Efficient Algorithms for Correlated Data Discovery

DISSERTATION

Submitted in Partial Fulfillment of

the Requirements for

the Degree of

DOCTOR OF PHILOSOPHY (Computer Science)

at the

NEW YORK UNIVERSITY
TANDON SCHOOL OF ENGINEERING

by

Aécio Solano Rodrigues Santos

May 2024

Efficient Algorithms for Correlated Data Discovery

DISSERTATION

Submitted in Partial Fulfillment of

the Requirements for

the Degree of

DOCTOR OF PHILOSOPHY (Computer Science)

at the

NEW YORK UNIVERSITY
TANDON SCHOOL OF ENGINEERING

by

Aécio Solano Rodrigues Santos

May 2024

Approved:

Department Chair Signature

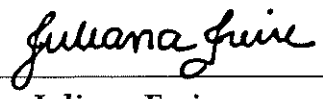
May 10, 2024

Date

University ID: N10789003
Net ID: asr498

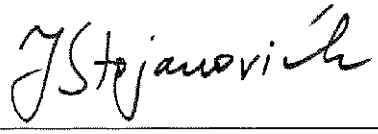
Approved by the Guidance Committee:

Major: Computer Science



Juliana Freire
Professor
NYU Tandon School of Engineering
May 1, 2024
Date

Christopher Musco
Assistant Professor
NYU Tandon School of Engineering
May 1, 2024
Date



Julia Stoyanovich
Institute Associate Professor
NYU Tandon School of Engineering
May 1, 2024
Date

Flip Korn
Research Scientist
Google Research
May 1, 2024
Date

Vita

Aécio Solano Rodrigues Santos was born in Teresina (Piauí), Brazil, in 1991. He has a Technology Degree in Analysis and Development of Systems from the Instituto Federal do Piauí (IFPI) and an M.S. in Computer Science from the Universidade Federal de Minas Gerais (UFMG), Brazil. He worked as a software engineer for a Brazilian start-up called Zunnit Technologies, where he helped develop a scalable news recommendation platform that was used by some of the major Brazilian media companies. In 2015, he moved to New York City to join New York University (NYU) as a Research Engineer, where he worked on multiple research projects (including DARPA's Memex and D3M programs) that required a combination of knowledge and skills in Software Engineering, Machine Learning, and Information Retrieval (Web Search). He started his Ph.D. in January 2020, where he worked on new methods for improving dataset search and discovery systems. During his Ph.D. program, he received the 2022 Deborah Rosenthal, MD Award for outstanding performance on the Ph.D. qualifying examination and the 2024 Pearl Brownstein Doctoral Research Award, which is given to students whose doctoral research shows the greatest promise. His research was supported by grants from the NSF, DARPA, and Google Research.

Acknowledgements

I would like to express my sincere gratitude to Prof. Juliana Freire, who has not only been a Ph.D. advisor but also a supervisor and collaborator since long before the beginning of my Ph.D. studies. We have been working together at NYU for over nine years, which is by far the longest time I have been affiliated with any school or organization. It has been a long ride with a lot of learning and growth, and hence I cannot describe in a few words my appreciation for your continuous support, experience, and influence in my past and future career.

I would also like to thank the committee members of my dissertation: Christopher Musco, Flip Korn, and Julia Stoyanovich. Your insightful questions and comments on my work led me to think more broadly and encouraged me to fill some fundamental missing gaps in this work. Special thanks go to Flip Korn and Chris Musco, who were also incredible mentors and have contributed significant ideas that are included in this dissertation. I have learned a lot from you. Thanks, Flip, for your careful diligence and attention to details that would have been easily missed by many; Thanks, Chris, for your patience and tutoring; before meeting you, I had never thought that one day I would ever be involved in research on the theoretical aspects of algorithms.

Having come so far, it could be easy to just forget one's early steps. However, I would like to thank all of you whom I met at every step of my academic and professional career. In the name of Prof. Nivio Ziviani (UFMG), I would like to thank all my previous mentors, supervisors, colleagues, and collaborators at IFPI, UFMG, and Zunnit from whom I have learned so much over the years. The inspiration and knowledge that I gained from all of you is an important part of who I am today and has substantially contributed to my current research accomplishments.

After being nine years at NYU VIDA lab and observing its constant flux of brilliant minds that come and go every year, the list of colleagues and friends that I owe a thanks has grown so much that I can't possibly list all of you here. Some of you were there since the very beginning – Ann Borray, Yamuna Krishnamurthy, Kien Pham, Rémi Rampin, Aline Bessa, Fernando Chirigati, Raoni Lourenço, Jorge Ono; Some of you arrived soon after me – Sonia Castelo, Vicky Rampin, João Rulff, Roque Lopez; Some of you arrived not long ago – Haoxiang, Yurong, Stella, Vitória, Priscylla, Eduardo, Suemy, Felipe; Some of you were never even officially at NYU but sure were and are a part of our extended family – Daniela Bertoli, Chris Shultz, Laura Durbin. I don't have proof yet, but I conjecture that there is a better sampling-based sketching algorithm that would have chosen a different and more comprehensive subsample of names of other very deserving people to be included here. Or perhaps an LLM could help write a better, personalized, a less boring acknowledgments section for each reader, but I digress. I'm grateful for having met all of you and have had the opportunity to join the lively discussions and and share the laughter in numerous moments in and off campus. You are what makes VIDA a good place to be in.

Last but not least, I have to thank my family. Without their unwavering support, I would have never gotten here. Thanks to my father, my mother, and brother, who are always rooting for me and are a constant source of inspiration and motivation; and to my partner, Juliana Barbosa, who was always together with me not only during most of the good moments mentioned above but also has been there for all the bad moments and has always held my hand. I can never thank you all enough.

Aécio Santos,

May 2024.

To my loving family.

ABSTRACT

Efficient Algorithms for Correlated Data Discovery

by

Aécio Solano Rodrigues Santos

Advisor: Prof. Juliana Freire, Ph.D.

Submitted in Partial Fulfillment of the Requirements for
the Degree of Doctor of Philosophy (Computer Science)

May 2024

The increase in our ability to collect and store data has led to an explosion in the number of data repositories containing both public and enterprise data. While this abundance creates exciting opportunities for data analytics and modeling, it also makes it harder to find data that is relevant to a given task. In this dissertation, we argue that existing data discovery systems have important limitations that hamper a user's ability to find data within large repositories that satisfy a given constraint (e.g., joinability and data correlations). We propose new systems and methods that address these limitations and make dataset discovery effective and efficient.

The first technical contribution of this dissertation is the design and implementation of Auctus, a search engine tailored for dataset discovery. With Auctus, we introduced new methods to improve dataset search, from the automatic generation of metadata from the dataset content to a user interface that streamlines the search process and the selection of relevant datasets. In addition, Auctus supports dataset-oriented queries which allow automatic augmentation of a given dataset using data discovered from datasets published in open-data repositories on the Web. Our experience with the development of Auctus led to the identification of key limitations of existing data discovery methods that we addressed in subsequent work.

The second contribution is the introduction and formalization of the concept of Join-Correlation Queries (JCQ): Given a query table T with a column Q and a join column K, the goal is to find tables in a large data collection that both (1) are joinable with the query table T on the join column K and (2) contain a column C that is strongly correlated with Q. This class of queries is motivated by the opportunity to discover useful features in large data repositories that may explain a target variable of interest or improve the predictive power of machine learning models. Some key challenges in evaluating these queries are performance and scalability. To solve these problems, we 1) propose a new sampling-based sketch that enables the construction of an index for a large number of tables and that provides accurate estimates for join-correlation queries, and 2) explore different scoring strategies that effectively rank the query results based on how well the columns are correlated with the query.

To further improve our approach, we refine our method and sketching algorithms in two directions. First, we improve our sketches with a tuple-based sampling scheme that allows handling joins over non-unique keys and demonstrate that this new method can effectively estimate complex non-linear relationships such as mutual information

over various data types (including numerical and categorical). Second, we generalize our definition of JCQ to Weighted Join-Correlation Queries, allowing users to specify preferences for joinability and correlation strength. Additionally, we propose a novel hashing scheme, QCR, leading to another indexing scheme that allows answering (Weighted) Join-Correlation Queries even more efficiently and achieves better trade-offs that significantly improve both the ranking accuracy and recall.

Finally, we show that the algorithms we propose are justified by formal theoretical analysis and extensive experimental evaluations, which verify the accuracy and efficiency of our methods across diverse synthetic and real-world datasets.

Table Of Contents

List of Figures

List of Tables

Chapter 1

Introduction

Thesis Statement. *Existing systems and methods do not provide adequate support for interactive discovery of data relationships (such as correlations) for datasets stored in large data repositories. We aim to make data discovery more effective and efficient by designing new systems, methods, and algorithms that expand the set of information needs that can be supported by data discovery systems, with a special focus on correlated data search over large data repositories.*

The amount of data that we can collect and store is constantly increasing. This has led to an explosion in the number of data repositories containing both public [121, 122, 123, 25] and enterprise data [4, 81, 105]. This data abundance creates new opportunities to enrich data analysis and improve machine learning models: by incorporating information from various external datasets, we can explain confounding bias [168], test hypotheses and explain salient features in data [40, 14, 12], as well as improve predictive models [39, 31].

While data is abundant, it is difficult to find *relevant data*. Not surprisingly, data search has received substantial attention both in industry and academia. Several "data catalog" systems that support data search across multiple systems and enterprise data lakes are available [81, 105, 4, 162]. The market for such systems is projected to grow from USD 528.3 million in 2019 to USD 5.46 billion in 2027 [130]. Google Dataset Search supports search over datasets that are published on the Web [25]. There are also open-source systems, such as CKAN [41], Socrata [147], and DataVerse [54], that are used to deploy vertical data repositories (e.g., NYC Open Data [121], Data.gov [52], and Harvard's DataVerse [55]). These systems fill important gaps in the data discovery space, notably by providing a unified interface to large collections of datasets collected from multiple, distributed sources. However, they have important limitations when it comes to query capabilities: their interfaces are often limited to simple keyword-based queries and faceted search over dataset metadata. These types of queries have limited expressiveness, making it difficult (and sometimes impossible) to express specific information needs.

In this dissertation, we design new systems and methods with the overarching goal of enabling *interactive data discovery* through *data-driven queries*. We propose novel types of *data-driven relationship queries* that enable the *discovery of tables* in a data repository that are related to an input query table *according to a given relationship measure*. Such measures can be simple linear relationships such as Pearson's correlation coefficient, non-linear correlations such as Spearman's correlation, or even more complex dependence measures such as Mutual Information (MI). Moreover, we propose *efficient sketching algorithms* that allow performing these queries efficiently over large dataset collections. We also provide theoretical analyses and empirical evaluations to demonstrate the effectiveness and runtime performance of the proposed algorithms.

1.1 Motivation

We use two examples to motivate the importance of data-driven relationship queries. The first illustrates the wide applicability of *correlated dataset search* by discussing the relationship between linear correlation and simple linear regression. The second provides a concrete and practical example that illustrates how adding new variables from external datasets (which we refer to as *relational data augmentation* in this dissertation) can lead to the improvement predictive machine-learning models.

Example 1 (*The Case for Correlated Data Search*). Numerical correlations have many important practical applications, which range from the use of data analytics for understanding real-world phenomena and confirming (or refuting) hypotheses [11, 99, 131, 9] to improving machine learning models through feature selection [84, 85]. To illustrate this, we use the example of *Simple Linear Regression* [129], one of the most fundamental and widely used methods for data analysis.

Linear Regression is widely used to estimate the parameters in a linear equation that predicts values of a variable Y based on another variable X. Formally, we can say that $Y = b_0 + b_1X$, where b_0 is the intercept and b_1 is the slope of the line. A linear regression "learns" the parameters b_1 and b_0 of the linear equation predicting an outcome variable, Y, based on values of a predictor variable, X.

The Pearson's correlation coefficient, which provides a measure of the strength of association between two variables, is closely related to linear regression since we can view it as the standardized slope of the regression line. It can be shown that $b_1 = r\frac{s_X}{s_Y}$, where s_i is the standard deviation of the variable i, and r is the Pearson's correlation between X and Y [132].

In other words, the stronger the correlation is, the higher the predictive power of a variable. This shows that by finding correlated variables in large dataset collections, we can discover data that may "explain" or "predict" other variables of interest. While correlations do not imply causation, discovering data correlations is a starting point for more detailed analyses that identify true causal data relationships and, in turn, can be used to improve predictive models. □

Example 2 (*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 1.1(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 1.1(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 1.1(c)). The association between demand and population for the ZIP Codes of pick-up locations suggests a strong dependency. □

In Example 2 (*Understanding Taxi Demand*), the analysts knew (or had an intuition) that weather is likely to impact taxi demand. They also knew that a dataset containing such data was available, and knew where to find it. However, often this is not the case: finding relevant data is a difficult and time-consuming task [73]. On the Web, data

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).

Figure 1.1: 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`.

are distributed over many sites and repositories, and in enterprises, they are stored in a plethora of systems and databases. Thus, to discover external relevant tables for augmenting their data, users must go through a time-consuming manual process that includes finding the datasets, joining the datasets with their data, and retraining their models to verify if the new data leads to improvements. As a point of reference, internal research at Lyft has found that data discovery is the second-most time-consuming activity for their data scientists' workflows (25% of their time) – they only spend more time (40%) on model development and product deployment [81].

1.2 Challenges

Discovery Effectiveness and Efficiency. In an attempt to optimize this process, several methods have been proposed for automating *relational data augmentation* [178, 39, 32, 73, 177, 119, 62, 120]. Most of the existing work focuses on efficient support for data integration queries, including algorithms to discover "joinable" tables [178, 32, 73, 177, 166, 120, 69], which are typically based on join key containment or overlap. These

methods have an important limitation in that they can return *too many irrelevant tables*. Consider our example: if we search the NYC Open Data repository [121] to discover additional features that can help improve taxi demand prediction, we will find a very 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 prohibitively expensive.

Automatic relational data augmentation systems [39, 96, 110, 76] attempt to address this problem by applying feature selection techniques on pre-joined tables [157]: given a list of joinable tables, they first materialize the joins before automatically selecting attributes that increase the accuracy of the predictive model given as input. Because they rely on data discovery systems [73, 31] to return the joinable tables, they can end up with both a *large number of joins* to perform and, consequently, too many irrelevant features to consider. Given the *high cost of evaluating the joins* and the computational complexity of feature selection methods, it is expensive to identify the useful features.

