Efficiently Estimating Mutual Information Between Attributes Across Tables

Aécio Santos
New York University
aecio.santos@nyu.edu

Flip Korn
Google Research
flip@google.com

Juliana Freire
New York University
juliana.freire@nyu.edu

2024 IEEE 40th International Conference on Data Engineering (ICDE) | 979-8-3503-1715-2/24/$31.00 ©2024 IEEE | DOI: 10.1109/ICDE60146.2024.00022

***Abstract*—Relational data augmentation is a powerful technique for enhancing data analytics and improving machine learning models by incorporating columns from external datasets. However, it is challenging to efficiently discover relevant external tables to join with a given input table. Existing approaches rely on data discovery systems to identify "joinable" tables from external sources, typically based on overlap or containment. However, the sheer number of tables obtained from these systems results in irrelevant joins that need to be performed; this can be computationally expensive or even infeasible in practice. We address this limitation by proposing the use of efficient mutual information (MI) estimation for finding relevant joinable tables. We introduce a new sketching method that enables efficient evaluation of relationship discovery queries by estimating MI without materializing the joins and returning a smaller set of tables that are more likely to be relevant. We also demonstrate the effectiveness of our approach at approximating MI in extensive experiments using synthetic and real-world datasets.**

***Index Terms*—data discovery, mutual information estimation**

I. Introduction

Our increasing ability to collect and store data has led to an explosion of data repositories, both for open [1]–[4] and enterprise data [5]–[7]. This abundance creates opportunities to enhance data analytics and machine learning models: by incorporating columns from various external datasets, we can explain confounding bias [8], test hypotheses and explain salient features in data [9]–[11], as well as improve predictive models [12], [13]. Consider the following example.

Example 1 (Understanding Taxi Demand)**.** *A data scientist wants to* improve a regression model for predicting taxi demand *that was constructed using historical data containing pick-up times and ZIP Codes where the pick-up occurred (see table $\mathcal{T}_{taxi}$ in Figure 1(a)). Here demand is measured by the total number of taxi rides (*`NumTrips`*) originating from the same spatio-temporal region (ZIP Code and Time). Since weather is known to impact taxi demand, the data scientist obtains a new table $\mathcal{T}_{weather}$ (Figure 1(b)) that contains information about temperature and precipitation. By augmenting the taxi trips table $\mathcal{T}_{taxi}$ (through a join on* `Date`*) with hourly average temperature and hourly rainfall as additional features, the mean absolute error of a random forest regressor improves significantly. In an effort to identify additional factors that may help* explain the demand variability in different neighborhoods*, the data scientist joins $\mathcal{T}_{taxi}$ with $\mathcal{T}_{demographics}$ (on* `ZipCode`*) containing demographics statistics (Figure 1(c)). The association between demand and population for the ZIP Codes of pick-up locations suggests a strong dependency.* □

Relational Data Augmentation. The manual process users must go through to discover external tables is time-consuming. In an attempt to automate this process, approaches have been proposed for *relational data augmentation* [12], [14]–[20]. Most of the existing work focuses on efficient support for data integration queries, including algorithms to discover "joinable" tables [14]–[17], [20]–[22], typically based on containment or overlap. These have an important limitation in that they can return too many irrelevant tables. Consider our example: if we search the NYC Open Data repository [1] to discover additional features that can help improve taxi demand prediction, we will find a large number of datasets that overlap in time with the taxi data, i.e., that are joinable with $\mathcal{T}_{taxi}$ on `Date`. Testing each of these by performing the join, re-training, and testing the model is wasteful and can be too expensive.

Automatic relational data augmentation systems [12], [23]–[25] attempt to address this problem by applying feature selection techniques [26]: given a list of joinable tables, they materialize the joins and automatically select attributes that increase the accuracy of predictive model given as input. Because they rely on data discovery systems [13], [16] to return the joinable tables, they can end up with both a large number of joins to perform and, consequently, too many candidate features to consider. Given the cost of evaluating the joins and the computational complexity of feature selection methods, it is expensive to identify the useful features.

To reduce the number of irrelevant candidate features, recent methods were proposed to discover tables that are not only joinable with a query table but also have attributes strongly correlated with an input target variable [27]–[29]. Instead of joining all table pairs, these methods attain scalability by relying on inverted indexes and sketching algorithms. While indexes allow efficient identification of tables that have join attributes with overlapping values, the sketches reduce the cost of joins and correlation computation by enabling efficient estimation of correlations between the input target variable and the top-k discovered candidate attributes. In essence, these methods allow performing feature pruning without materializing joins, and return tables (along with their corresponding features) that are more likely to enhance model performance or provide explanations for a target variable of interest.

2375-026X/24/$31.00 ©2024 IEEE
DOI 10.1109/ICDE60146.2024.00022

Date	ZipCode	NumTrips
2017-01-01	11201	136
...	...	...
2017-01-02	10011	112

(a) Daily taxi trips ($\mathcal{T}_{taxi}$).

Date	Time	Temp	UVIndex	Wind	Rainfall
2017-01-01	14:00:00	44.1	2	12mph	0.21
...	...	...	...	...	...
2017-01-01	16:00:00	42.0	1	15mph	0.25

(b) Hourly weather indicators ($\mathcal{T}_{weather}$).

ZipCode	Borough	Population	Income
11201	Brooklyn	53,041	[89k to 170k]
...	...	...	...
10011	Manhattan	50,594	[89k to 170k]

(c) Demographic statistics by ZIP code ($\mathcal{T}_{demographics}$).

Date	ZipCode	AVG[Temp]	AVG[Rainfall]	Borough	Population	NumTrips
2017-01-01	11201	43.0	0.19	Brooklyn	53,041	136
...	...	...	...	...	...	...
2017-01-02	10011	44.1	0.11	Manhattan	50,594	112

(d) Result of the augmentation of table (a) with features derived from attributes of tables (b) and (c).

Fig. 1: Example of relational data augmentation for the problem of taxi demand prediction. Adding new features, such as `AVG[Temp]` and `AVG[Rainfall]`, derived from external tables helps predict, or explain the variance of, the `NumTrips` attribute. The augmented table (d) is derived by joining $\mathcal{T}_{taxi}$ and $\mathcal{T}_{weather}$ on `Date`, and with $\mathcal{T}_{demographics}$ on `ZipCode`.

Unfortunately, these sketches have limitations. Notably, they do not properly handle repeated values on join keys (that are common in real data), which may cause estimation issues when performing left joins (especially on skewed distributions). Furthermore, correlation measures may fail to identify non-monotonic relationships and they are only applicable to numerical attributes. For example, while Pearson's correlation may be used to identify the relationship between the numerical attributes `NumTrips` and `Rainfall`, it cannot be directly applied to `Borough`, which is a categorical attribute. It also can fail to identify the non-monotonic relationship between `NumTrips` and `Population`, assuming taxis have fewer pick-ups both in neighborhoods with small populations (due to fewer customers) and large populations (due to heavy traffic).

Beyond Correlation-Based Data Discovery. Ideally, we would like to use a more general measure of statistical dependence, such as *Mutual Information (MI)*, which is invariant under homomorphism. Because MI is defined over probability distributions (for scalars and vectors), it is applicable to multiple data types and dimensionalities. Due to its generality, MI has found applications in numerous problems including the analysis of gene expression [30], functional dependency discovery [31]–[34], explanatory data analysis [35], causality detection [36]. In machine learning, MI is also used in many feature selection methods [26], [37], [38]. Moreover, strong theoretical connections between MI and model generalization error have been found showing that regression and classification errors are minimized when features having the largest conditional MI with the target are selected [39]–[41].

Challenges of MI Estimators. Estimating MI from finite data samples is far from trivial. Commonly used MI estimators based on empirical entropy do a poor job of modeling the underlying distribution due not only to small sample sizes but also to inherent estimator bias [42] (see Section II).

Moreover, while MI is well-defined for different data types, existing estimators typically either handle only discrete-categorical or continuous-numerical data. A common approach to handle continuous data is to discretize it using binning techniques [43] and then use an estimator for discrete data. Unfortunately, binning makes strong assumptions about the data distribution, may lead to information loss, and has bias that increases with the number of distinct values for the maximum likelihood estimator (MLE) on discretized data [44].

Instead of binning numerical data, some implementations [45] employ different estimators depending on the attribute data type. Specifically, they use the MLE estimator (i.e., maximum likelihood estimator plug-in of the empirical distribution [46]) for categorical/discrete data and the KSG estimator for numeric [47] or mixed numeric-categorical [48] values. While this approach avoids the pitfalls of data transformation techniques, it is not clear if using this combination of estimators with different properties (e.g., bias) is more effective.

MI Estimation over Joins. In the particular setting of relational data augmentation, these issues are compounded by additional problems created by many-to-one left joins and the need for efficiency. In this scenario, the goal is to augment a given base table with additional features that will be used to train a machine learning model to better predict or explain a target variable. Therefore, we need to keep the number of rows in the original table intact via a left-outer join. However, this creates some issues.

First, joining tables on non-unique join-key attributes leads to the creation of feature attributes containing repeated values that follow the distribution of the join key. For instance, while the original `Population` attribute in Figure 1(c) may have unique values, the derived `Population` attribute in the augmented table shown in Figure 1(d) will have additional repeated entries according the distribution of the `ZipCode` join key. In the particular case where the external attribute has a continuous distribution, the derived feature attribute will be a mixture of continuous distributions (with repeated values) that need to be handled properly by specialized MI estimators [49].

Second, to avoid the cost of fully materializing joins, systems are limited to estimating MI using a small number of samples of the join obtained using sketches [27], [50]. State-of-the-art methods typically use coordinated sampling (based on minwise hashing) to increase the number of samples that contribute to the join [51]. However, given that coordination is typically achieved by hashing values of the join-key attributes, these algorithms may introduce sampling bias and dependence

on the join-key that leads to violating the i.i.d. assumptions of estimators. When this happens, the bias of MI estimators is further increased (as shown in Sections IV-B and V).

Finally, existing join-sampling algorithms [50], [52], [53] are not directly applicable as they use estimators that are specialized for the specific function they aim to estimate, typically `COUNT` or `SUM`. Moreover, they have multiple sampling rate parameters that are difficult to set in practice, and do not provide a bound on sketch size.

Contributions. We define the problem of MI estimation over joins for data augmentation, and identify challenges that arise when estimating it over left joins with non-unique keys. We propose a new sketch that has a single parameter, the maximum sketch size, and addresses MI estimation challenges while avoiding the full join computation, thus providing efficient support of MI-based data discovery. Unlike previous approaches, our sketching method: does not assume that join keys are unique, and guarantees a fixed-size sketch while keeping an unbiased uniform sample of the join and thus can be used with any existing sample-based MI estimator.

