Q. Pu, E. Zhu, B. Ghadiri Bashardoost, and R. J. Miller, "Organizing data lakes for navigation," in *Proceedings of the 2020 ACM SIGMOD International Conference on Management of Data*, 2020, pp. 1939–1950.

[53] Y. Zhang and Z. G. Ives, "Finding related tables in data lakes for interactive data science," in *Proceedings of the 2020 ACM SIGMOD International Conference on Management of Data*, 2020, pp. 1951–1966.

[54] A. Santos, S. Castelo, C. Felix, J. P. Ono, B. Yu, S. R. Hong, C. T. Silva, E. Bertini, and J. Freire, "Visus: An interactive system for automatic machine learning model building and curation," in *Proceedings of the Workshop on Human-In-the-Loop Data Analytics*, 2019, pp. 1–7.

[55] G. Li, "Human-in-the-loop data integration," *Proc. VLDB Endow.*, vol. 10, no. 12, pp. 2006–2017, 2017. [Online]. Available: http://www.vldb.org/pvldb/vol10/p2006-li.pdf

[56] Y. Zhuang, G. Li, Z. Zhong, and J. Feng, "Hike: A hybrid human-machine method for entity alignment in large-scale knowledge bases," in *Proceedings of the 2017 ACM on Conference on Information and Knowledge Management, CIKM 2017, Singapore, November 06 - 10, 2017*. ACM, 2017, pp. 1917–1926. [Online]. Available: https://doi.org/10.1145/3132847.3132912

[57] A. Bonifati, R. Ciucanu, and S. Staworko, "Learning join queries from user examples," *ACM Trans. Database Syst.*, vol. 40, no. 4, pp. 24:1–24:38, 2016. [Online]. Available: https://doi.org/10.1145/2818637

[58] N. Vesdapunt, K. Bellare, and N. Dalvi, "Crowdsourcing algorithms for entity resolution," *PVLDB*, vol. 7, no. 12, pp. 1071–1082, 2014.

[59] D. Firmani, B. Saha, and D. Srivastava, "Online entity resolution using an oracle," *PVLDB*, vol. 9, no. 5, pp. 384–395, 2016.

[60] J. Wang, G. Li, T. Kraska, M. J. Franklin, and J. Feng, "Leveraging transitive relations for crowdsourced joins," in *Proceedings of the 2013 ACM SIGMOD International Conference on Management of Data*, 2013, pp. 229–240.

[61] C. Gokhale, S. Das, A. Doan, J. F. Naughton, N. Rampalli, J. Shavlik, and X. Zhu, "Corleone: Hands-off crowdsourcing for entity matching," in *Proceedings of the 2014 ACM SIGMOD international conference on Management of data*, 2014, pp. 601–612.

[62] G. Li, "Human-in-the-loop data integration," *PVLDB*, vol. 10, no. 12, pp. 2006–2017, 2017.

[63] Y. Li, X. Wang, Z. Miao, and W. Tan, "Data augmentation for ml-driven data preparation and integration," *Proc. VLDB Endow.*, vol. 14, no. 12, pp. 3182–3185, 2021. [Online]. Available: http://www.vldb.org/pvldb/vol14/p3182-li.pdf

[64] A. Kumar, J. Naughton, J. M. Patel, and X. Zhu, "To join or not to join? thinking twice about joins before feature selection," in *Proceedings of the 2016 International Conference on Management of Data*, 2016, pp. 19–34.

[65] J. Liu, F. Zhu, C. Chai, Y. Luo, and N. Tang, "Automatic data acquisition for deep learning," *Proceedings of the VLDB Endowment*, vol. 14, no. 12, pp. 2739–2742, 2021.

[66] C. Chai, J. Liu, N. Tang, G. Li, and Y. Luo, "Selective data acquisition in the wild for model charging," *PVLDB*, vol. 15, no. 7, pp. 1466–1478, 2022.

[67] Z. Zhao and R. Castro Fernandez, "Leva: Boosting machine learning performance with relational embedding data augmentation," in *Proceedings of the 2022 International Conference on Management of Data*, 2022, pp. 1504–1517.

[68] H. Altwaijry, D. V. Kalashnikov, and S. Mehrotra, "Query-driven approach to entity resolution," *Proc. VLDB Endow.*, vol. 6, no. 14, pp. 1846–1857, 2013. [Online]. Available: http://www.vldb.org/pvldb/vol6/p1846-altwaijry.pdf

[69] Y. Altowim, D. V. Kalashnikov, and S. Mehrotra, "Progressive approach to relational entity resolution," *Proc. VLDB Endow.*, vol. 7, no. 11, pp. 999–1010, 2014. [Online]. Available: http://www.vldb.org/pvldb/vol7/p999-altowim.pdf