Diversity of Information Needs. Another challenge arises from the *diversity of data types and information needs* of different applications. Traditional correlation measures are only applicable to numerical data and do not capture non-monotonic relationships that naturally occur in real data. For instance, consider again the tables in Figure 1.1. While the Pearson's correlation coefficient may be used to identify the relationship between `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).

Ideally, data discovery methods should be flexible enough to support (1) multiple data types that exist in real data tables and (2) general data relationship measures. For instance, we should be able to use a more general measure of statistical dependence such as Mutual Information (MI), which is invariant under homomorphism and applies to different data types. Due to its generality, MI has found applications in numerous problems including the analysis of gene expression [56], functional dependency discovery [113, 114, 115, 127], explanatory data analysis [169], and causality detection [87]. In machine learning, MI is also used in many feature selection methods [34, 108, 157]. 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 [8, 126, 27].

However, 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 [134]. While MI is well-defined for different data types, *estimators that work with different data types usually have different bias and variance characteristics.* Some approaches address this issue by transforming numerical data to categorical before applying an estimator [83], however, these techniques have their own problems and it is unclear if this is more effective than using a combination of estimators for different data types.

Efficient Relationship Estimation Across Different Tables. In the particular setting of relational data augmentation, these issues are compounded by additional problems created by the need for efficiency and different types of join key distributions, which leads to different join types (e.g., one-to-one or many-to-one). 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. In such tables,

each row represents an instance or entity of interest. 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 1.1(c) may have unique values, the derived `Population` attribute in the augmented table shown in Figure 1.1(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 [79].

Second, to avoid the cost of fully materializing joins, systems are limited to estimating MI using a *small number of samples* of the join. 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 [92] which results in poor accuracy. State-of-the-art methods typically use coordinated sampling (based on minwise hashing) to increase the number of samples that contribute to the join [43]. 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 estimators is further increased (as shown in Chapter 5).

1.3 Contributions

This dissertation contributes new systems and methods with the overarching goal of supporting *interactive data discovery* through *data-driven relationship queries*. The first contribution is the design and implementation of *Auctus*, a dataset search engine that allows searching over datasets published in multiple open-data repositories on the Web. With Auctus, we introduced new methods to improve dataset search, including the automatic profiling of tables to support search, and a user interface that streamlines the search process and the selection of relevant datasets. In addition, Auctus supports dataset-oriented queries that allow augmentation of a given dataset using data discovered from these repositories. In Chapter 3, we describe Auctus' architecture, user interface, and novel use cases that Auctus enables.

To achieve interactive response times, Auctus' implementation uses state-of-the-art methods that enable efficient query execution of data discovery queries, such as indexes for finding joinable tables based on locality-sensitive hashing (LSH) [32]. Our experience developing and using Auctus allowed us to identify key limitations in the existing techniques described in the literature. For instance, existing methods did not cater for the information needs involving correlations discussed in our Examples 1 and 2. To address these limitations, we proposed new methods and algorithms that allow efficient evaluation of data discovery queries that support these needs. Next, we discuss these contributions in detail.

1.3.1 Join-Correlation Queries

Our second contribution is the proposal and formalization of a new class of queries, which we refer to as *Join-Correlation Queries*, that more precisely express the information needs described in Examples 1 and 2. Formally, we define it as follows:

Definition 1 (Join-Correlation Query)**.** *Given a column Q and a join column K_Q from a query table $\mathcal{T}_Q$, a* join-correlation query *finds tables $\mathcal{T}_X$ in a dataset collection such that $\mathcal{T}_X$ is joinable with $\mathcal{T}_Q$ on K_Q and there is a column $C \in \mathcal{T}_X$ such that Q is strongly correlated with C.*

In Example 2, the analysts could issue a query to retrieve datasets $\mathcal{T}_X$ that join with the daily taxi trips dataset ($\mathcal{T}_Q = \mathcal{T}_{taxi}$) on join columns (e.g., $K_Q = \texttt{Date}$ or $K_Q = \texttt{ZipCode}$) and that also contain a column that correlates with the actual number of trips ($Q = \texttt{NumTrips}$). As output, the query returns a list of tables $\mathcal{T}_X$ that contain correlated columns. For instance, if $\mathcal{T}_Q = \mathcal{T}_{taxi}$ and $K_Q = \texttt{ZipCode}$, then the output would include $\mathcal{T}_X = \mathcal{T}_{demographics}$ with $C = \texttt{Population}$, assuming a strong correlation between $\texttt{Population}$ and $\texttt{NumTrips}$.

Note that this definition is rather generic and does not strictly specify the meaning of "*strongly correlated*", as this depends on the application information needs. For example, the correlation could be measured using a traditional linear correlation measure such as Pearson's correlation coefficient or a non-linear dependence measure such as Mutual Information. Moreover, the "strength" of the correlation could mean that the correlation is greater than a given threshold τ or it could indicate that the correlation is among the top-k greatest correlations (for given a $k > 0$) in a table collection. In this dissertation we use these two meanings, and the semantics of "*strongly correlated*" will be clearly specified when describing each specific algorithm and its experimental evaluation.

We also did not formally specify the semantics of "joinable". In this dissertation, it refers to the ability to join two tables using a relational equi-join operation, and it is typically measured using the join key's *Overlap* or *Jaccard Containment*. Furthermore, the query did not prescribe how a user may specify (or combine) the levels of joinability and correlation. We address this problem in Chapter 6, where we propose *Weighted Join-Correlation Queries*, a generalization of this query definition that provides better control over its parameters to the user.

1.3.2 Algorithms for Evaluating Join-Correlation Queries

The main issue in evaluating *join-correlation queries* is how to do so *efficiently*. A naïve approach first finds joinable tables, and then explicitly computes correlations between Q and all columns of all discovered tables. However, this approach requires the joinable datasets returned to be downloaded and joined with the query table. When these tables are large, they may not fit in memory and the cost of executing join operations can be prohibitive. Furthermore, some correlation measures are expensive to compute, e.g., to compute Spearman's correlation the data must first be sorted. This problem is compounded for queries that return a large number of datasets and require many joins and correlation computations to be performed.

In Chapter 4, we address this problem by proposing efficient probabilistic algorithms for evaluating join-correlation queries that trade off accuracy by performance. We 1) propose a sketching method that enables the construction of an index for a large number of tables and that provides accurate estimates needed for answering join-correlation queries, and 2) explore different scoring strategies that effectively rank the query results based on how well the columns are correlated with the query. Here, our main contribution is a formalization of the problem of *join-correlation estimation* along

with a new sketching algorithm (CSK) that allows efficient approximation (with accuracy guarantees) of the correlation between columns of unjoined tables without materializing their join. This sketch is a key component that allows for efficient evaluation of join-correlation queries without incurring the cost of joining tables. We also conduct an experimental evaluation using both synthetic and real data, which shows that our sketches attain high accuracy and the scoring strategies lead to high-quality rankings.

In Chapter 5, we refine our sketching algorithm for estimating join-correlation in two directions. First, we propose a new algorithm (TUPSK) that does not assume that the join key column on the left side of the join has unique values, solving a key limitation of the CSK algorithm proposed in Chapter 4. Second, we extend our sketches to estimate correlations between columns of different data types, including continuous (numerical), discrete (categorical), and mixtures of discrete-numerical data using Mutual Information (MI). We also perform an extensive evaluation of our sketches coupled with several MI estimators. This evaluation allows us to (1) confirm the superiority of TUPSK over CSK for general left joins, and (2) identify differences in the behavior of various combinations of sketches and MI estimators, and *when* and *why* they fail.

In Chapter 6, we generalize our initial definition of join-correlation queries (from Definition 1) to incorporate user preferences (weights) regarding joinability and correlation. We refer to this refinement as *Weighted Join-Correlation Queries*. To evaluate them, we propose a novel hashing scheme, namely QCR, that enables the construction of indexes that enable retrieval of columns that are both joinable and correlated *in a single step*. Our experimental evaluation shows that this approach leads to a significant increase in both recall and ranking quality at a smaller storage cost compared to indexes created using only the CSK hashing scheme.

1.4 Summary and Dissertation Outline

In summary, this dissertation provides the following contributions:

- We describe the design and architecture of a dataset search engine that supports interactive data discovery and augmentation (Chapter 3);
- We formalize the idea of Join-Correlation Queries and Join-Correlation Estimation (Chapter 4);
- We propose the idea of "correlation sketches" and an approach for efficiently evaluating join-correlation queries using CSK sketches (Chapter 4);
- We describe new correlation bounds and scoring functions for ranking tables using join-correlation estimates computes using CSK sketches (Chapter 4);
- An experimental evaluation of CSK sketches using synthetic and real-world data that verifies their effectiveness and efficiency (Chapter 4);
- We propose TUPSK, an alternative method for building correlation sketches that is suitable for estimating quantities over many-to-one joins (Chapter 5);
- An experimental evaluation of MI estimation using multiple MI estimators with TUPSK sketches, which allows us (1) to assess the effectiveness of TUPSK sketches and (2) to identify differences in the behavior of various combinations of sketches and MI estimators for different data types, and when and why they fail (Chapter 5);
- We propose Weighted Join-Correlation Queries as a way to provide more control to users over query parameters (Chapter 6);
- We propose a new indexing approach, based on our new QCR hashing scheme, for evaluating (weighted) join-correlation queries in a single stage (Chapter 6);

- An experimental evaluation of QCR indexes using synthetic and real-world data that shows they are more time and space efficient than alternative approaches (Chapter 6);
- An experimental evaluation of QCR and CSK-based methods for discovering and ranking relevant features over large data lakes and wide tables (Section 7).

The system described in Chapter 3 has been described in papers published at VLDB 2022 [31] and SIGMOD 2021 [12]. The research described in Chapters 4, 5, and 6 was led by this dissertation's author and has been published at *ACM SIGMOD 2021* [136], and *IEEE ICDE 2024* [139] and *IEEE ICDE 2022* [137], respectively. Additionally, related papers that spun out of this research have been published at PODS 2023 [13] and VLDB 2024 [48]. Of particular interest is [48], which includes another sketching approach for estimating join-correlation that is not included in this dissertation.

Chapter 2

Background & Related Work

The methods proposed in this dissertation are based on hashing techniques that have been used for several problems, including cardinality estimation of sets, set intersection, and set unions. In this chapter, we cover in detail some background knowledge that serves as the basis for these methods. We also cover the literature on correlation estimation and discuss the properties of correlation estimators that we use in this dissertation, as well as some information-theoretic measures such as entropy, differential entropy, and mutual information. Finally, in Section 2.4, we review additional research related to the systems and methods proposed in this dissertation.

2.1 Cardinality Estimation via Sketches

Estimating Distinct Values. The problem of determining the number of distinct values (DV) in a dataset has been extensively studied [29, 95, 124]. Since computing the *exact* number of distinct elements is expensive, approximate methods have been proposed that can scale to massive collections of datasets. Effective approaches for DV

estimation rely on hashing techniques, require a single pass through the data, and use a bounded amount of memory [86].

Let h_u be a hash function that maps distinct values randomly and uniformly to the unit interval $[0, 1]$, and D be the number of distinct elements in a dataset. The key idea behind DV estimators is that, if we use h_u to map elements to the unit interval and the number of distinct elements in a dataset is large (i.e., $D \gg 1$), then the expected distance between any two neighboring points in the unit interval is $1/(D + 1) \approx 1/D$, and the expected value of the k^{th} smallest point, $U(k)$, is estimated with $\mathbb{E}[U(k)] \approx \sum_{j=1}^{k}(1/D) = k/D$. Thus, the number of distinct values in the dataset can be approximated by $D \approx k/\mathbb{E}[U(k)]$. The simplest estimator of $\mathbb{E}[U(k)]$ is $U(k)$ itself, yielding the basic estimator: $\hat{D}_k^{BE} = k/U(k)$.

Based on this idea, algorithms and methods for building sketches have been developed to estimate set cardinality [15]. An example is the popular k *Minimum Values* (KMV) sketch (also known as bottom-k sketches), introduced by Bar-Yossef et al. [5]. Concretely, a KMV sketch of a set X comprises the k minimum hash values of the elements of X, generated by a hash function h_u mapping to $[0, 1]$. To estimate the number of distinct elements $|X|$, one can use this sketch and a DV estimator, such as $\hat{D}_k^{BE}$. Alternatively, an improved DV estimator proposed by Beyer et al. [16] can be used. Their estimator, given by $\hat{D}_k^{UB} = (k - 1)/U(k)$, is unbiased, has a lower mean squared error, and has the minimal possible variance of any DV estimator when there are many distinct values and the sketch size is large.

Cardinality Estimation under Set Operations. Beyer et al. [16] also considered how multiple KMV sketches, created independently, can be combined to estimate the cardinality of sets that result from multi-set operations (e.g., union, intersection, and difference). As an example, consider a set X composed of two partitions X_A and X_B,

i.e., $X = X_A \cup X_B$. Next, let L_A and L_B be the KMV sketches of sets X_A and X_B, with sizes k_A and k_B respectively. We can combine L_A and L_B to build a valid KMV sketch $L = L_A \oplus L_B$, where $\oplus$ is an operator for combining two KMV sketches. L represents the set comprising the k smallest values in $L_A \cup L_B$, where $k = min(k_A, k_B)$. To estimate the number of distinct elements $D_\cup$ in the union $X_A \cup X_B$, we can directly use estimator $\hat{D}_k^{UB}$ on the sketch L. To estimate the number of distinct values $D_\cap$ in the intersection $X_A \cap X_B$, we must first compute the number of common distinct hashes in L_A and L_B (i.e., $K_\cap = |\{v \in L : v \in L_A \cap L_B\}|$). Then, we can estimate $D_\cap$ as:

$$\widehat{D_\cap} = \frac{K_\cap}{k}\frac{k-1}{U(k)} \tag{2.1}$$

2.2 Correlation Estimation

The problem of measuring dependence between a pair of vectors has been studied for over a century [132], and new correlation measures continue to be developed [3, 149]. Pearson's correlation coefficient is one of the oldest and most widely used correlation measures [129]. While our methods can be used to estimate any measure of correlation, we use Pearson's as our main motivating example.