We assess the effectiveness of our sketching method and different MI estimators through an extensive experimental evaluation. To do so, we design a synthetic benchmark that allows us to compare the estimated and the true MI obtained analytically from the data distributions used to generate the data. This benchmark allows us to observe the impact of dependence between the join-key attributes and the feature attributes on the MI estimates computed using the sketches. Furthermore, it allows us to identify differences in the behavior of various combinations of sketches and MI estimators, when and why they fail. We also evaluate our sketches using real-world data from open-data repositories. Our results confirm that the proposed sketch enables efficient approximation of the MI computed on the full join, and uncovered useful observations that help guide the implementation and deployment of MI-based sketches for data augmentation.

II. Background

Before presenting our approach, we first present the terminology used in the paper and some information-theoretic measures such as entropy and mutual information.

Data Types. As a simplification, we shall use the terms **discrete** and **continuous** to distinguish types of value distributions, with the former reserved for what is referred to in the literature as (often unordered) categorical, and the latter as (ordered, often floating-point) numerical [54]. We are aware that real data is more complicated and may include integral categories (e.g., UPC code) and floating-point values that represent discrete categories (e.g., Dewey Decimal). We assume such cases are represented as strings in Section V.

It is also important to differentiate a single **mixture** attribute, that contains a mixture of continuous distributions (e.g., the variable `AVG[Temp]` from the weather table in Figure 1(d) has repeated values for each zip code on the same date), from a pair of attributes where each contains a different data type.

Entropy. *Entropy* quantifies the amount of "information" or "uncertainty" in the possible outcomes of a random variable. Let X be a discrete random variable that assumes values from $dom(X) = \{1, ..., u_X\}$ and has *probability mass function* $p(X)$. The entropy $H(X)$ of the random variable X is:

$$H(X) = \mathbb{E}[-\log p(X)] = -\sum_{i=1}^{u_X} p(i) \log p(i) \tag{1}$$

Analogously, when X is a continuous random variable whose support is defined over the set $\mathcal{X}$ and has probability density function $f(X)$, the *differential entropy* is defined as:

$$H(X) = \mathbb{E}[-\log f(X)] = -\int_{\mathcal{X}} f(x) \log f(x)\, dx \tag{2}$$

These measures can be generalized to multiple variables. Let X and Y be random variables, whose support values are the ranges $[1, u_X]$ and $[1, u_Y]$, respectively, with joint probability mass function $p(X, Y)$. We define the *joint entropy* as:

$$H(X, Y) = -\sum_{i=1}^{u_X} \sum_{j=1}^{u_Y} p(i, j) \log p(i, j) \tag{3}$$

Analogously, if X and Y are continuous with support on $\mathcal{X}$ and $\mathcal{Y}$, respectively, and joint probability density function $f(x, y)$, the *joint differential entropy* is defined as:

$$H(X, Y) = -\int_{\mathcal{X},\mathcal{Y}} f(x, y) \log f(x, y)\, dx\, dy \tag{4}$$

Mutual Information (MI). The MI between two variables X and Y quantifies the amount of information obtained about one variable by observing the other. It is defined as:

$$I(X, Y) \triangleq H(X) + H(Y) - H(X, Y) \tag{5}$$

Intuitively, MI represents the amount of information learned about the target variable Y after observing the feature X. It is widely used in applications including feature selection [39], data augmentation [24], and causality analysis [55]. In decision tree learning, it is known as *information gain*.

Note that $I(X, Y) \geq 0$, with $I(X, Y) = 0$ only when X and Y are independent variables. What makes MI particularly attractive is its robustness due to invariance under reparameterizations: any bijection on discrete values and any homomorphism on continuous values, including affine transformations, has the same MI [47].

Estimating Entropy and Mutual Information. The measures above are defined over probability distributions. However, the distribution is usually unknown in real-life applications. Therefore, applications often use an estimator based on a finite number of observations to approximate the distribution.

The classical *maximum likelihood estimator (MLE)* of entropy is obtained by estimating the probability mass function using frequencies as follows. Given attribute X, let N be the number of observations in X and N_i be the frequency of each element i in X (hence, $N = \sum_{i=1}^{u_X} N_i$). The empirical entropy is estimated as:

$$\hat{H}_{MLE}(X) = \sum_{i=1}^{u_X} \frac{N_i}{N} \log \frac{N_i}{N}.$$

The MLE estimator is known to be systematically biased downward from the true entropy, and the bias is influenced by the number of samples N and distinct values m_X, and the distribution of X [42].

The MLE estimator is only applicable to discrete variables. To estimate entropy over values from a continuous domain X, one can estimate $H(X)$ from the average distance to the k-nearest neighbor (k-NN), averaged over all $x_i \in X$. It is well known that $H(X) \approx \frac{1}{N-1}\sum_{i=1}^{N-1}\log(x_{i+1}-x_i) + \psi(1) - \psi(N)$, where ψ is the digamma function [47].

The *mutual information* (MI) can be obtained by estimating $H(X)$, $H(Y)$, and $H(X,Y)$ separately and calculating Equation 5. When both X and Y are discrete, estimates can be obtained using the MLE estimator. When both of them are continuous, MI can be obtained using the KSG estimator [47] which computes $I(X,Y)$ in a slightly different way to avoid compounding errors of the terms. Alternatively, if both components are mixtures of discrete-continuous distributions, the MixedKSG estimator [49] can be used. MixedKSG proceeds similarly to KSG but recovers the plug-in estimator in discrete regions of the distribution (if they exist). Finally, when components have different types of distributions (i.e., discrete-continuous or continuous-discrete cases), another variation of the KSG estimator can be used [48]: first, the k-NN distances are computed for each discrete value using only the continuous variable, then the cardinality among all continuous values for those distances is calculated.

These MI estimators have different biases. The MLE estimator (for the discrete-discrete case) has a bias proportional to the number of distinct values and sample size [42]:

$$I(X,Y) - \mathbb{E}[\hat{I}_{MLE}(X,I)] \approx \frac{m_X + m_Y - m_{XY} - 1}{2N} \quad (6)$$

The KSG estimators, on the other hand, have a bias that stems from uniformity assumptions on the density and depends on neighbor distances [47]. In Section V, we provide an experimental comparison of these estimators on multiple datasets.

III. MI Estimation For Data Augmentation

We are interested in estimating mutual information in the relational data augmentation setting: augment a given base table with new attributes from an external table through a join operation. Since these new attributes may be used as additional features to train a machine learning model to predict a target variable, we need to keep the number of rows in the original base table intact by performing a left-outer join.

A. Problem Statement

Let $\mathcal{T}_{train}$ denote the base table containing (1) a target variable Y that we want to predict or explain, and (2) an attribute K_Y (or set of attributes) that can be used as a join key in a relational equi-join operation. Let $\mathcal{T}_{aug}$ be an external table that contains (1) an attribute X (or a set of attributes) that can be used as a feature; and (2) an attribute K_X that can be used to join $\mathcal{T}_{aug}$ with the base table $\mathcal{T}_{train}$ on the attribute K_Y. This is formally defined below. For exposition, we assume the case where K_X, K_Y and X are all single attributes, and we discard any rows with `NULL` values resulting from $\mathcal{T}_{aug}$ not containing some key k in K_X that is present in K_Y. [1]

Definition. (MI Estimation over Joins) *Given two tables $\mathcal{T}_{train}$ and $\mathcal{T}_{aug}$, the goal is to estimate the mutual information between the attributes X and Y from the table constructed through a left-outer-join of the two tables, $\mathcal{T}_{train} \bowtie \mathcal{T}_{aug}$, on keys K_X and K_Y, without having to compute the join.*

Consider the example in Figure 1. Here, $\mathcal{T}_{train} = \mathcal{T}_{taxi}$ is a table containing the number of daily taxi trips in New York City (NYC). The attribute Y represents the number of taxi trips `NumTrips` that happened in a particular ZIP Code with $K_X = K_Y =$ `ZipCode`. We are interested in enriching $\mathcal{T}_{train}$ with new features that may help predict the taxi demand (number of trips) on a given day. $\mathcal{T}_{aug} = \mathcal{T}_{demographics}$ represents another table discovered in a different source. To determine if $\mathcal{T}_{aug}$ may be useful for our predictive task, we want to estimate the mutual information between the columns X and Y (e.g., `Borough` and `NumTrips`), obtained after the join between $\mathcal{T}_{train}$ and $\mathcal{T}_{aug}$, without computing the full join.

B. Joining Arbitrary Tables

Many-to-Many Joins. Our problem definition assumes that there is a many-to-one relationship between $\mathcal{T}_{train}$ and $\mathcal{T}_{aug}$, that is, each tuple from $\mathcal{T}_{train}$ joins with at most one tuple from $\mathcal{T}_{aug}$. This is required by applications such as model improvement, where the number of rows in the original training set must remain intact to avoid introducing bias. Furthermore, if new rows are added during the join, it is not clear which labels should be assigned to these rows. However, while searching for augmentations, we can find candidate tables $\mathcal{T}_{cand}$ that have a many-to-many relationship with $\mathcal{T}_{train}$. For example, when joining taxi trips $\mathcal{T}_{taxi}$ and weather $\mathcal{T}_{weather}$ on `Date`, since temperature `Temp` values are recorded at hourly intervals, there are multiple temperature readings associated with each `Date`.

In such cases, since the augmentation will lead to duplicate key values, we transform the candidates to ensure that the augmented table will have the same number of rows as $\mathcal{T}_{train}$. To do so, we define a *featurization* function `AGG` that derives the augmentation table $\mathcal{T}_{aug}$ from a candidate table $\mathcal{T}_{cand}$. Given a candidate table $\mathcal{T}_{cand}$ with key column K_Z and value column Z, and a featurization function `AGG`, the following join-aggregation query maps $\mathcal{T}_{cand}[K_Z, Z] \rightarrow \mathcal{T}_{aug}[K_X, X]$, generating an intermediate aggregate table $\mathcal{T}_{aug}$ which is then combined with $\mathcal{T}_{train}$. This can be expressed using the following SQL query:

```
SELECT T_train[K_Y], T_train[Y], T_aug[X]
FROM T_train
LEFT JOIN (
  SELECT K_Z AS K_X, AGG(Z) AS X from T_cand
  GROUP BY K_Z
) AS T_aug
ON T_train[K_Y] = T_aug[K_X];
```

[1] While we limit joins to having high containment, our method does not prevent using existing strategies for handling `NULL`s in MI estimation [56], [57]. Evaluating these approaches is beyond the scope of this paper.

Note that while the aggregation of $\mathcal{T}_{cand}$ can be performed separately as a preprocessing step, the materialization of $\mathcal{T}_{aug}$ is not required for MI estimation. As we will discuss in Section IV, sketches can be constructed directly from $\mathcal{T}_{cand}$, which avoids the cost of aggregating join keys that are not needed for estimating the MI.