When applied to a population, Pearson's correlation coefficient is usually referred to as ρ [129]. For a pair of random variables $\langle X, Y \rangle$, the coefficient is defined as:

$$\rho_{XY} = \frac{\mathbb{E}[(X - \mu_X)(Y - \mu_Y)]}{\sigma_X \sigma_Y} \tag{2.2}$$

where σ_X (resp. σ_Y) is the standard deviation of the random variable X (resp. Y), and μ_X (resp. μ_Y) is the mean of X (resp. Y). ρ_{XY} can be estimated with a finite sample from distributions X and Y using what is usually referred to as Pearson's sample

correlation (r):

$$r_{XY} = \frac{\sum_{i=1}^{n}(x_i - \bar{x})(y_i - \bar{y})}{\sqrt{\sum_{i=1}^{n}(x_i - \bar{x})^2}\sqrt{\sum_{i=1}^{n}(y_i - \bar{y})^2}}. \tag{2.3}$$

Above n is the sample size, x_i and y_i are individual samples, and $\bar{x}$ and $\bar{y}$ are, respectively, the means of the sub-samples of X and Y.

There is a lot of prior work on understanding the accuracy of sample correlation estimators, but these works typically make strong data assumptions. When data follows a bivariate normal distribution, the sampling distribution of r is asymptotically normal and centered around ρ [144]. For a finite sample of size n, the variance of r is known to depend both on n and on the underlying population correlation ρ [145]:

$$Var(r) = \frac{(1-\rho^2)^2}{n-1} \tag{2.4}$$

When data is not normally distributed (as is often the case in practice), less is known. Nevertheless, there has been an increasing interest in the non-normal setting in recent years [170, 57, 17, 18, 19, 20, 91]. Yuan and Bentler [170, 171] show asymptotically that the standard deviation of the sample estimator r depends on the joint fourth-order moments, or kurtoses, of the variables. In agreement, empirical simulations confirm that the presence of a high excess kurtosis can lead to increased bias and estimator errors [57, 19].

Robustness and Alternative Correlations. One challenge in obtaining finite sample accuracy bounds, as we do in Section 4.3.3, is that Pearson's correlation coefficient is known to be sensitive to outliers. In fact, it has been shown that a single sample (x_i, y_i) can have an unbounded effect on the correlation and can potentially lead to catastrophic estimation errors [59]. This fact has spurred the development of correlation estimators that are robust against outliers and distribution contamination [59, 144].

However, these robust estimators are less efficient than r, i.e., require larger sample sizes. We refer the reader to [144] for an extensive review of robust correlation estimators. Resampling-based approaches such as bootstrapping [67] can also be used to reduce error in estimating Pearson's correlation, especially at small sample sizes [19]. These approaches, however, have a much higher computational cost [19].

Alternative correlation measures, such as the Spearman's rank correlation coefficient [129], and data distribution transformations, such as the Rank-based Inverse Normal (RIN) [21, 18], may be more effective than Pearson for highly non-normal data [19]. However, they have different semantics from Pearson's (e.g., they capture non-linear relationships), and thus the choice of correlation measure depends on the user's application. All of these correlations can be estimated with the sketching methods proposed in this dissertation, and we experimentally study the accuracy of their estimators when used with our sketching methods in real-world data collections (see Section 4.4).

2.3 Mutual Information

Most traditional correlation measures (as discussed in Section 2.2) are limited to measuring relationships between numeric data types. In this section, we introduce Mutual Information (MI), a different approach to measuring data relationships between variables based on information theory. Because MI is defined on probability spaces, it is more general and applies to variables of any data type that we can estimate their probability distributions. Before formally introducing mutual information, we first present the terminology used in this dissertation and the concept of entropy, which is needed for the MI definition.

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 [143]. In reality, 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 5.3.

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 1.1(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:

$$H(X) = \mathbb{E}[-\log p(X)] = -\sum_{i=1}^{u_X} p(i) \log p(i) \tag{2.5}$$

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:

$$H(X) = \mathbb{E}[-\log f(X)] = -\int_{\mathcal{X}} f(x) \log f(x)\, dx \tag{2.6}$$

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:

$$H(X,Y) = -\sum_{i=1}^{u_X}\sum_{j=1}^{u_Y} p(i,j)\log p(i,j) \tag{2.7}$$

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:

$$H(X,Y) = -\int_{\mathcal{X},\mathcal{Y}} f(x,y)\log f(x,y)\,dx\,dy \tag{2.8}$$

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:

$$I(X,Y) \triangleq H(X) + H(Y) - H(X,Y) \tag{2.9}$$

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 [8], data augmentation [110], and causality analysis [148]. 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 [102].

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 [134].

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 [102].

An estimate for the *mutual information* (MI) can be obtained by estimating $H(X)$, $H(Y)$, and $H(X, Y)$ separately and calculating Equation 2.9. 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 [102] 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 [79] 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 [133]: 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 [134]:

$$I(X,Y) - \mathbb{E}[\hat{I}_{MLE}(X,I)] \approx \frac{m_X + m_Y - m_{XY} - 1}{2N} \tag{2.10}$$

The KSG estimators, on the other hand, have a bias that stems from uniformity assumptions on the density and depends on neighbor distances [102]. In Section 5.3, we provide an experimental comparison of these estimators on multiple datasets.

2.4 Additional Related Work

Besides the work discussed above on join size estimation using k-minimum values (KMV) sketches, there is a large body of research that includes other approaches for cardinality estimation [86] as well as sampling and sketching algorithms used to handle massive data streams [46]. Also related to our work is recent research on efficient query processing for web search [153], dataset search [25, 173], and methods that support data augmentation queries which, given a query dataset, find datasets that can be joined or concatenated [177, 119, 178, 32, 166] or that contain related entities [106, 174]. In the remainder of this chapter, we provide an overview of each of these research areas.

Cardinality Estimation. Several methods have been proposed that summarize massive datasets into succinct data structures to support approximate queries using bounded memory [46]. A widely studied problem in this area is cardinality estimation, i.e., estimating the number of distinct elements in a set. Approaches to this problem can be

broadly classified into two groups: sampling and sketch-based algorithms. *Sampling algorithms* avoid scanning the full dataset and estimate cardinalities based on data samples. These algorithms have been criticized for their inability to provide good error guarantees [36, 82]. In contrast, *sketch-based algorithms* fully scan the dataset once, compute hashes of the items, and create a sketch that can be used to compute cardinality estimates [5, 16, 66, 44, 15, 49, 151, 45]. Harmouch and Naumann [86] categorized algorithms these into different families: *count trailing 1s*, *count leading 0s*, k^{th} *minimum values*, and *linear synopses*. Each of these families has their best algorithms which exhibit different trade-offs, which were studied in detail in [86].

While the best algorithms based on counting trailing 1s an 0s (such as HyperLogLog (HLL) [74]) are able to provide better accuracy per bit, it is not clear how they can be extended to estimate cardinalities of value-sets that satisfy arbitrary properties not known apriori, before the sketch is built. Algorithms from the k^{th} minimum values (KMV) family, on the other hand, can be extended for this purpose at the cost of storing the sample identifiers in addition to their hashed values [50]. The reason why HLL-like algorithms cannot estimate such properties is that, HLL does not maintain any sample of identifiers from the data. For this same reason, HLL sketches are not suitable for join-correlation sketches, which require the alignment of numeric values based on their join key values. CSK sketches builds on the KMV family and derives a new sketch that is able to align numeric values based on their key hashed values to reconstruct a random sample that can ultimately be used for estimating correlations. While in Chapter 4 we focus on numerical correlations and in Chapter 5 we focus on mutual information, the sketches we proposed can be used to compute other statistics that are based on paired numeric values.

Join Size Estimation. The ability to accurately estimate join sizes is crucial for query optimizers. Acharya et al. [1] have established very early results on the hardness of sampling over joins, in the general foreign-key setting, by showing that it is generally not possible to create a uniform random sample by simply joining uniform random samples from each independent relation. Since then, approaches have been developed to address this problem, which can be broadly grouped in the following categories: sampling, sketching, indexing, and machine learning.

Sketching-based algorithms [135] leverage the techniques mentioned above to create sketches on the join attribute and ignore all the other attributes. These algorithms usually provide accurate estimates of the join size on queries without selection predicates. In order to support queries with predicates, however, 2-way joins need to be transformed into 3-way joins by creating a additional sketch to represent the predicate [38], which substantially deteriorates the estimation quality [156].

Sampling-based algorithms keep a sample of the tuples, apply predicates on the sample to select the tuples that satisfy them, and finally estimate the join size using only these tuples. Multiple tuple-sampling strategies have been proposed [1, 70, 156, 38]. Recent works [156, 38] incorporate ideas similar to the strategies used in this dissertation and in the KMV sketches family: they use a random hashing function to map join values to the unit range and then select the tuples based on some selection strategy. For instance, the strategy adopted by the correlated sampling algorithm [156] is equivalent to the strategy of the G-KMV sketch [166], where tuples are selected if the hashed keys are smaller than a probability threshold. In contrast, CSK and TUPSK sketches include tuples in the sketch up to a fixed number, which avoids assigning too much space to large datasets and leads to more predictable performance for query evaluation.

Index-assisted algorithms [107, 109, 77] rely on indexes to perform the sampling. The use of indexes allows algorithms to retrieve only tuples that are relevant to the query, thus avoiding worst-case scenarios where no samples are available to perform estimations. A drawback of this approach is that indexes are not always available, and repeated index look-ups become expensive when the index does not fit into the main memory.

When a join involves keys of multiple tables and highly selective predicates, it becomes harder to estimate the join size because the join keys intersection gets increasingly smaller. Recently proposed approaches based on *machine-learning* [100, 167] are able to improve the estimation performance for highly-selective worst-case queries.

There are key differences between our work and approaches to join-size estimation algorithms both in terms of their goals and challenges. Notably, our sketches (CSK and TUPSK) do not need to deal with selection conditions. Moreover, CSK assumes that one-many as well as many-many relationships can be reduced to one-one joins (Section 4.2), allowing us to create uniform random samples of the resulting join table. TUPSK handles the many-one case, where repeated values in the right table can be aggregated but the repeated values on the left cannot.

Dataset Discovery and Search. 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 [141]. More closely related to our work are the approaches that retrieve datasets that are "joinable" with the query dataset. To measure joinability, the most common measure is the Jaccard Containment (JC) similarity, which is defined as $JC(X, Y) = |X \cap Y|/|X|$ where X is the set of values of the query table join attribute, and Y is the set of values of the retrieved table join attribute. While algorithms such as JOSIE [177] provide an exact solution to this problem, others such as LSH Ensemble [178],

GB-KMV [166] and Lazo [32] propose approximate approaches. Nargesian et al. [119] proposed a probabilistic solution to the problem of searching for unionable tables within massive data repositories. While these works focus on either finding datasets that are joinable or unionable, we focus on finding joinable tables that have strongly correlated columns.

The methods studies in this dissertation are complementary to the work of Kumar et. al. [103, 142], where they propose conservative decision rules to predict when the features obtained through a join can improve models. In contrast to our methods that aim to estimate statistics over joined tables without materializing the join, they derive a simple rule that predicts whether it is safe to not perform the join at all using only statistics derived from the join key on the base table.

Applications of these methods include decision support, data mining, ML model improvement, and causality analysis, and have resulted in several end-to-end systems. For instance, Data Civilizer [58] is a system that uses a linkage graph to help identification of relevant data to a user task. JUNEAU [176] formalized multiple data search tasks, table relatedness measures (e.g., column and row overlap, provenance, textual similarity), and proposed linear combinations of these measures for providing data search within data science environments. ARDA [39] is a system that focuses on automatic data augmentation, i.e., how to select the best features discovered from external dataset search systems. In Chapters 4, 5, and 6 we address an orthogonal problem: instead of building an end-to-end system, we focus on *efficient* data discovery and estimation of data relationships, such as correlation and mutual information, over joins. Our techniques can be integrated as relationships in linkage graphs [58, 73] or as a table relatedness measure [176] to improve search applications.

Feature Selection. There are three main classes of feature selection methods (see [157] for a survey): filter methods, which start from a join over all tables and use a lightweight proxy 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 dissertation is related to filter methods since it enables the estimation of correlations and MI, which are proxy measures commonly used in several feature selection algorithms [157, 84, 28].

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 sub-sampling [159, 37], which can be extended to MI. As we showed in Section 5.2, our methods compute the entropy over a subsample recovered from sketches to estimate the MI between two variables.

Some approaches considered MI estimation over data streams [94, 98, 22]. Ferdosi et. al. [72] 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. 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 2.3. Additionally, recent work has shown that no estimator can *guarantee* an accurate estimate of mutual information without making strong assumptions on the population distribution [116]. While previous work had provided

intractability results for specific estimators, such as KSG [78], this result is universal to all MI estimators.

Correlation Estimation. Correlation measures have been studied extensively in the literature (Section 2.2). Most works in this area focus on statistical inference, i.e., given samples of a potentially infinite population, the goal is to infer properties by deriving estimates or performing statistical tests. Recent work has focused on reducing bias and estimation errors [18], hypothesis testing [17, 57], or confidence intervals (CI) that work well under non-normal distributions [20, 19, 91]. Differently from existing CIs, we develop a CI for sub-samples based on concentration inequalities that make no distributional assumptions. This setting is particularly attractive for sketching algorithms that perform a single pass over the data, allowing us to leverage statistics about data (e.g., the range of the values) that would otherwise be unknown.

Web Tables and Ad-hoc Table Search. Another related line of work focuses on discovering and performing automatic extensions of web tables that contain entities in textual form [174, 106]. Lehmberd et. al. [106] propose an engine that searches an indexed corpus for additional data describing the entities contained in a user-provided dataset, and automatically extends its input table with the discovered information. Zhang et al. [173, 175] formalize the problem of ad-hoc table retrieval using semantic similarity, and propose machine-learning methods for addressing the problem. While these works retrieve semantically similar tables that contain entities, our focus is on retrieving tables that are numerically related, such as datasets that contain columns that are highly correlated with a column in the input query dataset.

Search Engine Architectures and Fast Top-k Retrieval. Our work builds upon prior work on efficient query processing algorithms for top-k document retrieval [26, 60, 154, 146]. We refer the reader to [153] for a comprehensive survey on the topic.

While these techniques have been traditionally used for retrieving textual documents for scalable web search, we extend them with data sketching methods to efficiently retrieve correlated tables. In particular, we use an implementation of the Block-Max WAND [60] dynamic pruning algorithm for fast retrieval of sketches. State-of-the-art top-k query processing algorithms rely on dynamic pruning algorithms that require a property known as *non-negative monotonicity*, which means that the scores computed for each term in the index cannot be negative [153, 71]. As we discuss in Section 6.2.1, this property prevents the use of these algorithms for correlated table retrieval. In Chapter 6, we show that by decomposing the retrieval into two smaller queries, we can leverage these algorithmic optimizations. Our methods also use a technique that has been described as cascading [153]. More specifically, our proposed hashing method allows for the efficient retrieval of correlated columns (with a high recall), which can then be passed onto another cascading layer that re-ranks candidates using sketches to improve the overall ranking.

Chapter 3

Auctus: A Search Engine for Data Discovery and Augmentation

With the push towards transparency and openness, scientists, governments, and companies have been increasingly publishing structured data on the Web. Google Dataset Search alone indexes over 30 million datasets [7]. The availability of these data creates opportunities to answer many important scientific, societal, and business questions.

The Need for Data Discovery. While data are abundant, finding *relevant data* is difficult. Data are spread over a large number of sites and repositories. Recognizing this challenge, a number of approaches have been proposed to *organize and index data collections* [35], from domain-specific repositories such as NYCOpenData, which collects datasets from the various NYC agencies [121], general data portal infrastructure and data lakes [147, 54], to Google Dataset Search, which indexes a wide range of datasets published on the Web [25]. While these present a significant step towards simplifying data discovery, they have an important limitation: they only support simple, keyword-based search queries over published dataset metadata. This greatly limits a user's ability

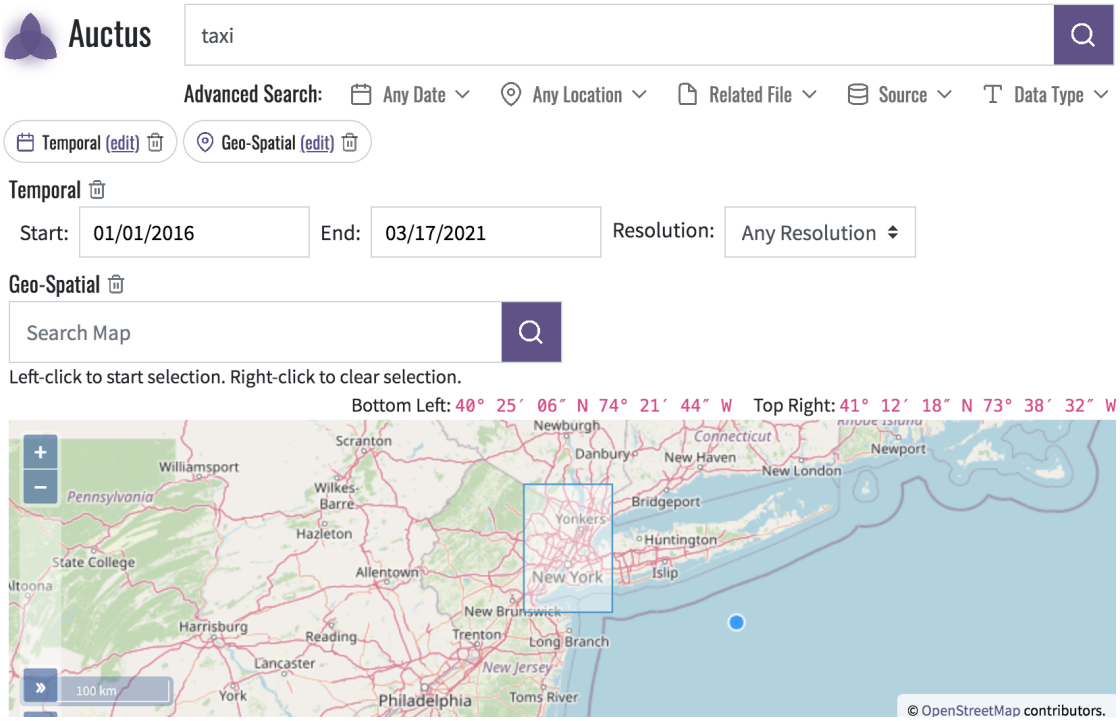


Figure 3.1: Searching for datasets that mention *"taxi"* and contain records within the NYC area for the 2016-2021 period.

to express information needs. In addition, published metadata is often incomplete, and in some cases it can be inconsistent with the actual data. Thus, relying solely on the metadata also limits the discoverability of datasets.

Chapter Contributions. In this chapter, we introduce `Auctus`, an *open-source dataset search engine that was designed to support data discovery and augmentation.* It supports a rich set of discovery queries: in addition to keyword-based search, users can specify spatial and temporal queries, data integration queries (i.e., searching for datasets that can be concatenated to or joined with a query dataset), and they can also pose complex queries that combine multiple constraints, as shown in Figure 3.1. These queries are enabled in part by a data profiler [53] that we developed to extract useful information from the actual datasets. This includes not only summaries (or sketches) of column

contents, but also their data types. In particular, it detects columns that contain spatial and temporal information. This information is then used to construct indices that support efficient query evaluation. Users can explore large dataset collections through an easy-to-use interface that guides them in the process of specifying complex queries. To help users identify relevant datasets, Auctus displays snippets that summarize the contents of datasets.

Auctus has been developed in the context of the DARPA D3M program [51], and it has been used in production and to support research of different groups in the project [138, 90, 39, 11]. In this chapter, we give an overview of the architecture and features of Auctus (Section 3.1) and discuss a few use cases that it enables (Section 3.2).

3.1 The Auctus System

3.1.1 Auctus Architecture

The high-level architecture of Auctus is depicted in Figure 3.2. In what follows, we describe its key components.

Data Ingestion. Auctus makes use of plugins to retrieve datasets from repositories using their APIs. This makes the system extensible and able to ingest data from many different sources. Currently, it supports Socrata [147] (a platform for open government data), Zenodo [172] (open-access data repository), the World Bank Open Data [163] (datasets on global development), among others. Auctus also allows users to upload their own datasets.

Profiling. Once datasets are ingested, Auctus profiles them to infer relevant metadata necessary to support discovery queries as well as to construct dataset summaries for

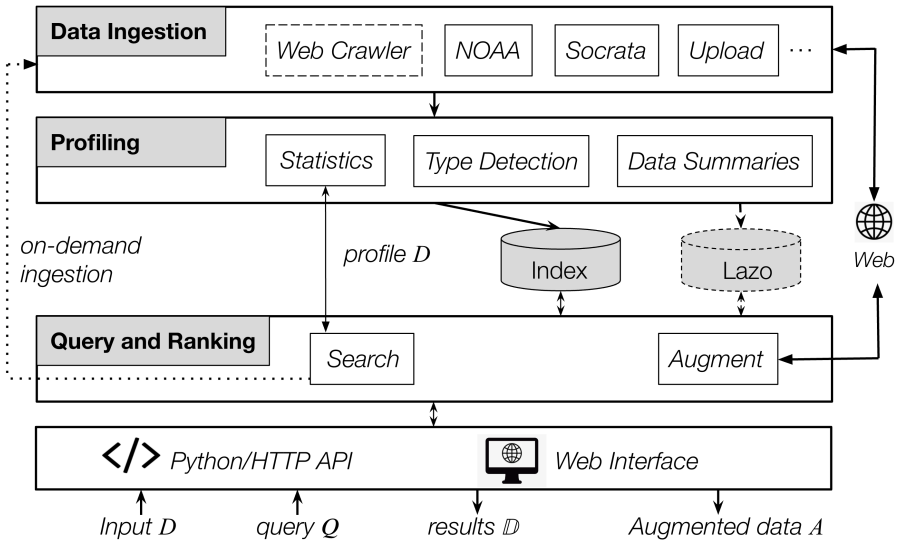


Figure 3.2: Overview of Auctus' architecture.

result presentation. The profiler performs different tasks, including: *type detection*, i.e., it detects whether columns correspond to categorical, numerical, spatial, or temporal attributes; type-dependent *statistics computation* (e.g., frequency of values, mean, and variance for numerical values), and *data summarization* (see below).

Storing Data and Data Summaries. Since Auctus was designed to serve as a dataset discovery system, it stores dataset summaries instead of the full datasets. These summaries are concise and sufficient to create the indices required to answer all queries supported by the system. Currently, Auctus creates summaries for categorical, numerical, spatial, and temporal attributes. The data summaries generated by Auctus are represented by the ranges of their corresponding attributes. During the search phase, Auctus uses these summaries to estimate the size of the intersection between two attributes, concluding whether a join is feasible. To estimate join intersections well, Auctus captures fine-grained ranges for data attributes by using the clustering algorithm k-means. This strategy produces the desired results while also being efficient. Auctus also stores provenance information to enable the retrieval of the datasets.

This allows the system to perform augmentations, and users to download the datasets. Auctus can also cache datasets for efficiency purposes.

Indices. After the metadata are generated, including data summaries, they are indexed in an Elasticsearch [68] server. Numerical and temporal summaries are indexed using range data types, and spatial summaries are indexed using geo-shape data types. To support joinable dataset search, we use Lazo [32], which is a method for set-overlap search based on MinHash sketches and locality-sensitive hashing (LSH), to build an index for categorical attributes.

Querying and Ranking. Auctus supports queries that combine multiple constraints including keywords, temporal, spatial, data type, and data source. The system also supports *data integration queries*: Given an input dataset D_Q, Auctus allows the user to search for datasets that can be concatenated to or joined with D_Q. The dataset D_Q must first be uploaded using a provided API or selected from the set of indexed datasets. The system will then generate a dataset profile which is used to probe those indices that support join and union. Finally, the lists of matching datasets from different indices are merged, ranked, and returned as search results.

Join Search. To find other datasets that can be joined with D, Auctus first searches, for each attribute a of D, which other attributes, in the index that corresponds to a's data type (e.g., temporal attributes searched in the Elasticsearch range index, or categorical attributes probed against the Lazo index), have summaries intersecting the summary of a. Every dataset that has at least one intersecting attribute is a potential join result.

Union Search. To find datasets that can be concatenated with D, the indices are searched for any dataset that has attributes with the same data types present in D, as well as

similar names. Name similarity is computed with the *fuzzy query* in Elasticsearch. Results from union searches may match only a subset of D's attributes.

Ranking. The results of join and union searches, after being filtered based on query Q, are ranked and returned as $\mathbb{D}$. Join results are ranked based on the intersection of the summaries; union results are ranked based on the Levenshtein similarity between the names of the matching attributes.

Augmentation. Besides providing search capabilities for data augmentation, `Auctus` also performs the actual augmentation. Users can choose a dataset $R \in \mathbb{D}$, and `Auctus` will materialize it (using the provenance annotated in the metadata) and perform the join or union operation with D, returning the new, augmented dataset A. If multiple attribute pairs match between R and D for a join operation, users can choose which pair(s) they want for the join. For temporal and spatial joins, the attributes are translated into the same resolution before the augmentation.

3.1.2 Auctus User Interface

`Auctus` provides an easy-to-use interface where users can query for datasets, explore search results including data exploration and augmentation options, explore ingested datasets, and upload new datasets.

Data Discovery Queries. Users can query indices by specifying keywords and constraints using various filters (see Figure 3.3(A)).

Temporal and Spatial Search. Users search for datasets by specifying a date range (Figure 3.1) – datasets containing a temporal column that overlaps with that range will be retrieved. They can refine the search by specifying a temporal resolution (e.g., year). To perform a spatial search, users can either draw a bounding box around a geographical area on the map (Figure 3.1), or specify an administrative area, which

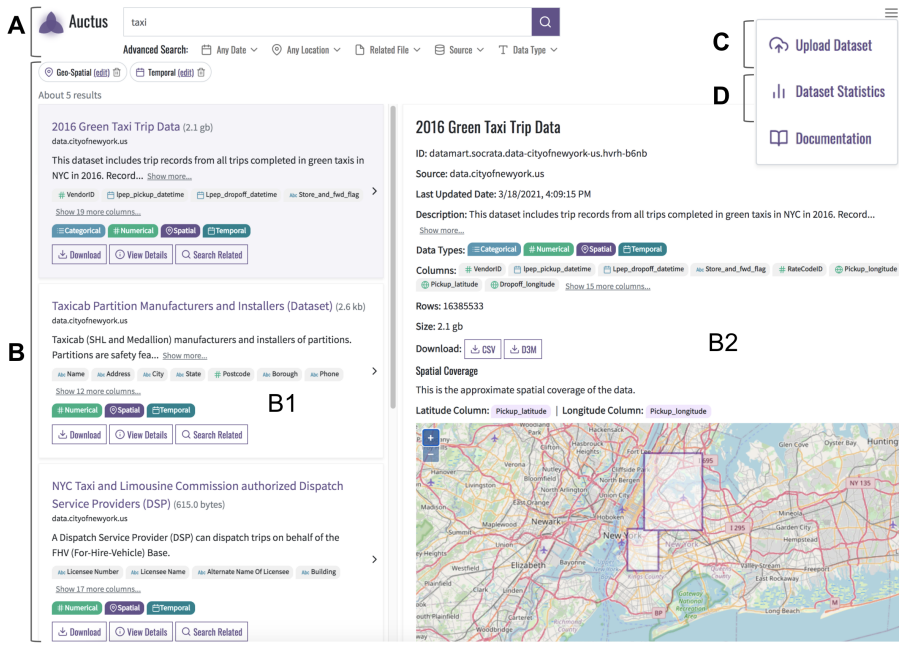


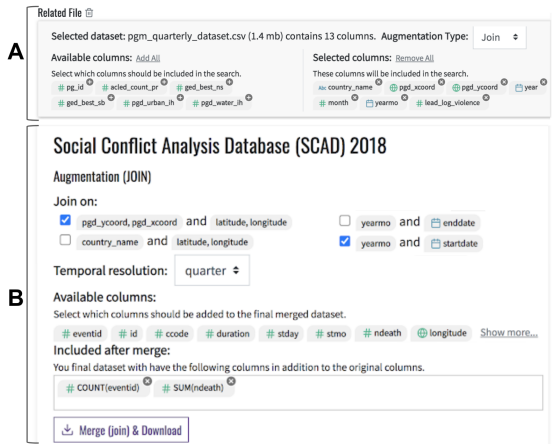
Figure 3.3: Components of Auctus' user interface: keyword and filter-based search box **(A)**; search results **(B)**; dataset-snippets **(B1)**; dataset summary **(B2)**; dataset upload **(C)**; dataset collection statistics **(D)**.