Figure 1 illustrates an example where the attribute $Z =$ `Temp` in $\mathcal{T}_{cand} = \mathcal{T}_{weather}$ is transformed (i.e., the values associated with a given date averaged) and joined on `Date` with $\mathcal{T}_{train} = \mathcal{T}_{taxi}$, as shown in the resulting table in Figure 1(d). Note that the numbers of rows in the tables $\mathcal{T}_{train}$ and $\mathcal{T}_{cand}$ need not be equal and their join attributes may have repeated values that need to be mapped to a feature value. We give a more concrete example below.

Example 2. *Let K_Y and K_Z be the keys for the tables $\mathcal{T}_{train}$ and $\mathcal{T}_{cand}$, respectively. Let $\mathcal{T}_{train}[K_Y] = [a, a, b, c]$ and $\mathcal{T}_{cand}[K_Z] = [a, b, b, b, c, c, c]$ and $\mathcal{T}_{cand}[Z] = [1, 2, 2, 5, 0, 3, 3]$, respectively. The first step is to group values based on the key values, i.e., $\{a \rightarrow [1], b \rightarrow [2, 2, 5], c \rightarrow [0, 3, 3]\}$. Next, the aggregate function `AVG` generates an intermediate aggregate table T_{aug} with the mappings $[a \rightarrow 1, b \rightarrow 3, c \rightarrow 2]$. Joining T_{aug} with the training table T_{train} generates the column $X = [1, 1, 3, 2]$. Similarly, if we applied `MODE` (to return the most frequent value), the output would be $X = [1, 1, 2, 3]$, and `COUNT` would generate $X = [1, 1, 3, 3]$.*

From the example above, we can draw a few observations about featurization and its implications for MI estimation.

Data Distribution. The distribution of the feature $\mathcal{T}_{train}[X]$ depends on the function `AGG` and the join key $\mathcal{T}_{train}[K_Y]$. For instance, distribution parameters such as the mean of X will likely be different for functions such as `AVG` and `MAX`. Note also that repeated values in K_Y lead to repeated values in X, e.g., the value 1 is repeated twice in X because a repeats twice in K_Y. Finally, note that $\mathcal{T}_{train}[X]$ may be even independent of $\mathcal{T}_{cand}[Z]$ when using a function such as `COUNT`, in which case it only depends on the key frequency distribution of K_Y (assuming no `NULL` values exist in Y).

Choice of Aggregation Function. There are many choices for aggregate functions and some may be more appropriate for specific data types, e.g., `AVG` for ordered-continuous data versus `MODE` for unordered-discrete data. Furthermore, the data type of the function output depends on the aggregation function and the input data type (e.g., while `COUNT` always outputs a discrete number regardless of input type, `MODE` outputs the same type as the input data. Note also that the aggregation function is not limited to outputting scalar values. It could also return multidimensional vectors (such as embeddings of text values), in which case, any MI estimator that supports multidimensional variables such as KSG [47] and its derivatives could be used (see Section II). In practice, to support general datasets and information needs, it may be necessary to create multiple augmentation columns using different aggregation functions and then examine the results.

IV. MI Estimation using Sketches

The task of estimating quantities over join results without materializing the join is a long-standing problem in the literature [50], [58]. One way to do so is to use sampling. A naïve approach to obtain samples of an equi-join is to join rows sampled *independently* from the two tables via Bernoulli sampling. Unfortunately, this results in a quadratically smaller join size [58] which results in poor accuracy.

To address this issue, we use *coordinated sampling* [27], [29], [50]–[53], [59]–[61]. In this approach, a shared-seed random hashing function is applied to the join keys, and a small set of tuples with the minimum hash values is selected to be included in the sketch [50], [59]. This implies that if a row with join key k is chosen from table $\mathcal{T}_{train}$, then a corresponding row with the same key k in $\mathcal{T}_{aug}$ is more likely to be sampled. Hence, we essentially forgo sample independence for an increased join size. However, this may lead to samples that are not identically distributed, which increases estimator bias. While there exist weighting schemes such as Horvitz-Thompson for correcting this bias [62], [63], these estimators are tightly coupled with the measures being estimated and, thus, are not directly applicable to estimating MI (see Section II). We, therefore, focus on sampling schemes that allow us to use existing MI estimators.

Another challenge is how to deal with repeated values in the join key. Coordinated sampling, which chooses samples based on hash values of join keys, typically assumes that the keys are unique or, if not, can be aggregated [27], [52], [59]. As discussed in Section III, this does not hold in the relational data augmentation setting since the number of rows must be kept intact. To address this issue, some join sampling algorithms suggest an all-or-nothing approach that includes all entries associated with a selected join key [52]. However, this is known to increase the variance of estimators and results in unbounded size [53]. Recent approaches introduce multiple sampling-rate parameters to select a fraction of the repeated entries [50], [53]. While this improves variance, these parameters are hard to set in practice and the resulting size is sensitive to table size and the join key distribution.

We propose new sampling-based sketches for estimating MI over joins that address these challenges. We start by proposing an extension of existing two-level sampling schemes [50], [53] that takes only a single parameter as input and provides a hard bound on sketch size (Section IV-A). We refer to this baseline approach as LV2SK. We provide an analysis that shows that LV2SK leads to non-uniform sampling (hence, not identically distributed) that can increase the bias of MI estimators; this is confirmed experimentally in Section V-B3. To address this issue, we propose TUPSK (Section IV-B), a novel tuple-based sampling scheme that leads to uniform sampling probabilities, which better matches assumptions made by the MI estimators. As shown in Section V-B3, this reduces bias and leads to higher accuracy of MI approximation.

Approach Overview. We assume access to hash function h_u that maps input uniformly to the unit range $[0, 1]$. We also assume that inputs to h_u are integers. Otherwise, we transform

the input to integers using a collision-free hash function h that maps objects to integers before feeding them to h_u, i.e., $h_u(h(x))$ where x is the input data. In practice, it suffices to use pseudorandom functions. To implement h, we used the well-known 32-bit MurmurHash3 function. For h_u, we used Fibonacci hashing [64].

Our approach works as follows. Given tables $\mathcal{T}_X$ and $\mathcal{T}_Y$ with schemas $[K_X, X]$ and $[K_Y, Y]$ respectively, we first build small sketches $\mathcal{S}_X$ and $\mathcal{S}_Y$. The sketch $\mathcal{S}_X$ is composed of a set of tuples $\langle h(k), x_k \rangle$ where $h(k)$ is the hash of a join key value $k \in K_X$ and x_k is a value from X; the sketch $\mathcal{S}_Y$ is built analogously to $\mathcal{S}_X$. Sketches are typically built in an offline preprocessing stage. When it is time to estimate MI between attributes in two different tables, we merge their sketches to recover a useful sample of the join for estimating the mutual information. Specifically, given a pair of sketches $\mathcal{S}_X$ and $\mathcal{S}_Y$, we create a sketch $\mathcal{S}_{join}$ by performing a join between the sketches on their hashed keys $h(k)$, resulting in tuples $\langle h(k), x_k, y_k \rangle$. These tuples are a subset of the full table join $\mathcal{T}_{join}$. Finally, we apply a function $\mathcal{F}$ that uses the sample of paired values $\langle x_k, y_k \rangle$ in $\mathcal{S}_{join}$ to estimate the MI of X and Y, i.e., $\hat{I} = \mathcal{F}(\mathcal{S}_{join})$. The function $\mathcal{F}$ uses existing MI estimators such as the ones described in Section II.

Our sketching algorithms only differ in the strategy they use to select samples that are included in the sketch. As inputs, they are given tables $\mathcal{T}_{train}$ (left table) and $\mathcal{T}_{cand}$ (right table). Specifically, when sketching $\mathcal{T}_{train}$, it must sample values associated with repeated key values, whereas, for $\mathcal{T}_{cand}$, it must aggregate repeated values in order to create a sketch that represents $\mathcal{T}_{aug}$. In what follows, we provide a detailed description of these methods.

A. Baseline: Two-Level Sampling (LV2SK)

Our first sketching method works as follows. In the first level, it performs coordinated sampling based on (distinct) join keys to select the same set of keys from both tables, thus maximizing the expected join size between $\mathcal{T}_X$ and $\mathcal{T}_Y$. However, given that this does not provide a bound on the sketch size, it performs a second sampling step to cap the number of samples per key, limiting the final sketch size.

Building LV2SK Sketches. We choose the set of tuples $\langle h(k), x_k \rangle$ as follows. At the first sampling level, we select the n keys that have the minimum values of $h_u(k)$. For each of these n keys, we filter a subset of the tuples having key k using independent Bernoulli sampling. The number of tuples to be included in the sketch is chosen in proportion to the frequency of k in the original table $\mathcal{T}$, as follows:

1) For $\mathcal{T}_{cand}$, we apply aggregate function `AGG` to the set of values $\{x_k\}$ associated with each key k to generate a single value $\texttt{AGG}(\{x_k\})$.
2) For $\mathcal{T}_{train}$, we keep $n_k = \max(1, \lfloor np_k \rfloor)$ samples per key, where $p_k = N_k/N$ is the probability of the key k in K_X.

The sampling strategy above guarantees that (1) the sketch contains at least one sample for each of the n chosen keys, and (2) for the chosen keys, the frequency of the key in the sketch is proportional to the frequency of the key in $\mathcal{T}_{train}$. We can build the sketch described above after sorting $\mathcal{T}_{train}$. Alternatively, it can be done in a single pass using reservoir sampling: we only need to maintain a reservoir with n samples for each of the n minimum keys and the number of repeated entries associated with each of the minimum keys, which is needed to determine the desired number samples n_k for each join key [65]. At the end of this pass, we keep only the first n_k samples of the reservoir of each key k and discard the remaining entries.

Sketch Size. The size of LV2SK sketches is upper bounded by $2n$, but it is typically close to n; To see this, note that the sketch size is given by: $\sum_{k_i \in \text{KMV}(K_X)} n_i = \sum_{k_i \in \text{KMV}(K_X)} \max\left(1, \lfloor \frac{nN_i}{N} \rfloor\right)$ where $k_i \in \text{KMV}(K_X)$ denotes the set of n minimum values in K_X selected in the first-level sampling. It is easy to show that the upper bound for its size is $2n$, and $\sum_{k_i \in \text{KMV}(K_X)} n_i \geq n$ holds whenever the number of unique values in the join key $m_{K_X} \geq n$.

Analysis. Let $p_i = \Pr[t_i]$ be the probability of selection of one tuple t_i in the first sampling level, and $q_i = \Pr[t_i]$ be the selection probability of t_i in the second-level. Since we assume a uniform hashing function h_u, in the first level any join-key value $t_i[K]$ has a uniform inclusion probability $p_i = 1/m_K$, where m_K is the number of distinct values in K. In the second level, where we perform Bernoulli sampling of N_i tuples that were sampled in the first level, we have that $q_i = 1/\max\left(1, \lfloor n\frac{N_i}{N} \rfloor\right)$. Given that the inclusion in the second level is independent of the first, the final probability of selecting the tuple t_i is $\Pr[t_i] = p_i q_i = 1/\left(m_K \cdot \max\left(1, \lfloor n\frac{N_i}{N} \rfloor\right)\right)$. Here we can see that the tuple selection probability is clearly dependent on the frequency distribution of the join-key values. Note, however, that in the special case where join keys are unique, i.e., $m_K = N$, the sampling probability becomes uniform as $\Pr[t_i] = 1/\left(N \cdot \max\left(1, \lfloor n\frac{1}{N} \rfloor\right)\right) = 1/(N \cdot 1) = 1/N$. An example of this arises when creating the sketch $\mathcal{S}_{aug}$, as it always aggregates the keys to a single value.

B. Proposed Approach: Tuple-based Sampling (TUPSK)

LV2SK sketches have important limitations. When a join key k is not selected for inclusion in the sketch in the first-level sampling, none of the rows that contain k will be included in the sample. Moreover, the sampling of a join key value does not take into account its frequency in the table. To see why this is a problem, consider the following extreme example.

Assume that we have a table $\mathcal{T}_{train}[K_Y, Y]$ of size $N = 100$. Let $K_Y = [a, b, c, d, e, f, f, f, ..., f]$ and $Y = [0, 0, 0, 0, 0, 1, 2, 3, ..., 95]$. Given that $\Pr[Y = 0] = 0.05$ and $\Pr[Y = i] = 0.01$ when $i \neq 0$, we have that the entropy of Y is $\hat{H}(Y) = -0.05 \log(0.05) - 95 \times 0.01 \log(0.01) \approx 4.5247$. Now consider a LV2SK sketch of size $n = 5$. In the case that the keys a, b, c, d, e are selected in the first-level sample, then $\mathcal{S}_{train}[Y] = [0, 0, 0, 0, 0]$, and thus its entropy estimate $\hat{H}(Y) = -1 \log 1 = 0$. Given that the entropy upper-bounds MI, we also have that the MI estimate between $\mathcal{S}_{train}[Y]$ and any feature column X, regardless of what its values are, must also be 0. Additionally, note that the probability

of selecting a tuple t_i such that the key value is equal to f, $\Pr[t_i[K_Y] = f]$, is $1/(6\max(1, 4.75)) = 0.035$ (since $n\frac{N_i}{N} = 5\frac{95}{100} = 4.75$), whereas for each of the other key values is $1/(6\max(1, 0.05)) \approx 0.167$. This example illustrates how dependence between the join key and the target attribute can lead to estimation bias.

To address these problems, we propose a coordinated sampling scheme that (1) considers individual rows as a sampling frame (i.e., each row is considered for sampling individually) and (2) leads to identically distributed samples (i.e., each row has the same probability of being sampled). Moreover, given that the probability of sampling each row is uniform, the expected number of sampled rows that contain a given join key k is proportional to the frequency of k in the original table. We refer to this method as TUPSK.

Building TUPSK Sketches. To build TUPSK sketches, we select rows from $\mathcal{T}_{train}$ by hashing keys as follows. We use the tuple $\langle k, j\rangle$ to identify the row where k appears for the j^{th} time in sequence, resulting in derived keys $\langle k, 1\rangle$, $\langle k, 2\rangle$, ..., $\langle k, N_k\rangle$. Then, instead of selecting the rows based on the minimum hash values of $h_u(k)$, we select tuples based on $h_u(\langle k, j\rangle)$. The final sketch, however, stores only tuples containing the hashed key and its associated value: $\langle h(k), x_k\rangle$. In other words, the tuples $\langle k, j\rangle$ are only used for deciding whether or not to include a row in the final sketch $\mathcal{S}_{train}$.

When sketching $\mathcal{T}_{cand}$, we handle repeated values as in LV2SK: we apply AGG to the set $\{x_k\}$, of values from X appearing with key k, to derive a single value $x_k = \texttt{AGG}(\{x_k\})$. Then we select tuples with the n minimum values of $h_u(\langle k, 1\rangle)$, since the aggregation results in unique keys. Hashing on $\langle k, 1\rangle$ provides sample coordination between the sketches $\mathcal{S}_{train}$ and $\mathcal{S}_{aug}$.

Analysis. Let p_i be the probability of selecting a tuple t_i to be included in a TUPSK sketch. It is easy to see that each $\langle k, j\rangle$ uniquely identifies a row in the table. Since h_u is a uniform hashing function, $p_i = 1/N$. Note that, unlike LV2SK sketches, the probability is uniform regardless of the frequency distribution of the join keys. Note also that, for the particular case of data augmentation, the tuples $\langle k, j\rangle$ also uniquely identify the rows in the final left join since the join is many-to-one and its output has the same size as the left table. Hence, the sample recovered by a sketch join is a uniform sample of the full join result.

It is worth noting that not all samples in the sketch are coordinated. This happens because the aggregation of tuples with repeated keys limits the domain of the tuples to $\langle k, 1\rangle$. In contrast, when sketching $\mathcal{T}_{train}$, the domain of the tuples $\langle k, j\rangle$ depends on the frequency distribution of the join key $\mathcal{T}_{train}[K_Y]$. This implies that the hashes of all tuples from $\mathcal{S}_{train}$ having $j > 1$ cannot match any tuples from $\mathcal{S}_{aug}$. Consequently, the sampling of such tuples is equivalent to a Bernoulli sampling.

Finally, note that unlike in LV2SK, each repeated key in $\mathcal{T}_{train}$ may evict other tuples from the n minimum values in $\mathcal{S}_{train}$. While somewhat counter-intuitive, less coordination means higher sample quality as this reduces dependence on the join keys (i.e., the sample becomes closer to an independent Bernoulli sample). Overall, our experimental results show that this scheme leads to a better trade-off between coordination and independence (Section V-B).

Accuracy Guarantees. TUPSK provides unbiased uniform samples of the join, but the actual estimation accuracy guarantees provided by our sketch also depend on the selected MI estimator used. All MI estimators used in this paper have been proven to be consistent estimators [44], [44], [49] under some assumptions, such as i.i.d samples. While TUPSK does not guarantee sample independence (due to coordination), our experiments (Section V-B1) show that the estimates converge to true MI when the sample size increases. Moreover, the high-probability error bounds for the empirical entropy and MI using the MLE estimator proposed in [66] (and subsequently improved in [67]) apply to sampling without replacement, which is similar to our setting. These bounds guarantee that the approximation error (i.e., the difference between an MI estimate computed on a subsample and the MI estimate computed on the full data) reduces in a near square root rate with respect to the subsample size (i.e., the sketch join size, in our case), and allow computing confidence intervals around the estimate that get tighter as the sketch join size approaches the full join size. While it is unclear if all assumptions in this bound hold for the samples generated by TUPSK, we have also observed this behavior in our experiments.

V. Experimental Evaluation

To evaluate the efficacy of our proposed sketching methods, we performed experiments using both synthetic and real-world data to answer the following questions: (Q1) How accurate are sketches at estimating the true mutual information of attributes obtained after the join? (Section V-B2); (Q2) How does join-key distribution affect the MI estimation accuracy? (Sections V-B3 and V-B4); (Q3) How does accuracy vary depending on target and feature data types and, thus, the applied MI estimator? (Section V-B); (Q4) How do sketches behave when estimating MI on real data collections? (Section V-C)

Mutual Information Estimators. Many MI estimators have been proposed. We consider a representative set of estimators that are widely used in practice. Unless otherwise noted, we choose estimators based on the data types of variables X and Y. When dealing with real data, we consider the following cases: (1) If both X and Y have string values (i.e., the discrete-discrete case), we use the maximum likelihood estimator (**MLE**); (2) If X and Y are numerical variables (e.g., float, integer), we consider the **MixedKSG** estimator [49]. This estimator is able to handle not only continuous distributions but also mixtures of discrete and continuous distributions in the same variable, making it flexible for dealing with real data where the distributions are unknown; (3) When one of the variables is numerical and the other is string (the discrete-continuous case), we use the estimator proposed by Ross in [48] that handles this case, referred to here as **DC-KSG**.

Sketching Methods. We evaluate LV2SK and TUPSK (Section IV). We also implemented another two-level approach that performs weighted sampling based on key frequencies (using priority sampling [68]) instead of the uniform sampling used in the first level of LV2SK, which we refer to as PRISK. Since its results are very similar to those of LV2SK, we omit them for the analysis using synthetic data. As baselines, we compare against independent Bernoulli sampling (INDSK) and a straightforward extension of Correlation Sketches (CSK) [27] that estimates MI instead of correlation measures. Since CSK does not prescribe how to handle repeated join keys, we use the first value seen associated with a join key (instead of applying an aggregation function that would modify the original values).

A. Synthetic Data Generation

To better understand the behavior of the proposed sketches and answer questions Q1, Q2, and Q3, we used synthetic data to control both the data distribution and join-key dependencies. We designed a data generation process to create tables given the join key distribution and true MI between X and Y post-join as input. This is achieved by generating the post-join target Y and feature X by drawing random values from analytic distributions as the join result. Then we decomposed the join into two separate tables, establishing connections using key (K_Y) and foreign-key (K_X) attributes. This approach allows us to calculate the true MI after the join, providing a reliable measure to evaluate the effectiveness of our method.

Target/Feature Generation. In the first experiment, we generated random variables (X, Y) using a multinomial distribution $Mult(m, \langle p_1, p_2 \rangle)$, which we refer to as `Trinomial`. This generates three discrete random variables that assume integer values in $\{1, ..., m\}$. Each value represents the number of times that each of the three possible outcomes has been observed in m trials, with each outcome having probabilities $p_1, p_2,$ and $(1 - p_1 - p_2)$. The number of trials m is chosen based on the desired number of distinct values in the data. We refer to the first two variables associated with p_1 and p_2 as X and Y, and the third variable is discarded.

To control the desired level of MI between X and Y, we use the following property of the trinomial distribution (see [69, chapter 11.1]). The central limit theorem ensures that $Mult(m, \langle p_1, p_2 \rangle)$ converges to a bivariate normal distribution $N(\mu, \sigma)$ with mean $\mu = \langle mp_1, mp_2 \rangle$ and variance $\sigma^2 = m\sqrt{p_i p_j}(\delta_{ij} - \sqrt{p_i p_j})$ as $m \to \infty$. Hence, solely for the purpose of selecting model parameters to achieve a desired MI, we can use the closed-form MI formula from the analogous bivariate normal distribution to approximate the MI for the trinomial, which is known to be $-\frac{1}{2}\ln\left(1 - r^2\right)$ where r is the Pearson's correlation coefficient of X and Y. Based on this and standard properties of the trinomial distribution, we derived the following algorithm to select the distribution parameters p_1 and p_2:

1) Choose the true mutual information $I_{true} \sim Unif(0, 3.5)$, then compute the equivalent correlation $r = \sqrt{1 - \exp\left(-2 \cdot I_{true}\right)}$; note that $I_{true} = 3.5$ is equivalent to $r \approx 0.999$.
2) Choose $p_1 \sim Unif(0.15, 0.85)$ (as the approximation works better when p is not near to 0 or 1).
3) Finally, calculate p_2 using the values of r and p_1 based on the trinomial variance and closed-form expression for correlation $r = -p_1 p_2 / \left(\sqrt{p_1(1-p_1)}\sqrt{p_2(1-p_2)}\right)$. If p_2 is not in the desired range (i.e., $[0.15, 0.85]$) then repeat.