Auctus translates into a polygon. Datasets containing spatial attributes that overlap with the search polygon are returned.

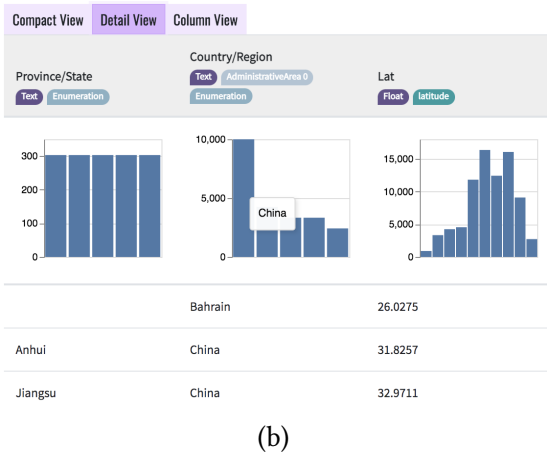
Source filter. The source filter allows users to restrict the data sources of interest, and only results from the selected sources are retrieved.

Data Type filter. The data type filter allows users to search for datasets based on the types of their attributes, e.g., categorical, numerical, spatial and temporal, which were inferred by the profiler.

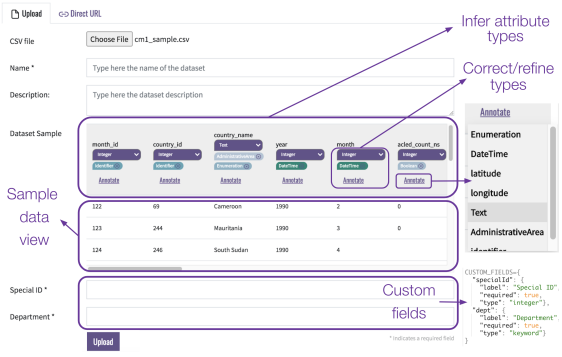
Data Integration Queries. The related file filter allows users to find datasets that can augment a given input dataset (see Figure 3.4a(A)). The user can upload the input dataset or select a dataset from a set of search results. Result snippets for data integration queries include an "augment options" button, that allows users to request and customize the augmentation to be performed by Auctus.



(a)



(b)



(c)

Figure 3.4: (a) A data integration query in [A], and the augmentation interface in [B]. (b) Data Summary views. (c) Upload page.

Result Presentation and Exploration. Unlike Web documents which can be summarized using short text snippets, datasets have many different facets that the user must consider to determine their relevance. Thus, an important challenge for Auctus is how to present search results. Figure 3.3 shows the results for the query specified in Figure 3.1. On the left, there is a list of search results displayed as snippets (Figure 3.3(B1)). Users can select a dataset to inspect its details (Figure 3.3(B2)), which include description, source, attribute names and types, as well as a summary of the dataset's contents. For spatial datasets, a visualization showing the geographical extent covered by the dataset is displayed. The summary also includes a sample of the dataset records and statistics about its columns (Figure 3.4(b)). The tabs above the dataset sample allow for the visualization of these statistics in different levels of detail. For example, *Detail View* shows the columns' value distributions.

Augmentation. If the user provides an input file through the Related File filter, the snippets display a button "Augment Options" under each result on the left. If the user clicks on this button, the augmentation interface will be shown (see Figure 3.4a(B)). If the result is a union, users can select matched pairs of columns, and the columns from the result (right) will be appended to the matching columns of the input data (left). If the result is a join (an example is shown in Figure 3.4a(B)), users can select the columns to be matched at the top of the augmentation screen. Those are the "join keys" that determine which records from the input data should be matched with records from the selected result. Under this area, there is an interface to include columns from the selected result into the augmented table. Users can grab the desired columns from the "available columns" area, drag them to the "included after merge" box, and drop them over the aggregation function they wish to use for that column.

Uploading Data and Curating Metadata. Users can add new datasets to Auctus. After loading a data file, Auctus automatically infers its data types with our datamart-profiler library [53]. As any method for type inference, our profiler is not fool-proof and can derive incorrect results. To address that, Auctus enables users to correct data types manually, and to provide additional annotations for the columns. Auctus also displays a dataset sample so that the user can verify if the detected data types are correct, and check the uploaded data. Additionally, Auctus provides support for custom metadata fields (e.g., data source or grid size). Since these fields can vary widely for different applications, we defined a flexible configuration schema that allows users to customize them for different deployments. The upload page is shown in Figure 3.4c.

3.1.3 Scalability and APIs

Auctus has been implemented with scalability in mind: the search engine is entirely containerized using Docker [61]. Each data discovery plugin is an independent container, allowing multiple plugins to be executed in parallel. Auctus can also spin up as many profiling and query containers as required in response to load. Our system can be accessed via a Web UI or programmatically via Python and REST APIs. So far, we have indexed over 19,000 datasets.

3.2 Use Cases

Bicycle Usage Prediction. Predicting the number of daily bicycle trips is an important step towards implementing better policies for this means of transportation in NYC. Consider that an expert from the NYC department of transportation decides to build a model using the East River Bicycle Trips data. This department installed automated

counters in all East River bridges, which provide the number of bikes crossing them on a daily basis. This number can be used as a proxy for the overall bicycle usage in NYC. Besides daily bicycle counts, the data also contains maximum daily temperatures for NYC. Unfortunately, her initial dataset only covers bicycle usage for April 2020, which is not very informative. To take into account a larger period of time, she uses `Auctus` to find compatible data for more months of 2020, which can then be concatenated to her original input data within our system. When using a random forest regressor to predict the number of bicycle trips using daily temperatures as a model feature, she obtains an R^2 score close to 0.25. She is relying on a single feature to make her predictions, so her model is probably underfitting. To improve it, she uses `Auctus` again to find and augment her current input data with weather information, including daily rainfall levels. The R^2 score then increases to approximately 0.56, which represents a substantial improvement. A video demonstrating this use case is available at `http://bit.ly/auctus-demo`.

Conflict Forecasting. While analyzing conflict forecasting problems, researchers often use predictive modeling to guide policy-making decisions, and to assess and compare theories of conflict [64]. In this context, it is crucial to identify new data sources, merge those data, and evaluate the contribution of different features. Furthermore, most conflicts materialize as events, and if they occur in heterogenous spatio-temporal levels and need to be analyzed in tandem, data integration can be challenging. Consider that a researcher is studying conflict events in Africa. She uses the grid [128] dataset for Africa with conflict events aggregated into quarterly counts to predict state-based conflicts. Since protest events data can be very useful, and since her grid dataset does not have any measure of protests, she decides to use `Auctus` to discover and join this type of dataset to her input data. She uses "protest" as a keyword, and also uploads the grid dataset as an input query in `Auctus`. After running the query, the highest

ranked result is the Social Conflict Analysis Database (SCAD). To verify its suitability, she explores the dataset through the "View Detail" tab. After inspecting the Summary Data view, she realizes that SCAD contains information not only on protests, but also on other destabilization events. She can also quickly see that it captures events in Africa because of the map visualizations, and also finds spatio-temporal matches. She then checks the possible augmentation options moving in to the augmentation interface. As shown in in Figure 3.4a(B), `Auctus` automatically detects joinable columns. She then selects the temporal and spatial levels of her interest, and uses aggregation functions to count the number of destabilizing events and to sum the number of fatalities. Next, she presses the augmentation button and all the events in SCAD become aggregated into the grid dataset. The augmented dataset can be used to explore new research questions, such as whether the number of destabilizing events at time *t* is a predictor for state-based violence at time *t+1*.

Chapter 4

Evaluating Join-Correlation Queries using Sketches

In this chapter, we introduce *join-correlation queries*: a new class of data search queries that *find tables that can be joined with and that also contain attributes that are correlated with those of a given query table.* They are useful for uncovering correlations between numerical columns in unjoined datasets (such as in discovery tasks introduced in Examples 1 and 2 from Chapter 1). We recall the definition of Join-Correlation queries introduced in Chapter 1:

Definition 1 (Join-Correlation Query)**.** *Given a column Q and a join column K_Q from a query table $\mathcal{T}_Q$, a* join-correlation query *finds tables $\mathcal{T}_X$ in a dataset collection such that $\mathcal{T}_X$ is joinable with $\mathcal{T}_Q$ on K_Q and there is a column $C \in \mathcal{T}_X$ such that Q is strongly correlated with C.*

One naïve approach to answer this query is to first identify all joinable tables, and then explicitly calculate correlations between Q and all columns of the discovered tables. The main bottleneck in this approach is downloading the joinable datasets and

computing their joins with the query table. Large tables may not fit in memory, and the cost of executing join operations can be prohibitively high.

Additionally, some correlation measures, such as Spearman's correlation, require sorting the data before computing the correlation. This problem is compounded for queries that return multiple datasets and require numerous joins and correlation computations. For reference, joining a dataset containing taxi pickups (about 1GB) with a precipitation dataset (about 3MB) took about 29 seconds (on an Intel Core i5 2.4GHz CPU). After the join, it took about 5 seconds to calculate the Spearman's coefficient between the number of pickups and precipitation levels.

While previous research has proposed efficient methods to support queries that retrieve datasets that can be joined with a given query dataset [178, 177, 32, 166], these methods alone are not sufficient to support join-correlation queries since joining tables is still an expensive operation. Next, we describe a more efficient approach for evaluating this type of query.

4.1 Ranking Datasets via Correlation Estimates

To reduce the evaluation cost of join-correlation queries, as an alternative to using the entire data, we investigate the use of data sketches (also known as *synopses*) to estimate the results. Data-intensive algorithms often can be optimized by reducing the size of the input data with sampling techniques [46], at the cost of obtaining approximate results. In our setting, however, naïvely sampling data to estimate correlations does not work: it is not possible to sample columns to estimate the correlation without first executing the join. To solve this problem, we propose a sketching method for *estimating correlations between columns from unjoined datasets* based on data sketches.

These sketches are constructed using only data from individual tables (independently), and thus they can be *pre-computed and indexed* to support the discovery of joinable datasets and *fast correlation estimation* at query time. In this dissertation, we refer to such sketches generally as "correlation sketches". In the remainder of this chapter, we propose a particular method of building correlation sketches that we refer to as *CSK*.

The CSK method constructs a sketch $\mathcal{S}_X$ for any given pair of columns $\langle K_X, X \rangle$ that belongs to a table $\mathcal{T}_X$, where K_X is a categorical column and X is a numerical column. A pair of sketches $\mathcal{S}_X$ and $\mathcal{S}_Y$ (for tables $\mathcal{T}_X$ and $\mathcal{T}_Y$ respectively) can be used to estimate the correlation between the numerical columns $X_{X \bowtie Y}$ and $Y_{X \bowtie Y}$, generated *after* joining tables $\mathcal{T}_X$ and $\mathcal{T}_Y$ on columns K_X and K_Y. Note that $\mathcal{T}_X$ and $\mathcal{T}_Y$ are heterogeneous and need not have the same join keys or the same number of rows. As such, our sketching method enables the construction of an index for a large number of tables that can be used to support *both* joinability queries and to estimate the correlation between a query column and indexed columns.

The CSK method builds upon and extends state-of-the-art hashing techniques [5, 15, 16, 166, 92]. In Section 4.2, we prove that our sketches can reconstruct a uniform random sample of the paired columns $X_{X \bowtie Y}$ and $Y_{X \bowtie Y}$. This leads to an important property: besides correlations, our sketching approach can handle *any* statistic that can be estimated from random samples (e.g., entropy and mutual information). While we focus on numerical correlation measures in this chapter, we extend this algorithm in Chapter 5 to support different data types and measures such as Mutual Information.

We show both theoretically (Section 4.3.3) and experimentally (Section 4.4) that our approach is effective and provides accurate estimates for correlation. Moreover, our analysis provides mathematical tools for dealing with approximation errors typical of sketching algorithms. For join-correlation queries, these errors may lead to false

positives: columns are returned which seem more correlated, based on the sketch, than they actually are. This problem is an issue for queries over large dataset collections, as there can be many false positives, simply by chance. To address this issue, we derive sub-sample confidence interval bounds to estimate approximation errors. Our bounds are based on simple column statistics like sample size and data range and, in contrast to prior work [20, 19, 10], do not rely on distributional assumptions. We use these bounds to design a set of scoring functions that rank datasets based on both their estimated correlation with a query dataset, and on our confidence in that estimate.

In Section 4.4, we evaluate our method experimentally using both synthetic and real-world datasets. A comparison of the estimates produced by CSK sketches with the actual correlation values shows that it derives accurate estimates. In addition, we assess the effectiveness of different ranking functions that leverage CSK sketches, and show that they improve ranking performance up to 193% in terms of mean average precision when compared to a scoring scheme based on overlap size, commonly used for joinability queries [32, 178, 177].

Chapter Contributions. We introduce *join-correlation queries*, a new class of queries to find correlated columns within a collection of unjoined tables, and propose a new method to efficiently support such queries over large dataset collections. To the best of our knowledge, ours is the first work that addresses this problem. The remainder of this chapter is organized as follows:

- In Section 4.2, we propose CSK sketches, a new sketch that simultaneously summarizes information about joinability and correlation, allowing the estimation of different correlation measures between columns of unjoined datasets.

- In Section 4.3, we derive new correlation confidence interval bounds that allow us to measure the risk of estimation errors. These bounds serve as the basis for the design of scoring functions that use correlation sketches to rank the discovered columns.

- We perform an extensive experimental evaluation of our method and show that: CSK sketches estimate correlations with good accuracy in both synthetic and real data for different correlation measures; and the scoring functions we propose are effective and derive high-quality rankings (Section 4.4).

4.2 Estimating Join-Correlation

Before presenting CSK sketches, we introduce notation and formally define the join-correlation estimation problem. Consider a query table $\mathcal{T}_X$ composed of a categorical column K_X and a numerical column X, and a table $\mathcal{T}_Y$ in a dataset collection containing a categorical column K_Y and a numerical column Y (we discuss below how multi-column tables are handled). Columns K_X and K_Y are the join attributes, i.e., $\mathcal{T}_{X \bowtie Y} = \pi_{k,x_k,y_k}(\mathcal{T}_X \bowtie_{K_X = K_Y} \mathcal{T}_Y) = \{\langle k, x_k, y_k \rangle : k \in K_X \cap K_Y\}$. We denote by x_k and y_k the numerical values of X and Y (respectively) associated with the row identified by key k. This is illustrated in Figure 4.1.

Definition 2 (Join-Correlation Estimation). *Given two tables $\mathcal{T}_X$ and $\mathcal{T}_Y$, we aim to efficiently estimate the correlation $r_{X \bowtie Y}$ of the numerical attributes $X_{X \bowtie Y}$ and $Y_{X \bowtie Y}$ in $\mathcal{T}_{X \bowtie Y}$ without having to compute the join and aggregations for $\mathcal{T}_X$ and $\mathcal{T}_Y$.*

One approach to estimate $r_{X \bowtie Y}$ is to use data sketches instead of the full datasets. To do so, we can build sketches $\mathcal{S}_X$ and $\mathcal{S}_Y$ that serve as summaries of $\langle K_X, X \rangle$ and $\langle K_Y, Y \rangle$ respectively. However, naïve approaches do not yield useful summaries.

$\mathcal{T}_X$

K_X	X
2021-01	6.0
2021-02	4.0
2021-03	2.0
2021-04	3.0
2021-05	0.5
2021-06	4.0
2021-07	2.0

$\mathcal{T}_Y$

K_Y	Y
2021-01	5.5
2021-01	4.5
2021-02	3.9
2021-02	2.0
2021-03	4.0
2021-03	1.0
2021-04	4.0

$\mathcal{T}_{X \bowtie Y}$

$K_{X \bowtie Y}$	$X_{X \bowtie Y}$	$Y_{X \bowtie Y}$
2021-04	3.0	4.0
2021-03	2.0	2.5
2021-02	4.0	3.0
2021-01	6.0	5.0

Figure 4.1: Table $\mathcal{T}_{X \bowtie Y}$ is the join of the input tables $\mathcal{T}_X$ and $\mathcal{T}_Y$, aggregated using the `mean` function. Correlation sketches efficiently reconstruct a sample of the table $\mathcal{T}_{X \bowtie Y}$ to estimate the correlation between the columns $X_{X \bowtie Y}$ and $X_{X \bowtie Y}$, without computing the full join.

Limitations of Random Sampling. Consider, for example, two numerical vectors of size n, S_X and S_Y, randomly sampled from the numerical columns $X \in \mathcal{T}_X$ and $Y \in \mathcal{T}_Y$, respectively. The correlation between S_X and S_Y is not a valid estimate of the correlation between $X_{X \bowtie Y}$ and $Y_{X \bowtie Y}$ because the pairs $\langle x \in S_X, y \in S_Y \rangle$ are not aligned. By sampling directly from X and Y, we lose information on what keys are associated with what numerical values, which is necessary to align pairs $\langle x_k, y_k \rangle$. Another alternative would be to include the keys by randomly sampling rows from original tables $\mathcal{T}_X$ and $\mathcal{T}_Y$, and then joining the row samples. However, since the final set of keys $k \in K_{X \bowtie Y}$ depends on input columns $K_X \in \mathcal{T}_X$ and $K_Y \in \mathcal{T}_Y$, it is unlikely that the keys selected from K_X *and* contained in $K_{X \bowtie Y}$ will also be selected from K_Y. We discuss this issue more formally in the next subsection.

4.2.1 Correlation Sketches (the CSK method)

Correlation Sketches address the limitations of naïve approaches by *enabling the reconstruction of a uniform random sample of the joined table.* Our method uses hashing techniques to carefully select a small sample of tuples $\langle k, x_k \rangle$ from a table $\mathcal{T}_X =$

$\mathcal{S}_X$

$h(k)$	$h_u(k)$	x_k
bac52e98	0.48	2.0
16dab449	0.34	2.0
26f79756	0.47	3.0
4da33cf5	0.34	6.0

$\mathcal{S}_Y$

$h(k)$	$h_u(k)$	y_k
16dab449	0.34	2.5
bd5a7c1f	0.89	3.0
26f79756	0.47	4.0
4da33cf5	0.34	5.0

$\mathcal{S}_{X \bowtie Y}$

$h(k)$	x_k	y_k
16dab449	2.0	2.5
26f79756	3.0	4.0
4da33cf5	6.0	5.0

Figure 4.2: The tables $\mathcal{S}_X$ and $\mathcal{S}_Y$ represent correlation sketches for the tables $\mathcal{T}_X$ and $\mathcal{T}_Y$, for sketch size $n = 3$ and `mean` aggregation. While we explicitly show the column $h_u(k)$ for illustrative purposes, it does not need to be stored as it can be easily computed from $h(k)$.

$\langle K_X, X \rangle$, that is used to build a sketch that enables data from different tables to be aligned and correlations to be estimated.

Sketch Construction. We use two different hashing functions to create the sketch $\mathcal{S}_X$ for table $\mathcal{T}_X$. The first one, h, is a collision-free hash function that randomly and uniformly maps key values $k \in K_X$ into distinct integers. Given that these integers $h(k)$ are unique, they are used as the tuple identifiers in sketch $\mathcal{S}_X$. Next, we use hashing function h_u to map integers $h(k)$ to real numbers in the range $[0, 1]$, uniformly at random. The function h_u plays a key role in the selection of the n tuples that compose the sketch $\mathcal{S}_X$: the tuples that correspond to the n smallest h_u values are the ones included in the sketch. More formally, we select n samples of pairs $\langle h(k), x_k \rangle$ with minimum values of $h_u(k)$, i.e., $\mathcal{S}_X = \{\langle h(k), x_k \rangle : k \in min(k, h_u(k))\}$, where min is a function that returns a set containing the keys k with the n smallest values of $h_u(k)$. To illustrate this, consider the example table $\mathcal{T}_X$ in Figure 4.1. To build a correlation sketch, we apply the hashing functions to each one of the keys $k \in K_X$ and then select the n tuples associated with the smallest hashed values, which are used to create the *correlation sketch* $\mathcal{S}_X$ in Figure 4.2.

Once these sketches are created (independently) for separate tables, they are used to estimate correlation by computing a *joined sketch* $\mathcal{S}_{X\bowtie Y}$, also illustrated in Figure 4.2. $\mathcal{S}_{X\bowtie Y}$ has a row for every key k that appears in both $\mathcal{S}_X$ and $\mathcal{S}_Y$. As we argue in Theorem 1, this table contains a uniform random sample of paired numerical values from $\mathcal{T}_{X\bowtie Y}$, so it can be used directly to estimate correlation or any other statistic over $\mathcal{T}_{X\bowtie Y}$.

The accuracy of an estimate, however, depends on the size of this sample being large, i.e., on $\mathcal{S}_{X\bowtie Y}$ having many rows. The key idea behind our method is that, by selecting samples using h_u, we introduce dependence that increases the probability of $\mathcal{S}_X$ and $\mathcal{S}_Y$ including the same keys [92]. To see this, consider an extreme example where both K_X and K_Y have the same set of N distinct keys. Suppose that n key-value tuples $\langle k, x_k\rangle$ are included in $\mathcal{S}_X$ uniformly at random (without hashing), and n are also included in $\mathcal{S}_Y$, independently and uniformly at random. If a key k is included in $\mathcal{S}_X$, the probability that it also appears in $\mathcal{S}_Y$ is n/N. Thus, the expected number of rows in $\mathcal{S}_{X\bowtie Y}$ is n^2/N, vanishingly small when the sketch size n is much smaller than N. With little or no key overlap, we have no way of effectively estimating correlation from $\mathcal{S}_{X\bowtie Y}$. On the other hand, when the inclusion is determined by the values of $h_u(k)$, the events become dependent: if k is included in $\mathcal{S}_X$, k must also be included in $\mathcal{S}_Y$. In this case, the number of rows in $\mathcal{S}_{X\bowtie Y}$ increases to n, the maximum number possible. In a less extreme case where K_X and K_Y do not have the exact same keys, the expected number of keys included in both sketches will depend on the Jaccard similarity between the key sets, but in any case, will be much larger than the n^2/N obtained by uniform random sampling.

Note that the use of a hashing function such as h_u to introduce dependence is not a new idea — it has been used in many algorithms [15, 156, 50, 92, 166]. Part of

our contribution lies in the combination of h_u with another function h to generate tuple identifiers that *allow the alignment of paired data samples at estimation time*. CSK sketches contains both the n minimum hashed values $h(k)$ and their corresponding numerical values x_k from column X. By keeping the hash of the key, it is possible to align the numerical values with values in other tables that are associated with the same key, and by storing the numerical values, we can estimate the correlations between the numeric columns.

Handling Repeated Keys. The process described above assumes that the keys uniquely identify each row in a table. However, real-world data often contain repeated categorical values (as in column K_Y in Figure 4.1). In such cases, there is a set of values associated with each distinct key k. For instance, in Figure 4.1, the set of values $\{5.5, 4.5\}$ is associated with the key "`2021-01`". Because correlation is only defined for sets of paired values, downstream applications that use correlation typically aggregate the numeric values associated with a key into a single number. This can be done by applying a user-defined function (e.g., *mean*, *sum*, *maximum*, *minimum*, *first*, *last*) to perform the aggregation before computing the correlation. In Figure 4.1, the column $Y_{X \bowtie Y}$ contains the aggregated values of Y using the *mean* function after the join between $\mathcal{T}_X$ and $\mathcal{T}_Y$.

Repeated keys can be handled during sketch construction: whenever a key k that already exists in the set of hashed minimum values is found again at time t, an aggregate function f can be applied to compute the value for time t by aggregating the existing x_k^{t-1} with the new incoming x_k, i.e., $x_k^t = f(x_k, x_k^{t-1})$. As long as the aggregation can be computed in a streaming fashion, the sketches can also be computed with a single pass over the data. $\mathcal{S}_Y$ in Figure 4.2 shows a sketch constructed from table $\mathcal{T}_Y$ in Figure 4.1, using *mean* as the aggregate function f.

Note that the choice of function affects the semantics of the data, thus this selection must be made by taking into account the requirements of the downstream application that makes use of the sketches. Nonetheless, our sketch is agnostic to such aggregations, and can easily be extended to take as input one or more functions. Additionally, note that performing aggregation over repeated keys in one of the tables may not be desirable in some situations. For example, when performing left outer joins to bring in new features for training machine learning models, the number of rows in the left table must kept intact. In Chapter 5 we describe another sketch that can better handle these situations.

Sketches for Multi-Column Tables. For simplicity, we described how to build sketches from a binary table, but it is trivial to extend the process to multi-column tables. For example, if a table contains multiple columns, $\mathcal{T}_{XZ} = \{K_{XZ}, X, Z\}$, the correlation sketch could be extended to $\mathcal{S}_{X,Z} = \{\langle h(k), x_k, z_k\rangle : k \in min(k, h_u(k))\}$. Alternatively, one could simply build one sketch for each pair of keys and numeric columns, e.g., $\langle K_X, X\rangle$ and $\langle K_Z, Z\rangle$.

4.2.2 Estimating Join-Correlation

An important property of CSK sketches is that it enables the construction of a *uniform random sample* of the join $\mathcal{T}_{X\bowtie Y}$. This can be formally stated as:

Theorem 1. *The set of paired numeric values $\langle x_k, y_k\rangle \in \mathcal{S}_{X\bowtie Y}$ is a uniform random sample of the set of paired numeric values $\langle x_k, y_k\rangle \in \mathcal{T}_{X\bowtie Y}$.*

Common measures of correlation, such as Pearson and Spearman, can be approximated with a sub-sample of data from those columns, as long as that sample is taken uniformly at random. Theorem 1 forms the basis of our algorithm for join-correlation estimation, which consists of two steps: (1) create the sketch table $\mathcal{S}_{X\bowtie Y}$ by performing a join

between two sketches $\mathcal{S}_X$ and $\mathcal{S}_Y$ on their hashed keys $h(x)$ (as illustrated in Figure 4.2), and (2) apply *any* sample correlation estimator to the numerical data of $\mathcal{S}_{X \bowtie Y}$ to estimate the correlation between columns X and Y in $\mathcal{T}_{X \bowtie Y}$.

Proof of Theorem 1. Let $\mathcal{T}_{X \bowtie Y} = \langle K_{X \bowtie Y}, X_{X \bowtie Y}, Y_{X \bowtie Y} \rangle$ be the table resulting of the join between $\mathcal{T}_X = \langle K_X, X \rangle$ and $\mathcal{T}_Y = \langle K_Y, Y \rangle$. By definition, $K_{X \bowtie Y} = K_X \cap K_Y$. Let $g = h_u(h(k))$ be the composition of the hash functions h, h_u described above; g maps keys from the set $K_X \cap K_Y$ uniformly at random to $[0, 1]$. For this proof, let $g(K) = \{g(k) : k \in K\}$, $S_{X \bowtie Y}$ be the set of tuples $\{\langle k, x_k, y_k \rangle : k \in K_{X \bowtie Y}\}$ with the n smallest values $g(k) \in g(K_{X \bowtie Y})$, and $n < |K_{X \bowtie Y}|$. Notice that, because g assigns values uniformly and randomly, the set of tuples $\langle x_k, y_k \rangle \in S_{X \bowtie Y}$ is a uniform random sample of the set of tuples $\langle x_k, y_k \rangle \in \mathcal{T}_{X \bowtie Y}$.