The approximation using bivariate normal distribution above was only used to choose the parameters. To compute the true MI of the distribution, we used the (open-form) entropy formula for the trinomial distribution [70].

As done in [49], we also generated a combination of discrete and continuous data for X and Y, respectively, which we refer to as `CDUnif`. X follows a uniform distribution over the integers $\{0, 1, ..., m-1\}$, while Y is uniformly distributed within the range $[X, X+2]$ for a given X. The true mutual information between X and Y can be computed as $I(X, Y) = \log(m) - (m-1)\log(2)/m$. Note that here the MI is a function of parameter m, which also represents the number of distinct values in Y.

Distribution Parameters. For `Trinomial`, we restrict generated data to having MI $\in [0, 3.5]$ and $m \in \{16, 64, 256, 512, 1024\}$. Since both X and Y are discrete and have ordered numeric values, it is possible to treat the data as either discrete, mixture, or continuous. A marginal variable can be made continuous via perturbation, by breaking ties using random Gaussian noise of low magnitude without any significant impact on the MI [47]. Thus, by doing this in just one of the marginals we can use an estimator for discrete-continuous variable pairs such as the DC-KSG [48]. Additionally, MixedKSG [49] can also be applied to variables with repeated values as it can handle ties naturally based on its formulation. Hence, to evaluate the impact of these estimators, we consider three representative data type combinations: we use the MLE for discrete-discrete, MixedKSG [49] for mixture-mixture, and DC-KSG [48] for discrete-continuous.

For `CDUnif`, we draw m uniformly in the range $[2, 1000]$, which leads to MI values in the range $[0.3, 6.2]$. In this distribution, Y is continuous and X is discrete. Hence, we only report results using MixedKSG [49] and DC-KSG [48], which are able to deal with discrete and continuous distributions seamlessly without any data transformation.

Decomposition Into Joinable Tables. To decompose (X, Y) into tables $\mathcal{T}_{train}$ and $\mathcal{T}_{aug}$ that can be joined to include X and Y as columns, we employ two different methods of generating key and foreign-key columns: `KeyInd` for one-to-one joins and `KeyDep` for many-to-one joins (which allows us to answer Q2). `KeyInd` provides maximum independence between keys by generating (sequential) unique join keys in K_X, leading to a maximum number of different key values in K_X pointing to the same value in X (e.g., if the value x appears 10 times in X then we have 10 different values in K_X that co-occur with x). This method establishes a one-to-one relationship between attributes $K_Y \in \mathcal{T}_{train}$ and $K_X \in \mathcal{T}_{aug}$. `KeyDep` simulates a strong dependence of join keys by making the value in K_X, for each row, equal to the value in X. Hence,

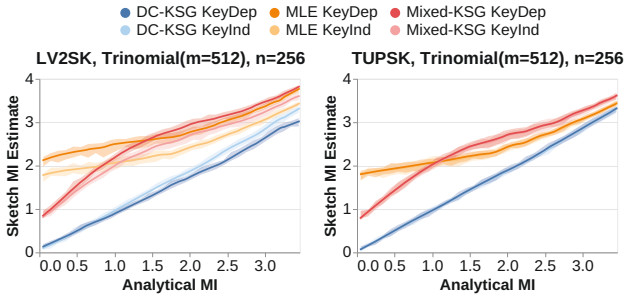

Fig. 2: True MI vs MI estimates computed using sketches of size $n = 256$. Each plot shows a different method (LV2SK on the left and TUPSK on the right) and each line shows results for different data types/estimators and join key generation processes. TUPSK is more robust to the join key distribution.

we have a single value in K_X for all the occurrences of a value in X, establishing a many-to-one relationship. Note that `KeyDep` is only applicable when X is discrete, as it would create unique join key values for continuous data distributions. Moreover, while the marginal distribution of X (and hence key frequencies in K_X) is uniform for `CDUnif`, it is a binomial distribution for `Trinomial`. Note also that although these methods represent two contrasting join scenarios, both methods enable table joins that exactly recover (X, Y).

B. Experiments Using Synthetic Data

1) True vs. Estimated MI on Full-Table Joins: Before presenting our sketch evaluations, we conducted a preliminary experiment to assess the behavior of different MI estimators. Our goal is to establish a baseline of the expected behavior of the MI estimators, especially when dealing with real data in Section V-C, where the true MI is unknown. For each data distribution, we consider all MI estimators that can be used with the given data types without applying any data transformations as described above: for `Trinomial` we used MLE, DC-KSG, and MixedKSG; for `CDUnif` we used DC-KSG and Mixed-KSG. We compare the true MI, calculated via the distribution parameters that were used to generate the data, to MI estimates obtained from the fully-materialized join containing $N = 10$k rows. For both `Trinomial` and `CDUnif`, the root mean squared error (RMSE) is smaller than 0.07, and the Pearson's correlation coefficient is greater than 0.99. We omit plots for these results since they are close to a straight line, as expected. The results demonstrate that MI estimates obtained from the full-table join provide a good approximation for the true MI (computed analytically), regardless of the data type assumptions made by each estimator. Although we can notice some small bias and variance in the range of lower true MI (especially for `Trinomial`), the overall error is very small in this setting with a large sample size.

2) Assessing Sketch Estimation Accuracy: Figure 2 shows the results for MI estimates for the `Trinomial` distribution ($m = 512$) computed using the proposed sketching methods, LV2SK and TUPSK. In this setting with limited sample size ($n = 256$), we see that both the bias and variance of the estimators increase significantly. Here, the MI is overestimated and the magnitude of the overestimation depends on the type of MI estimator being used: while the bias is highest for MLE estimator (●, ●) when the true MI is low, MixedKSG (●, ●) reaches a peak bias around the mid-range MI values.

While the bias and variance are also influenced by other factors (as discussed below), this result underscores the significance of selecting the appropriate estimator for the data type at hand. For example, while it may be simpler to use an MLE estimator with discrete ordered data (or, e.g., with binned continuous data), using a k-Nearest Neighbors approach may lead to a smaller bias.

Furthermore, this result suggests that comparing estimates of columns with different data types, which require distinct estimators, may not yield meaningful results due to the distinct bias and variance properties of different estimators. While the results of these estimators converge to the true MI when the sample size is large enough, as shown in Section V-B1, the approximation accuracy depends on many factors such as the data distribution, the sample size, and the underlying true MI. When these are unknown, it becomes challenging to determine whether such comparisons are meaningful. In Section V-C, we further discuss this issue based on our results on data sourced from real-world open data repositories.

3) Effect of the Join Key Distribution: In Section IV, we described two different methods to select samples to include in the sketch. In Figure 2, we can visualize how the join-key distribution affects the MI estimation accuracy of these methods. Specifically, we can compare the estimation difference caused by `KeyInd` vs `KeyDep` in a given estimator. For instance, consider the LV2SK method in Figure 2(left). When we compare the lines that represent the MLE estimator (●, ●), we note that the bias from `KeyDep` (●) is larger than that from `KeyInd` (●). Similarly, `KeyDep` leads to increased bias for the MixedKSG estimator (●) but, for the DC-KSG estimator, `KeyDep` leads to a small downward bias (●).

Differently from LV2SK, TUPSK is not as affected by the join-key distribution. We can see in Figure 2(right) that TUPSK is able to attain the same performance regardless of the join-key distribution. This is because the TUPSK sampling scheme reduces the dependence on the join keys by hashing on the tuple $\langle k, j \rangle$, which leads each row being sampled with uniform probability (given that each tuple $\langle k, j \rangle$ is unique in table $\mathcal{T}_{train}$). LV2SK on the other hand, samples entries non-uniformly and introduces additional bias due to its dependence on the key frequency distribution and the existing key-target correlation in the `KeyDep` distribution.

4) Effect of Distinct Values: To assess the impact of the number of distinct values in the distribution on the MI estimation accuracy, we vary the parameter m of `Trinomial` and `CDUnif` distributions while keeping the desired sketch size $n = 256$ constant. This means that the ratio m/n increases making it increasingly harder to estimate the MI, e.g., when $X \sim Uniform$ and $m/n = 1$ we expect to have only 1 sample to estimate the probability mass $p(x)$ of a given value $x \in X$. For `CDUnif`, $m = 256$ is equivalent

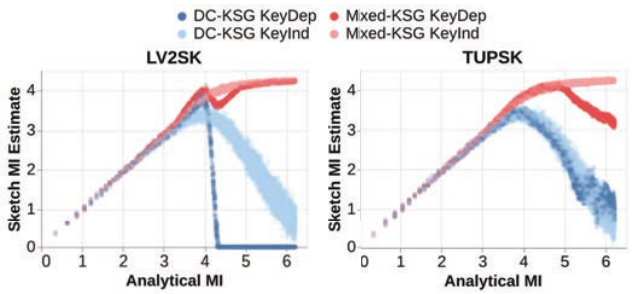

Fig. 3: True MI vs MI estimates computed using sketches of size $n = 256$ for `CDUnif`. Each plot shows a different sketching method while each line shows results for different data types/estimators and join key generation processes.

to $I(X,Y) \approx 4.85$. As Figure 3 shows, the MI estimators break down when $I(X,Y)$ approaches 4.85 for the `CDUnif` distribution. For LV2SK, the DC-KSG estimator completely breaks down even earlier, around $I(X,Y) \approx 4.25$. In contrast, TUPSK degrades more gracefully as $I(X,Y)$ increases.

Figure 4 shows the impact of increasing m for the `Trinomial` distribution. Here, the marginal distributions of X and Y are non-uniform (binomial) distributions (unlike `CDUnif` which has uniform marginals). The plots clearly show that increasing m leads to increased bias for estimators that handle discrete distributions such as MLE (●) and Mixed-KSG (●). Although the estimators do not completely break down here, we can see that the bias for the MLE estimator (●) is so large when $m = 1024$ that all estimates are considered to have a high MI in the small range $[2.5, 3.5]$.

Note that the maximum true MI value for low values of m is smaller than for large m. This is due to our data generation process (Section V-A), which relies on the central limit theorem and the bivariate normal distribution to approximate the MI. This, however, does not affect our results since we use the exact MI formula to compute the analytical MI in Fig. 4.