Now consider the size-n synopses $L_{\langle K_X, X \rangle}$ and $L_{\langle K_Y, Y \rangle}$ of tables $\mathcal{T}_X$ and $\mathcal{T}_Y$ respectively. Let L_{K_X} and L_{K_Y} be the sets of keys from their respective synopses, i.e., $L_{K_X} = \{k_x : k_x \in L_{\langle K_X, X \rangle}\}$ and $L_{K_Y} = \{k_y : k_y \in L_{\langle K_Y, Y \rangle}\}$. Moreover, let $L_{X \bowtie Y} = \{\langle k, x_k, y_k \rangle : k \in L_{K_X} \cap L_{K_Y}\}$. Because a synopsis L always keeps the numerical values associated with their respective keys, to prove that $\langle x_k, y_k \rangle \in L_{X \bowtie Y}$ is a uniform random sample of the set of tuples $\langle x_k, y_k \rangle \in \mathcal{T}_{X \bowtie Y}$, it suffices to show that the set of keys $\{k : k \in L_{X \bowtie Y}\}$ is a uniform random sample of $K_{X \bowtie Y}$.

By definition, the set of keys $L_{K_X} \in L_{\langle K_X, X \rangle}$ (resp. $L_{K_Y} \in L_{\langle K_Y, Y \rangle}$) only contains the n keys $k \in K_X$ (resp. $k \in K_Y$) with the smallest values of $g(K_X)$ (resp. $g(K_Y)$), and the joined synopsis table $L_{X \bowtie Y}$ contains their intersection: $L_{K_X} \cap L_{K_Y}$. Thus, it is easy to see that $L_{K_X} \cap L_{K_Y} \subseteq \{k : k \in S_{X \bowtie Y}\}$ always holds. Without loss of generality, assume that $|L_{\langle K_X, X \rangle}| = |L_{\langle K_Y, Y \rangle}| = |S_{X \bowtie Y}| = n$. The best case happens when the sets of keys are equal, i.e., $L_{K_X} = L_{K_Y}$, in which case $|L_{K_X} \cap L_{K_Y}| = |S_{X \bowtie Y}|$ and $L_{K_X} \cap L_{K_Y} = \{k : k \in S_{X \bowtie Y}\}$. When $L_{K_X} \neq L_{K_Y}$, then $|L_{K_X} \cap L_{K_Y}| < |S_{X \bowtie Y}|$. Now,

assume that $|L_{K_X} \cap L_{K_Y}| = 1$. Then, the single key $k \in L_{K_X} \cap L_{K_Y}$ has the smallest value of $g(k)$. The sample $S_{X \bowtie Y}$ of size 1 also contains the same key $k \in L_{K_X} \cap L_{K_Y}$. More generally, if $|L_{K_X} \cap L_{K_Y}| = m$, then the set of keys of the sample $S_{X \bowtie Y}$ of size m is equal to the set $L_{K_X} \cap L_{K_Y}$. Therefore, the set of tuples $\{\langle x_k, y_k \rangle : k \in L_{K_X} \cap L_{K_Y}\}$ induced by $L_{X \bowtie Y}$ is also a uniform random sample of $\mathcal{T}_{X \bowtie Y}$. □

4.2.3 Discussion

From Theorem 1, we know that correlation sketches provide a valid estimate of the correlation between any two data columns *after a join* – i.e., between $X_{X \bowtie Y}$ and $Y_{X \bowtie Y}$. While this estimate is often accurate, it is based on a sub-sample of data and inaccurate estimates are inevitable (see e.g., Figure 4.3). Of course, this will be true for any randomized estimator, not just our algorithm.

The variance of CSK sketches' estimates depends on the correlation estimator used. In general, as we show in Section 4.4, correlation estimates converge to the true correlation when the sketch join sample size (i.e., the number of rows in $\mathcal{S}_{X \bowtie Y}$) increases. This sketch join size depends on multiple factors. First, there is a space-accuracy trade-off: as the number of minimum hash n increases, the probability of having larger join sizes also increases. The sketch join size also depends on the distribution of the join keys. Therefore, the hashing selection strategy used to include tuples in the sketch affects the sketch intersection size, and ultimately it also affects the variance of correlation estimation.

The fixed-size sample selection strategy adopted in CSK sketches is similar to the hashing strategy in [16], i.e., the sketch contains the n minimum values of $h_u(k)$. However, there is a wide range of possible hashing strategies with variable size [50, 166] that may have different accuracy-space trade-offs and could be used to build other

types of correlation sketches. Exploring the effect of different selection strategies on join-correlation estimation is outside the scope of this chapter. However, in Chapter 5 we propose a different hashing strategy that is more appropriate when the join key contains repeated values.

Another benefit of CSK sketches is that it retains all information contained in a KMV sketch [5, 16]. Therefore, it allows not only estimating correlations between the associated numerical columns but also enables estimating all statistics supported by the family of minimum-value sketches (e.g., cardinality, Jaccard containment, and similarity). These could be used, for example, to estimate the number of distinct elements in each individual column (K_X and K_Y), the containment of K_X in K_Y, and the size of the resulting join table $\mathcal{T}_{X \bowtie Y}$.

4.2.4 Implementation Details

We used the well-known 32-bits MurmurHash3 function to implement h, since it has been shown to perform similarly to truly random hashing functions [47]. For h_u, we used Fibonacci hashing [101], a simple multiplicative hashing function (also known as the *golden ratio multiplicative hashing*). To build the sketches, we implemented a tree-based algorithm similar to the one described in [16]. In summary, the algorithm performs one pass in the data while maintaining a tree that keeps the n tuples $\langle h(k), h_u(k), x_k \rangle$ with minimum $h_u(k)$ values. As we discuss in Section 4.4.3, we also implemented multiple correlation estimators.

4.3 Ranking Correlated Columns

To query large dataset collections, we focus on a variation of join-correlation queries that retrieve the top-k results, which we define more formally as follows.

Definition 3 (Top-k Join-Correlation Query)**.** *Given an integer $k > 0$ and query table $\mathcal{T}_Q$ (containing a column Q and a join column K_Q), find the top-k tables $\mathcal{T}_C$ in a dataset collection such that $\mathcal{T}_C$ is joinable with $\mathcal{T}_Q$ on K_Q and has the strongest (after-join) correlations between the columns $Q_{Q \bowtie C}$ and $C_{Q \bowtie C}$ from the joined table $\mathcal{T}_{Q \bowtie C}$.*

Exact top-k join-correlation queries can be answered by finding all tables $\mathcal{T}_C$ that are joinable on K_Q, performing a full join to construct $\mathcal{T}_{Q \bowtie C}$, and finally finding the tables that have the k greatest correlations. As we discussed, this is inefficient and does not scale when querying large dataset collections. Instead, we propose an efficient approach to compute approximate answers for these queries that use CSK sketches to *rank the results based on correlation estimates.*

Answering approximate join-correlation queries is challenging due to the possibility of errors. While searching for columns correlated with query column Q within a large collection of disconnected tables, there will typically be many more poorly correlated columns than highly correlated ones. In this "needle-in-a-haystack" setting, estimation errors will lead to many false positive results: inevitably some poorly correlated columns will look far more correlated with Q than they actually are, even more correlated than the columns we aim to find. Therefore, simply ranking results based on the correlation estimates may eventually produce poor rankings. In the remainder of this section, we propose a framework for scoring columns that addresses this problem.

Another important aspect of the implementation of our method is query evaluation. While CSK sketches efficiently estimate join-correlations, a naïve approach that

computes correlations for all possible column pairs can still be expensive for large collections. Computing correlations for all pairs, however, is not necessary: not all possible join keys have a high key overlap to yield useful joins. To efficiently answer top-k join-correlation queries we can leverage efficient data structures and query processing algorithms for set overlap search. Recall that a sketch includes a set of pairs $\langle h(k), x_k \rangle$. Since $h(k)$ is a discrete value, we can leverage existing data structures for efficient querying such as inverted indexes available in off-the-shelf systems (e.g., PostgreSQL, Apache Lucene) and efficient query processing algorithms for set similarity search such as JOSIE [177], ppjoin+ [165], CRSI [155], and Lazo [32]. In the remainder of this chapter, we focus on the simple indexing method described above. In Chapter 6, we analyze it in depth and develop an alternative efficient indexing method. For more details and theoretical analysis, we refer the reader to Section 6.2.

4.3.1 Ranking with Uncertain Estimates

To take into account the uncertainty associated with the estimations, we adopt a simple risk-averse scoring framework that selects k results and maximizes:

$$\max \sum_{i=1}^{k} \Big(|\hat{r}_{Q \bowtie C_i}| * \big(1 - risk(Q, C_i)\big) \Big) \tag{4.1}$$

where $|\hat{r}_{Q \bowtie C_i}|$ is the absolute value of the correlation estimate computed using a correlation estimator applied to $L_{Q \bowtie C_i}$, and $risk(Q, C_i)$ is a function that returns a number in the range $[0, 1]$ and measures the dispersion of the correlation estimates using $L_{Q \bowtie C_i}$, such as standard error or confidence interval length. Intuitively, whenever the risk associated with the estimation is non-zero, a penalty factor proportional to the risk is

applied to the estimate. Note that we consider the absolute of the correlation in Eq. 4.1, given that negative correlations can be as useful as positive correlations.

4.3.2 Measuring the Estimation Error Risk

A natural approach to measure the risk of estimation error is to use standard statistics such as the standard error of the estimator or the length of the confidence interval. In our scenario, however, we do not have access to these: since we do not explicitly join the columns (given that doing so is computationally prohibitive), we have little information about the distributions of column values. For instance, it is not possible to compute the exact variance (and similarly standard deviation or standard error) for the correlation because it depends on the underlying correlation of the complete data and fourth-order moments of the variables [170] (see Equation 2.4 for the variance under normal distribution assumptions or see [24] for the general case). Moreover, estimating high-order moments using small sample sizes may be unstable [20].

One statistic that we can compute is the standard error of the sampling distribution of the Fisher's Z correlation transformation [10], given by $SE_z = 1/\sqrt{n-3}$. While it assumes a bivariate normal distribution, its computation is simple and only depends on the known sample size n. Despite the normality assumption, this statistic is asymptotically equivalent to the error found in our theoretical analysis of $1/\sqrt{n}$ (Section 4.3.3). Thus, we expect it to work increasingly well as the sample size increases for any data distribution.

While we expect the Fisher's Z standard error to be accurate for large sample sizes, one drawback is that it assumes normality and does not take into account any information about the data distribution. Given that real-world data is seldom normally distributed [117] and the actual data distributions are usually unknown, we are inter-

ested in calculating distribution-independent confidence interval bounds. In this setting, non-parametric approaches such as bootstrapping are applicable. Bootstrapping only makes use of the samples generated by our sketches and does not assume any prior data distribution. While bootstrapping has been shown to have good performance for estimating confidence intervals for non-normal distributions [20], it has the disadvantage of having a very high computational cost – specially in settings like ours where it needs to be computed repeatedly over many columns.

To address the limitations of Fisher's Z (normality assumption) and bootstrapping-based (high computational cost) methods, in Section 4.3.3, we derive new confidence bounds using finite-sample concentration bounds for sums of independent random variables. These bounds only depend on the maximum and minimum values in $X_{X\bowtie Y}$ and $Y_{X\bowtie Y}$. Since $X_{X\bowtie Y} \subseteq X$ and $Y_{X\bowtie Y} \subseteq Y$, the values in these columns lie strictly within the range of values in X and Y. Thus, bounds on the range can be computed with a single pass over the columns (i.e., at the same time we construct the CSK sketch). Given that these bounds can be computed in constant time, there is essentially no additional computational overhead.

4.3.3 Confidence Interval Bounds

As discussed in Section 2.2, one challenge in deriving confidence intervals is that Pearson's correlation is known to be highly sensitive to individual samples. This prevents directly using e.g., a standard McDiarmid's inequality to bound the accuracy of an estimate for the correlation. Instead, we use individual Hoeffding inequalities to obtain confidence intervals for each individual component of the correlation estimator, and then apply a union bound to combine these results into an overall confidence

interval. This is similar to the approach used for constructing confidence intervals for the sample variance for non-Gaussian data in [6].

Analysis. CSK sketches estimate the Pearson's correlation between two columns $X_{X\bowtie Y}$ and $Y_{X\bowtie Y}$. Let $C_{low} = \min\{x \in X, y \in Y\}$ and $C_{high} = \max\{x \in X, y \in Y\}$ be upper and lower bounds in columns X and Y, and let $C = C_{high} - C_{low}$. Let $A = X_{X\bowtie Y} - C_{low}$ and $B = Y_{X\bowtie Y} - C_{low}$. Since a constant shift does not affect Pearson's correlation, we note that the correlation between $X_{X\bowtie Y}$ and $Y_{X\bowtie Y}$ is equal to:

$$\rho = \frac{\langle A - \mu_A\vec{1}, B - \mu_B\vec{1}\rangle}{\|A - \mu_A\vec{1}\|_2\|B - \mu_B\vec{1}\|_2},$$

where $\mu_A = \frac{1}{N}\sum_{i=1}^{N} A_i$ and $\mu_B = \frac{1}{N}\sum_{i=1}^{N} B_i$ are the means of A and B and $\vec{1}$ is the all ones vector. This expression is equal to

$$\rho = \frac{\nu_{A,B} - \mu_A\mu_B}{\sqrt{\nu_A - \mu_A^2}\sqrt{\nu_B - \mu_B^2}},$$

where $\nu_A = \frac{1}{N}\sum_{i=1}^{N} A_i^2$, $\nu_B = \frac{1}{N}\sum_{i=1}^{N} B_i^2$ and $\nu_{A,B} = \frac{1}{N}\langle A, B\rangle = \frac{1}{N}\sum_{i=1}^{N} A_iB_i$. Let $a, b \in \mathbb{R}^n$ be vectors containing n samples drawn uniformly without replacement from A, B. According to Theorem 1, the estimate to Pearson's correlation obtained via our sketches is equivalent to:

$$r = \frac{\nu_{a,b} - \mu_a\mu_b}{\sqrt{\nu_a - \mu_a^2}\sqrt{\nu_b - \mu_b^2}},$$

where:

$$\mu_a = \frac{1}{n}\sum_{i=1}^{n} a_i, \qquad \mu_b = \frac{1}{n}\sum_{i=1}^{n} b_i,$$

$$\nu_a = \frac{1}{n}\sum_{i=1}^{n} a_i^2, \qquad \nu_b = \frac{1}{n}\sum_{i=1}^{n} b_i^2,$$

$$\nu_{a,b} = \frac{1}{n}\sum_{i=1}^{n} a_i b_i.$$

Union Bound. We want to compute a confidence interval for ρ, meaning that, for some specified α (e.g., $\alpha = .05$), our goal is to compute upper and lower bounds ρ^{low}, ρ^{high} depending on our estimate r such that: $\Pr[\rho^{low} \leq \rho \leq \rho^{high}] \geq (1-\alpha)$. To do so, we first compute the upper and lower bounds for $\mu_A, \mu_B, \nu_A, \nu_B, \nu_{A,B}$, for all of the 5 parameters r depends on. For any parameter c, we want: $\Pr[c^{low} \leq c \leq c^{high}] \geq (1 - \alpha/5)$. For example, that $\Pr[\mu_A^{low} \leq \mu_A \leq \mu_A^{high}] \geq (1 - \alpha/5)$. We will discuss how to obtain these bounds shortly, but for now, we show how they can be used to compute a confidence interval. First let:

$$num_{low} = \nu_{A,B}^{low} - \mu_A^{high}\mu_B^{high}$$
$$num_{high} = \nu_{A,B}^{high} - \mu_A^{low}\mu_B^{low}$$

$$den_{low} = \sqrt{\max\left[0, \nu_A^{low} - (\mu_A^{high})^2\right] \cdot \max\left[0, \nu_B^{low} - (\mu_B^{high})^2\right]}$$
$$den_{high} = \sqrt{\max\left[0, \nu_A^{high} - (\mu_A^{low})^2\right] \cdot \max\left[0, \nu_B^{high} - (\mu_B^{low})^2\right]}$$