5) Comparison to Other Baselines: In Table I, we report the average sketch join size and mean squared error (MSE) for all sketches, including the additional baselines: independent sampling (INDSK) and correlation sketches (CSK). Results are computed for sketches of size $n = 256$ and include tables with different join key distributions (`KeyDep`, `KeyInd`) and different distribution parameters (m). The results demonstrate that INDSK has difficulty matching join keys, resulting in a smaller join size than coordinated sampling approaches, which leads to large MSE. Coordinated sampling methods achieve significantly larger join sizes, making them more effective strategies. Among them, TUPSK achieves the best MSE, which is due not only to the larger number of samples recovered by the sketch join but also to unbiased samples. The average join size of the two-level sampling sketches is highly sensitive to the join key distribution: when keys are unique (as in `KeyInd`), it behaves as TUPSK, and a sketch of size n yields n join samples. However, when there are repeated keys, it may lead to either more or fewer samples than n. In our experiments, the average join size increases as m increases.

Dataset	Sketch	Avg. Sketch Join Size	%	MSE
`CDUnif`	CSK	194.2	75.87	4.56
	INDSK	107.9	42.16	9.57
	LV2SK	232.9	90.99	2.94
	PRISK	232.9	90.99	2.94
	TUPSK	256.0	**100.00**	**0.77**
`Trinomial`	CSK	155.2	60.62	1.37
	INDSK	133.7	52.22	1.19
	LV2SK	255.9	99.94	0.32
	PRISK	255.9	99.94	0.32
	TUPSK	256.0	**100.00**	**0.22**

TABLE I: Comparison of MI estimate versus the true MI using sketches of size $n = 256$. The "%" column is the percentage of "Avg. Sketch Join Size" relative to sketch size n.

C. Experiments Using Real Data

We now evaluate the behavior of our sketches on real-world data collected from two different open-data portals: the World Bank's Finance (WBF) [71] and the NYC Open Data (NYC) [1]. Our experimental data consists of snapshots of these repositories collected in September 2019 using Socrata's REST API [72].

From these collections, we sample pairs of tables $\mathcal{T}_{train}$ and $\mathcal{T}_{aug}$ as follows. For each table t in a data repository, we first create the set $\mathcal{P}_t$ of two-column tables, denoted as $\mathcal{T}_A[K_A, A]$, comprised of all pairs of join-key and data attributes $\langle K_A, A\rangle$ from $\mathcal{P}_t$ such that K_A is a string attribute and A contains either strings or numbers (i.e., ints, longs, floats, or doubles).[2] Let $\mathcal{C} = \bigcup_t \mathcal{T}_t$ be the set of all two-column tables in the repository. We then draw a uniform sample of the set of pairwise combinations $\mathcal{PC} = \{(\mathcal{T}_i, \mathcal{T}_j) \mid \mathcal{T}_i, \mathcal{T}_j \in \mathcal{C}\}$ and use the tables in these pairs as $\mathcal{T}_{train}$ and $\mathcal{T}_{aug}$. The final sample includes 36k table pairs for the WBF collection and 59k pairs for the NYC collection. The average domain size of join attributes for the left and right tables are approximately 3.1k and 3.5k for WBF, respectively, and 11.2k and 1k for NYC, respectively. Finally, the average full join size is 34k for WBF and 8.5k for NYC.

Given that it is not possible to know the true distribution of the data in these tables, we use the MI estimated over the full data as a proxy for the true MI. As shown in Section V-B1, the full join provides a good approximation of the true MI when the join size is large. Hence, from now on we compare the sketch estimates to the full-join estimates. Even though the full join may not always reflect the true MI, it is the only option available in many practical scenarios [66].

1) Approximation Accuracy: Table II summarizes the results for each sketching method for the two dataset collections. The results are computed using sketches with $n = 1024$. To discard meaningless estimates, we only include estimates computed on sketch join size greater than 100. First, we confirm that LV2SK, which may use a higher storage size for a given budget n (see Section IV-A), tends to generate a larger average join size. However, despite using less storage,

[2]We used the Tablesaw library [73] to perform type inference.

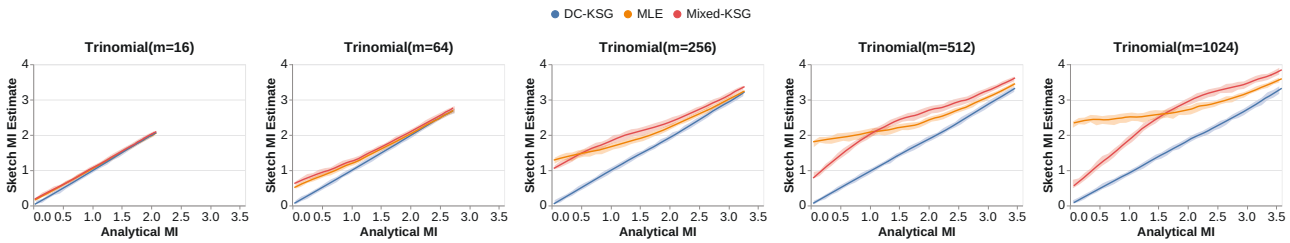

Fig. 4: Sketch MI estimate versus the true MI computed using distribution parameters. Sketch size is $n = 256$ for all plots.

Dataset	Sketch	Avg. Join Size	Spearman's R	MSE
NYC	LV2SK	230.9	0.81	1.41
	PRISK	231.1	0.79	1.36
	TUPSK	185.3	**0.86**	**0.93**
WBF	LV2SK	231.2	0.40	1.75
	PRISK	226.6	0.40	1.76
	TUPSK	194.9	**0.45**	**1.46**

TABLE II: Comparison of MI estimate using different sketching strategies versus the full join. While LV2SK can theoretically have a sketch size twice as large as TUPSK, in practice their sketch join sizes is similar. Even with this disadvantage, TUPSK outperforms LV2SK in estimation accuracy (stronger Spearman's R correlation) using less storage.

TUPSK outperforms LV2SK in terms of estimation accuracy measured by the mean squared error (MSE) metric.

We use Spearman's correlation, a rank-based measure, to quantify how well the ranking obtained using MI estimates computed from sketches approximates the ranking of MI estimates computed over the full tables. We can see that Spearman's correlation for TUPSK is the strongest. This is significant since for automatic data augmentation it is important to rank features based on their importance. This result confirms that TUPSK is able to generate higher quality samples than its competitors.

2) Effect of Sketch Join Size: In Figure 5, we break down the results by data types (and hence, MI estimators) and sketch join size. A larger sketch join size indicates that the tables are more joinable (i.e., have a larger overlap) and that the MI estimator is given a larger number of samples. Here, we can observe a behavior similar to what we observed with synthetic data. In particular, we note that when the sample size is small, (1) the MLE estimator (●) tends to overestimate the MI, and (2) the KSG-type estimators (●, ●) tend to break down and generate estimates close to zero.

3) Comparing MI Estimators: Another notable difference is the magnitude of the MI estimates generated by different estimators: the MLE estimator computes MI values that are significantly larger than the ones generated by KSG-based estimators: while MLE estimates reach the range $[4, 6]$, KSG-based estimates are never larger than 2. Although we cannot confirm whether this is an artifact of the estimator limitations or if numerical data indeed leads to smaller MI values, this result suggests that comparing MI estimates from different estimators may not be reasonable. For example, when ranking attributes for data discovery, it might be preferable to produce separate rankings of different MI estimators and then compare the utility of top-ranked attributes using a downstream (task-specific) evaluation measure (e.g., the increase in accuracy of an ML model computed on the labels).

D. Performance Evaluation

Due to space constraints, we omit a detailed runtime evaluation since the efficiency of the proposed sketches is similar to others evaluated in previous work [27], [29]. For completeness, we provide exemplar numbers for sketch size $n = 256$. As the table size grows from $N = 5k$ to $N = 20k$, the full join size time increases from 0.35ms to 2.1ms, whereas the sketch join time grows from 0.03ms to 0.18ms. Similarly, while MI estimation time increases from 2.2ms to 10.7ms, the sketch is approximately constant and took only 0.1ms.

VI. Related Work

To the best of our knowledge, no prior work has addressed the problem of estimating mutual information (MI) over joins for relational data augmentation using sketches. In what follows, we examine related prior research on data discovery systems, feature selection, MI estimation, and join sampling.

Data Discovery. The problem of finding related datasets (via unions and joins) on the Web and in data lakes, to unlock the utility of a provided table has been studied since [74]. Applications include decision support, data mining, ML model improvement, and causality analysis, and have resulted in systems including Aurum [16], ARDA [12] and Auctus [13]. Some related work focuses on methods that utilize sketches and Locality-Sensitive Hashing (LSH) indexes to efficiently discover joinable tables [14], [15]. Others address the problem of finding tables that are both joinable with the input query table and contain correlated columns, employing different techniques to identify these relationships [22], [27], [29], [75], [76]. MI has been used for discovering functional dependencies (FDs), often between columns within the same table [32]–[34]. FDs are not symmetric, unlike MI, so while MI can help discover FDs, the reverse is not always true. Finally, our method is complementary to the work of Kumar et. al. [75], where they propose conservative decision rules to predict when the features obtained through a join can improve models.

Feature Selection. There are three main classes of feature selection methods (see [26] for a survey): filter methods, which start from a join over all tables and use a lightweight proxy

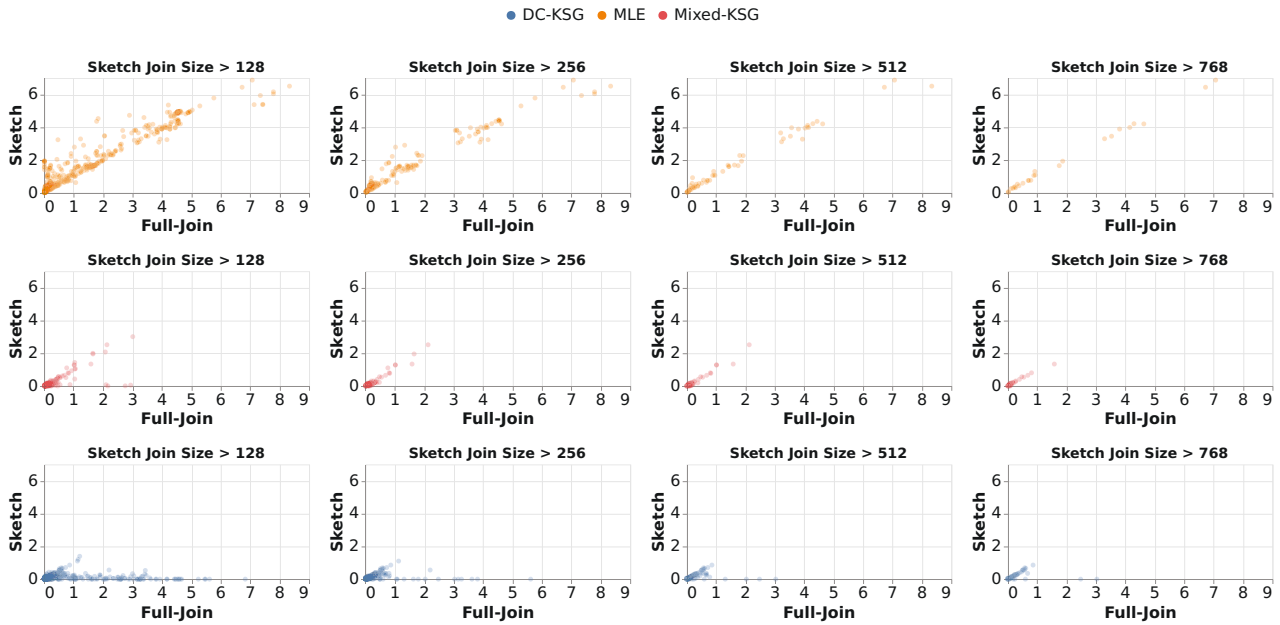

Fig. 5: Sketch MI estimate versus the MI estimate computed using the full join output for tables from the WBF collection. Sketches are created using TUPSK with size $n = 1024$ for all plots.

such as correlation to remove features; wrapper methods, which incrementally choose features based on the (more expensive) downstream task either by joining one table/feature at a time (forward selection) or removing one at a time from a join over all tables; and embedding methods, which use a proxy to select multiple features at a time and then measure using the downstream task. This work is related to filter methods since it enables the estimation of MI, which is a proxy measure commonly used in several feature selection algorithms [26], [41].

Mutual Information Estimation. When the number of available observations is large, computing the MI can be expensive. Various papers have considered efficient approximation algorithms, with confidence intervals, for entropy estimation via subsampling [66], [67], which can be extended to MI. Some approaches considered MI estimation over data streams [77]–[79]. Ferdosi et. al. [80] show how to approximately find a pair of columns having the largest mutual information in sub-quadratic time, however, they assume binary-valued attributes and that table joins are materialized. While our prior work on Correlation Sketches [27] examined the problem of sketches to avoid a full join, we are not aware of any work addressing the problem of estimating MI while avoiding the cost of materializing the entire join. There is also an extensive body of research on different estimators for MI, some of which we cover in Section II. Additionally, recent work has shown that no estimator can *guarantee* an accurate estimate of mutual information without making strong assumptions on the population distribution [81]. While previous work had provided intractability results for specific estimators, such as KSG [82], this result is universal to all MI estimators.

Join Sampling. The problem of creating samples of table join results has been extensively studied [50], [52], [53], [63]. We use ideas similar to the ones employed by these algorithms, such as sample coordination and two-level sampling. Unlike our sketches, however, these algorithms lead to variable sample sizes that depend on the input size and data distribution. Moreover, they usually employ unequal probability sampling that requires adjusting the estimators to remove bias. In contrast, we seek sampling strategies that lead to uniform probabilities that allow us to use existing MI estimators.

VII. Conclusion

In this paper, we introduced a new method for estimating mutual information (MI) using sketches. Our proposed method addresses the challenges of estimating MI over joins with non-unique keys and efficiently approximates the MI without the need to materialize the full join. Our experiments demonstrate the effectiveness of our sketches in approximating the true MI values. Additionally, we have shown how data types and estimators impact the accuracy of MI estimates, emphasizing the need for careful consideration when comparing MI estimates from different data types. Our results show that some estimators underestimate while others overestimate, there is thus a need to carefully choose an appropriate estimator based on application requirements. E.g., while MLE may offer high recall, estimators based on Laplace smoothing [34] may be more appropriate for controlling false discoveries. Exploring this trade-off further is a promising avenue for future work.

Acknowledgments. This work was partially supported by the DARPA D3M program, NSF award ISS-2106888, and a Google Research Collabs grant. Any opinions expressed in this material are those of the authors and do not necessarily reflect the views of the funding organizations.

REFERENCES

[1] "NYC OpenData," https://opendata.cityofnewyork.us.

[2] "City of Chicago Data Portal," https://data.cityofchicago.org.

[3] "United States Government Open Data," https://www.data.gov.

[4] D. Brickley, M. Burgess, and N. Noy, "Google dataset search: Building a search engine for datasets in an open web ecosystem," in *The World Wide Web Conference*, ser. WWW '19. New York, NY, USA: ACM, 2019, pp. 1365–1375. [Online]. Available: http://doi.acm.org/10.1145/3308558.3313685

[5] S. Bapat, "Discover, understand and manage your data with Data Catalog, now GA," https://cloud.google.com/blog/products/data-analytics/data-catalog-metadata-management-now-generally-available, 2020, [Online; accessed 22-June-2020].

[6] M. Grover, "Amundsen — Lyft's data discovery & metadata engine," https://eng.lyft.com/amundsen-lyfts-data-discovery-metadata-engine-62d27254fbb9, 2019, [Online; accessed 20-October-2019].

[7] M. Lan, "DataHub: A generalized metadata search & discovery tool," https://engineering.linkedin.com/blog/2019/data-hub, 2019, [Online; accessed 22-June-2020].

[8] B. Youngmann, M. Cafarella, Y. Moskovitch, and B. Salimi, "Nexus: On explaining confounding bias," in *Companion of the 2023 International Conference on Management of Data*, 2023, pp. 171–174.

[9] F. Chirigati, H. Doraiswamy, T. Damoulas, and J. Freire, "Data polygamy: the many-many relationships among urban spatio-temporal data sets," in *ACM SIGMOD*, 2016, pp. 1011–1025.

[10] A. Bessa, J. Freire, T. Dasu, and D. Srivastava, "Effective discovery of meaningful outlier relationships," *ACM Transactions on Data Science*, vol. 1, no. 2, pp. 1–33, 2020.

[11] A. Bessa, S. Castelo, R. Rampin, A. S. R. Santos, M. Shoemate, V. D'Orazio, and J. Freire, "An ecosystem of applications for modeling political violence," in *ACM SIGMOD*, 2021, pp. 2384–2388.

[12] N. Chepurko, R. Marcus, E. Zgraggen, R. C. Fernandez, T. Kraska, and D. Karger, "Arda: Automatic relational data augmentation for machine learning," *Proceedings of the VLDB Endowment*, vol. 13, no. 9, 2020.

[13] S. Castelo, R. Rampin, A. Santos, A. Bessa, F. Chirigati, and J. Freire, "Auctus: A dataset search engine for data discovery and augmentation," *Proceedings of the VLDB Endowment*, vol. 14, no. 12, pp. 2791–2794, 2021.

[14] E. Zhu, F. Nargesian, K. Q. Pu, and R. J. Miller, "Lsh ensemble: Internet-scale domain search," *Proc. VLDB Endow.*, vol. 9, no. 12, p. 1185–1196, Aug. 2016. [Online]. Available: https://doi.org/10.14778/2994509.2994534

[15] R. Castro Fernandez, J. Min, D. Nava, and S. Madden, "Lazo: A cardinality-based method for coupled estimation of jaccard similarity and containment," in *2019 IEEE 35th International Conference on Data Engineering (ICDE)*, April 2019, pp. 1190–1201.

[16] R. C. Fernandez, Z. Abedjan, F. Koko, G. Yuan, S. Madden, and M. Stonebraker, "Aurum: A Data Discovery System," in *ICDE '18*, 2018, pp. 1001–1012.

[17] 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: ACM, 2019, pp. 847–864. [Online]. Available: http://doi.acm.org/10.1145/3299869.3300065

[18] 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.

[19] Y. Dong and M. Oyamada, "Table enrichment system for machine learning," in *Proceedings of the 45th International ACM SIGIR Conference on Research and Development in Information Retrieval*, 2022, pp. 3267–3271.

[20] A. D. Nobari and D. Rafiei, "Efficiently transforming tables for joinability," 2022.

[21] Y. Yang, Y. Zhang, W. Zhang, and Z. Huang, "Gb-kmv: An augmented kmv sketch for approximate containment similarity search," in *2019 IEEE 35th International Conference on Data Engineering (ICDE)*, April 2019, pp. 458–469.

[22] M. Esmailoghli, J.-A. Quiané-Ruiz, and Z. Abedjan, "Mate: multi-attribute table extraction," *Proceedings of the VLDB Endowment*, vol. 15, no. 8, pp. 1684–1696, 2022.

[23] A. Ionescu, R. Hai, M. Fragkoulis, and A. Katsifodimos, "Join path-based data augmentation for decision trees," in *2022 IEEE 38th International Conference on Data Engineering Workshops (ICDEW)*, 2022, pp. 84–88.

[24] J. Liu, C. Chai, Y. Luo, Y. Lou, J. Feng, and N. Tang, "Feature augmentation with reinforcement learning," in *2022 IEEE 38th International Conference on Data Engineering (ICDE)*. IEEE, 2022, pp. 3360–3372.

[25] S. Galhotra, Y. Gong, and R. C. Fernandez, "Metam: Goal-oriented data discovery," in *ICDE*. IEEE, 2023.

[26] J. R. Vergara and P. A. Estévez, "A review of feature selection methods based on mutual information," *Neural computing and applications*, vol. 24, no. 1, pp. 175–186, 2014.

[27] A. Santos, A. Bessa, F. Chirigati, C. Musco, and J. Freire, "Correlation sketches for approximate join-correlation queries," in *Proceedings of the 2021 International Conference on Management of Data*, 2021, pp. 1531–1544.

[28] M. Esmailoghli, J.-A. Quiané-Ruiz, and Z. Abedjan, "Cocoa: Correlation coefficient-aware data augmentation." in *EDBT*, 2021, pp. 331–336.

[29] A. Santos, A. Bessa, C. Musco, and J. Freire, "A sketch-based index for correlated dataset search," in *2022 IEEE 38th International Conference on Data Engineering (ICDE)*, 2022, pp. 2928–2941.

[30] C. O. Daub, R. Steuer, J. Selbig, and S. Kloska, "Estimating mutual information using b-spline functions–an improved similarity measure for analysing gene expression data," *BMC bioinformatics*, vol. 5, no. 1, pp. 1–12, 2004.

[31] P. Mandros, M. Boley, and J. Vreeken, "Discovering reliable approximate functional dependencies," in *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2017, pp. 355–363.

[32] ——, "Discovering reliable correlations in categorical data," in *2019 IEEE International Conference on Data Mining (ICDM)*. IEEE, 2019, pp. 1252–1257.

[33] P. Mandros, D. Kaltenpoth, M. Boley, and J. Vreeken, "Discovering functional dependencies from mixed-type data," in *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, 2020, pp. 1404–1414.

[34] F. Pennerath, P. Mandros, and J. Vreeken, "Discovering approximate functional dependencies using smoothed mutual information," in *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, 2020, pp. 1254–1264.

[35] B. Youngmann, M. Cafarella, Y. Moskovitch, and B. Salimi, "On explaining confounding bias," in *2023 IEEE 39th International Conference on Data Engineering (ICDE)*. IEEE, 2023.

[36] K. Hlavackova-Schindler, M. Palus, M. Vejmelka, and J. Bhattacharya, "Causality detection based on information-theoretic approaches in time series analysis," *Physics Reports*, vol. 441 (2007) 1 – 46, 02 2007.

[37] G. Chandrashekar and F. Sahin, "A survey on feature selection methods," *Computers & Electrical Engineering*, vol. 40, no. 1, pp. 16–28, 2014, 40th-year commemorative issue. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0045790613003066

[38] J. Li, K. Cheng, S. Wang, F. Morstatter, R. P. Trevino, J. Tang, and H. Liu, "Feature selection: A data perspective," *ACM Comput. Surv.*, vol. 50, no. 6, dec 2017. [Online]. Available: https://doi.org/10.1145/3136625

[39] M. Beraha, A. M. Metelli, M. Papini, A. Tirinzoni, and M. Restelli, "Feature selection via mutual information: New theoretical insights," in *2019 International Joint Conference on Neural Networks (IJCNN)*. IEEE, 2019, pp. 1–9.

[40] H. Peng and Y. Fan, "Feature selection by optimizing a lower bound of conditional mutual information," *Information Sciences*, vol. 418, pp. 652–667, 2017.

[41] G. Brown, A. Pocock, M.-J. Zhao, and M. Luján, "Conditional likelihood maximisation: a unifying framework for information theoretic feature selection," *The journal of machine learning research*, vol. 13, pp. 27–66, 2012.

[42] M. S. Roulston, "Estimating the errors on measured entropy and mutual information," *Physica D: Nonlinear Phenomena*, vol. 125, no. 3-4, pp. 285–294, 1999.

[43] A. Hacine-Gharbi and P. Ravier, "A binning formula of bi-histogram for joint entropy estimation using mean square error minimization," *Pattern Recognition Letters*, vol. 101, pp. 21–28, 2018.

[44] L. Paninski, "Estimation of entropy and mutual information," *Neural computation*, vol. 15, no. 6, pp. 1191–1253, 2003.

[45] "scikit-learn: machine learning in python — scikit-learn 1.2.1 documentation," https://scikit-learn.org/.

[46] J. Jiao, K. Venkat, Y. Han, and T. Weissman, "Minimax estimation of functionals of discrete distributions," *IEEE Transactions on Information Theory*, vol. 61, no. 5, pp. 2835–2885, 2015.

[47] A. Kraskov, H. Stögbauer, and P. Grassberger, "Estimating mutual information," *Physical review E*, vol. 69, no. 6, p. 066138, 2004.

[48] B. C. Ross, "Mutual information between discrete and continuous data sets," *PloS one*, vol. 9, no. 2, p. e87357, 2014.

[49] W. Gao, S. Kannan, S. Oh, and P. Viswanath, "Estimating mutual information for discrete-continuous mixtures," *Advances in neural information processing systems*, vol. 30, 2017.

[50] D. Huang, D. Y. Yoon, S. Pettie, and B. Mozafari, "Joins on samples: a theoretical guide for practitioners," *Proceedings of the VLDB Endowment*, vol. 13, no. 4, pp. 547–560, 2019.

[51] E. Cohen, "Sampling big ideas in query optimization," in *Proceedings of the 42nd ACM SIGMOD-SIGACT-SIGAI Symposium on Principles of Database Systems*, 2023, pp. 361–371.

[52] D. Vengerov, A. C. Menck, M. Zait, and S. P. Chakkappen, "Join size estimation subject to filter conditions," *Proc. VLDB Endow.*, vol. 8, no. 12, p. 1530–1541, Aug. 2015. [Online]. Available: https://doi.org/10.14778/2824032.2824051

[53] Y. Chen and K. Yi, "Two-level sampling for join size estimation," in *Proceedings of the 2017 ACM International Conference on Management of Data*, ser. SIGMOD '17. New York, NY, USA: Association for Computing Machinery, 2017, p. 759–774. [Online]. Available: https://doi.org/10.1145/3035918.3035921

[54] V. Shah, J. Lacanlale, P. Kumar, K. Yang, and A. Kumar, "Towards benchmarking feature type inference for automl platforms," in *Proceedings of the 2021 International Conference on Management of Data*, ser. SIGMOD '21. New York, NY, USA: Association for Computing Machinery, 2021, p. 1584–1596. [Online]. Available: https://doi.org/10.1145/3448016.3457274

[55] V. Solo, "On causality and mutual information," in *2008 47th IEEE Conference on Decision and Control*, 2008, pp. 4939–4944.

[56] G. Doquire and M. Verleysen, "Feature selection with missing data using mutual information estimators," *Neurocomputing*, vol. 90, pp. 3–11, 2012, advances in artificial neural networks, machine learning, and computational intelligence (ESANN 2011). [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0925231212001841

[57] M. Hutter and M. Zaffalon, "Distribution of mutual information from complete and incomplete data," *Computational Statistics & Data Analysis*, vol. 48, no. 3, pp. 633–657, 2005.

[58] S. Acharya, P. B. Gibbons, V. Poosala, and S. Ramaswamy, "Join synopses for approximate query answering," *SIGMOD Rec.*, vol. 28, no. 2, p. 275–286, Jun. 1999. [Online]. Available: https://doi.org/10.1145/304181.304207

[59] A. Bessa, M. Daliri, J. Freire, C. Musco, C. Musco, A. Santos, and H. Zhang, "Weighted minwise hashing beats linear sketching for inner product estimation," in *Proceedings of the 42nd ACM SIGMOD-SIGACT-SIGAI Symposium on Principles of Database Systems*, 2023.

[60] K. Beyer, P. J. Haas, B. Reinwald, Y. Sismanis, and R. Gemulla, "On synopses for distinct-value estimation under multiset operations," in *Proceedings of the 2007 ACM SIGMOD International Conference on Management of Data*, ser. SIGMOD '07. New York, NY, USA: ACM, 2007, pp. 199–210. [Online]. Available: http://doi.acm.org/10.1145/1247480.1247504

[61] E. Cohen, "Coordinated sampling," in *Encyclopedia of Algorithms*, 2016, pp. 449–454. [Online]. Available: https://doi.org/10.1007/978-1-4939-2864-4_576

[62] M. Daliri, J. Freire, C. Musco, A. Santos, and H. Zhang, "Sampling methods for inner product sketching," *arXiv preprint arXiv:2309.16157*, 2023.

[63] C. Estan and J. F. Naughton, "End-biased samples for join cardinality estimation," in *22nd International Conference on Data Engineering (ICDE'06)*, 2006, pp. 20–20.

[64] D. Knuth, Addison-Wesley, and P. Education, *The Art of Computer Programming*, ser. Addison-Wesley series in computer science and information processing. Addison-Wesley, 1997, no. v. 3.

[65] J. S. Vitter, "Random sampling with a reservoir," *ACM Transactions on Mathematical Software (TOMS)*, vol. 11, no. 1, pp. 37–57, 1985.

[66] C. Wang and B. Ding, "Fast approximation of empirical entropy via subsampling," in *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, 2019, pp. 658–667.

[67] X. Chen and S. Wang, "Efficient approximate algorithms for empirical entropy and mutual information," in *Proceedings of the 2021 International Conference on Management of Data*, 2021, pp. 274–286.

[68] N. Duffield, C. Lund, and M. Thorup, "Priority sampling for estimation of arbitrary subset sums," *J. ACM*, vol. 54, no. 6, p. 32–es, Dec. 2007. [Online]. Available: https://doi.org/10.1145/1314690.1314696

[69] H.-O. Georgii, *Stochastics: Introduction to Probability and Statistics*. Berlin, Boston: De Gruyter, 2012. [Online]. Available: https://doi.org/10.1515/9783110293609

[70] Wikipedia contributors, "Multinomial distribution — Wikipedia, the free encyclopedia," 2023, [Online; accessed 01-August-2023]. [Online]. Available: https://en.wikipedia.org/w/index.php?title=Multinomial_distribution&oldid=1167221208

[71] "World Bank Group Finances," https://finances.worldbank.org.

[72] "The Socrata Open Data API," https://dev.socrata.com.

[73] "The Tablesaw Library," https://github.com/jtablesaw/tablesaw.

[74] A. D. Sarma, L. Fang, N. Gupta, A. Y. Halevy, H. Lee, F. Wu, R. Xin, and C. Yu, "Finding related tables," *Proceedings of the 2012 ACM SIGMOD International Conference on Management of Data*, 2012.

[75] 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.

[76] J. Becktepe, M. Esmailoghli, M. Koch, and Z. Abedjan, "Demonstrating mate and cocoa for data discovery," in *Companion of the 2023 International Conference on Management of Data*, 2023, pp. 119–122.

[77] P. Indyk and A. McGregor, "Declaring independence via the sketching of sketches," in *Proceedings of the Nineteenth Annual ACM-SIAM Symposium on Discrete Algorithms*, ser. SODA '08. USA: Society for Industrial and Applied Mathematics, 2008, p. 737–745.

[78] F. Keller, E. Müller, and K. Böhm, "Estimating mutual information on data streams," in *Proceedings of the 27th International Conference on Scientific and Statistical Database Management*, ser. SSDBM '15. New York, NY, USA: Association for Computing Machinery, 2015. [Online]. Available: https://doi.org/10.1145/2791347.2791348

[79] J. Boidol and A. Hapfelmeier, "Fast mutual information computation for dependency-monitoring on data streams," in *Proceedings of the Symposium on Applied Computing*, ser. SAC '17. New York, NY, USA: Association for Computing Machinery, 2017, p. 830–835. [Online]. Available: https://doi.org/10.1145/3019612.3019669

[80] M. Ferdosi, A. Gholamidavoodi, and H. Mohimani, "Measuring mutual information between all pairs of variables in subquadratic complexity," in *Proceedings of the Twenty Third International Conference on Artificial Intelligence and Statistics*, ser. Proceedings of Machine Learning Research, S. Chiappa and R. Calandra, Eds., vol. 108. PMLR, 26–28 Aug 2020, pp. 4399–4409. [Online]. Available: https://proceedings.mlr.press/v108/ferdosi20a.html

[81] D. McAllester and K. Stratos, "Formal limitations on the measurement of mutual information," in *International Conference on Artificial Intelligence and Statistics*. PMLR, 2020, pp. 875–884.

[82] S. Gao, G. Ver Steeg, and A. Galstyan, "Efficient estimation of mutual information for strongly dependent variables," in *Artificial intelligence and statistics*. PMLR, 2015, pp. 277–286.