Then set:

$$\rho^{low} = \begin{cases} \frac{num_{low}}{den_{high}} & \text{if } num_{low} \geq 0 \\ \frac{num_{low}}{den_{low}} & \text{if } num_{low} < 0 \end{cases} \tag{4.2}$$

$$\rho^{high} = \begin{cases} \frac{num_{high}}{den_{low}} & \text{if } num_{high} \geq 0 \\ \frac{num_{high}}{den_{high}} & \text{if } num_{high} < 0. \end{cases} \tag{4.3}$$

By a union bound, we have $\Pr[\rho^{low} \leq \rho \leq \rho^{high}] \geq (1-\alpha)$.

Individual Parameter Bounds. The next step is to obtain the confidence intervals for each of the parameters $\{\mu_A, \mu_B, \nu_A, \nu_B, \nu_{A,B}\}$, which we do using Hoeffding's concentration inequality for bounded random variables. This bound is usually stated for sampling *with* replacement (i.e., independent random sampling), but Hoeffding proves in his original paper that it also holds for *without* replacement sampling, which only gives better concentration. Specifically Theorems 2 and 4 in [88] give:

Lemma 1 (Hoeffding's Inequality)**.** *Let $X_1, \ldots, X_N$ be a set numbers bounded $\in [0, C]$ with mean $\mu_X = \frac{1}{N}\sum_{i=1}^{N} X_i$. Let $Y_1, \ldots, Y_n$ be drawn independently without replacement from this set. Then:*

$$\Pr\left[\left|\mu_X - \frac{1}{n}\sum_{i=1}^{n} Y_i\right| \geq t\right] \leq 2e^{-2nt^2/C^2}.$$

Since A and B have values in $[0, C]$ by definition, each of the terms $\{\mu_A, \mu_B, \nu_A, \nu_B, \nu_{A,B}\}$ is the average of N numbers, bounded between $[0, C]$ for μ_A, μ_B and between $[0, C^2]$ for the others. Accordingly, we can apply Hoeffding's inequality to obtain a confidence interval with $\alpha/5$, as required by our analysis above. For example, consider μ_A. We have from Hoeffding's that:

$$\Pr[|\mu_A - \mu_a| \geq t] \leq 2e^{-2nt^2/C^2}.$$

For $2e^{-2nt^2/C^2} = \alpha/5$, we solve for $t = \sqrt{\ln(10/\alpha) \cdot C^2/2n}$ and can then set $\mu_A^{low} = \mu_a - t$ and $\mu_A^{high} = \mu_a + t$. As another example, consider ν_A. From Hoeffding's,

$$\Pr[|\nu_A - \nu_a| \geq t] \leq 2e^{-2nt^2/C^4}.$$

For $2e^{-2nt^2/C^4} = \alpha/5$, we solve for $t' = \sqrt{\ln(10/\alpha) \cdot C^4/2n}$ and can then set $\nu_A^{low} = \nu_a - t'$ and $\nu_A^{high} = \nu_a + t'$. The final bounds for all five parameters are as follows:

$$[\mu_A^{low}, \mu_A^{high}] = [\mu_a - t, \mu_a + t], \qquad [\mu_B^{low}, \mu_B^{high}] = [\mu_b - t, \mu_b + t]$$
$$[\nu_A^{low}, \nu_A^{high}] = [\nu_a - t', \nu_a + t'], \qquad [\nu_B^{low}, \nu_B^{high}] = [\nu_b - t', \nu_b + t']$$

$$[\nu_{A,B}^{low}, \nu_{A,B}^{high}] = [\nu_{a,b} - t', \nu_{a,b} + t']$$

where $t = \sqrt{\ln(10/\alpha) \cdot C^2/2n}$ and $t' = \sqrt{\ln(10/\alpha) \cdot C^4/2n}$.

Discussion. The above analysis gives a simple procedure to compute a valid confidence interval for correlation estimates: we first compute t, t' as defined above, which only requires the desired confidence level α, C which is pre-computed, and the sample size n (i.e., the number of rows in $\mathcal{S}_{X \bowtie Y}$. Then, we compute upper and lower bounds for each parameter, and plug into Equations (4.2) and (4.3) to obtain a final confidence interval.

Our analysis shows that for a fixed $1 - \alpha$ confidence level, the confidence interval bounds depend on the sample size and up to the fourth power of the range C of the variables A and B. This is in line with previous findings which point out that the deviation of the Pearson's sample estimator depends on the fourth moment of the variables [170, 171]. The accuracy of the bound also scales inversely with the square root of the sample size n, as expected.

Correlation is difficult to estimate whenever the population variance is low, because the estimator can become unstable (as these terms appear in the denominator of the equation for r). In particular, if either $\|A - \mu_A \vec{1}\|_2$ or $\|B - \mu_B \vec{1}\|_2$ is close to zero, then any small deviation of $\|a - \mu_a \vec{1}\|_2$ or $\|b - \mu_b \vec{1}\|_2$ from the true values that we are trying to estimate would lead to a large difference in ρ vs. r.

So, to get a better sense of the bounds, let us assume that, for both A and B, $var(A) = \nu_A - \mu_A^2$ and $var(B) = \nu_B - \mu_B^2$ are $\geq c$ for some constant c. From the analysis above, it is relatively easy to show that if we set $n = O\left(\frac{C^4 \ln(1/\alpha)}{\epsilon^2 c^2}\right)$, we will obtain a final confidence interval with width 2ϵ – i.e., we can estimate the Pearson's correlation to accuracy $\pm\epsilon$. For data bounded by C, it would be natural for the variance c to be on the order of C^2, in which case, the number of samples required is just $n = O\left(\frac{\ln(1/\alpha)}{\epsilon^2}\right)$. In other words, as the sample size n grows (i.e., as our sketches size increases) our estimate converges with error roughly $\frac{1}{\sqrt{n}}$.

Effect of Small Sample Sizes. Note that when sample sizes are small, the bounds for the standard deviation terms ($\nu - \mu^2$) may become negative, which makes den_{high} and den_{low} to become zero and thus yield invalid bounds. To address this problem, while computing ρ^{low} and ρ^{high}, we can replace the denominator of the Equations 4.2 and 4.3 by the product of the sample standard deviation of the variables computed using the samples induced by the sketch join, i.e., we can set $den_{high} = den_{low} = \sqrt{\nu_a - \mu_a^2}\sqrt{\nu_b - \mu_b^2}$. We refer to these modified upper bound and lower bound, respectively, as ρ_{HFD}^{high} and ρ_{HFD}^{low}. Although these are not true probabilistic bounds, their resulting confidence interval length ($\rho_{HFD}^{high} - \rho_{HFD}^{low}$) still provides meaningful information to measure the estimation error risk. This is because the denominator term serves as a normalization factor of the covariance term in the numerator (for which we still compute the true bounds).

4.3.4 Scoring Functions

Based on the framework (Section 4.3.1) and statistics (Sections 4.3.2 and 4.3.3) described above, we finally derive four different scoring functions that optimize ranking for correlated column discovery. Let $ci_{length} = \rho_{HFD}^{high} - \rho_{HFD}^{low}$ be the confidence interval

length of a correlation estimate. Then, set ci_{max}^{high} and ci_{min}^{high} to be the maximum and the minimum confidence interval length in a ranked list, respectively. We define the following risk penalization factors:

$$se_z = 1 - \frac{1}{\sqrt{\max(4, n) - 3}} \qquad ci_b = 1 - \frac{\rho_{PM1}^{high} - \rho_{PM1}^{low}}{2}$$

$$ci_h = 1 - \frac{ci_{length} - ci_{min}^{high}}{ci_{max}^{high} - ci_{min}^{high}}$$

Recall from our framework that each of these factors assume values in the range $[0, 1]$ and when they are equal to 1, the error risk is the minimum possible. The equations above are direct applications of the statistics introduced in Sections 4.3.2-4.3.3, i.e., Fisher's Z transformation standard error, Bootstrap CI, and our Hoeffding's CI. By plugging them into Equation 4.1, we have (in addition, we also consider a no-penalization factor in s_1):

$$s_1 = r_p, \qquad s_2 = r_p * se_z,$$

$$s_3 = r_b * ci_b, \qquad s_4 = r_p * ci_h,$$

where r_p and r_b are the absolute of the correlation estimated using, respectively, the Pearson's sample estimator and the PM1 bootstrap estimator [161]. We also use PM1 for the confidence intervals in ci_b.

4.4 Experimental Evaluation

We performed extensive experiments using both synthetic and real-world datasets to evaluate the effectiveness of CSK sketches and the proposed ranking strategies.

4.4.1 Datasets

We used three different data collections: *Synthetic Bivariate Normal*, which is synthetically generated and follows a pre-defined, well-known data distribution; the *World Bank's Finance* [164] and the *NYC Open Data* [121] collections, which contain real-world datasets, currently published in open data portals. We used snapshots of these repositories collected in September 2019 using Socrata's REST API [147]. All datasets were stored in plain CSV text files, and we used the *Tablesaw* library [150] to automatically parse and detect the basic data types for each column.

World Bank Finances (WBF). This collection contains 64 datasets (tables) related to the World Bank's Finances [164]. There is missing data in several columns and some columns contain large monetary values. From each table, we extracted all possible pairs of categorical and numerical data columns $\langle K_X, X \rangle$, out of which we generated all possible unique 2-combinations of columns pairs – 9,979,278 pairs of column pairs, i.e., $\langle K_X, X \rangle$, $\langle K_Y, Y \rangle$.

NYC Open Data (NYC). The tables from this dataset contain data published by New York City agencies and their partners [121]. Our snapshot includes 1,505 different datasets. Using the same process described above for the World Bank Finances collection, we generated $12,497,500$ pairs of column key-value pairs.

Synthetic Bivariate Normal (SBN). This dataset was generated by creating t tables consisting of n tuples $\langle k, x_k, y_k \rangle$, where $k \in K$ is a random unique string, and $x_k \in X$ and $y_k \in Y$ are real numbers drawn from a bivariate normal distribution with mean $\mu = 0$. The covariance of the column vectors X and Y was chosen in such a way that the Pearson's correlation coefficient between X and Y was approximately equal to a given parameter r_{XY}. We then created t pairs of tables $\mathcal{T}_X = \langle K_X, X \rangle$ and

$\mathcal{T}_Y = \langle K_Y, Y \rangle$. Finally, we reduced the size of the table $\mathcal{T}_Y$ from n to n' by selecting a uniform random sample of size $n' = n * c$, where c is a random real number in the range $(0, 1)$ indicating the join probability between X and Y. We set the number of table pairs $t = 3000$. For each table pair, we set n to be a random number drawn uniformly in the range $(0, 500000)$, and the correlation r_{XY} is drawn uniformly at random from $(-1, 1)$.

4.4.2 Correlation Estimation Accuracy

To study the effectiveness of our sketching method, we compare the estimated correlations against the actual correlations. For each pair of columns, $\langle K_X, X \rangle$ and $\langle K_Y, Y \rangle$, we first build their correlation sketches $\mathcal{S}_X$ and $\mathcal{S}_Y$, and then compute their correlation estimates $\hat{r}_{\langle X \bowtie Y \rangle}$. Then, we compare the estimates with the actual column correlations $r_{X \bowtie Y}$ computed using the (complete) join of columns $X_{X \bowtie Y}$ and $Y_{X \bowtie Y}$. Here, we only show the results for Pearson's correlation. We discuss the performance of other estimators in Section 4.4.3.

Figure 4.3 shows scatterplots for the estimates computed for all three datasets against the actual Pearson's correlation values, using sketches of size 256. We can clearly see that CSK sketches produce quite accurate results for the SBN dataset (Figure 4.3a), which contains only data drawn from a bivariate normal distribution: the estimates are concentrated close to their actual correlation values with only a few points deviating from the actual correlation values.

For the NYC (Figure 4.3b) and WBF (Figure 4.3c) datasets, which contain real-world data from unknown distributions, as expected, there are more incorrect predictions. We can observe that for many points with actual correlation equal to 0, the correlation is overestimated – see the vertical line around the value zero in the x-axis. This happens because some of the joins computed from the sketches can be too small, leading to this

inaccuracy. Note that Figures 4.3a, 4.3b, and 4.3c reflect estimates computed with joins (sketch intersection) that contain as few as 3 tuples ($n \geq 3$). Nonetheless, we can still observe a large concentration of points around the diagonal line, suggesting that CSK sketches are effective.

To better understand the behavior of the sketches with respect to sample size, we also plot in Figure 4.3d the results for the NYC data showing only join samples with size at least 20. The plot shows that indeed, for larger sample sizes, the behavior is more similar to that of the SBN dataset – the points are more concentrated closer to the diagonal, indicating the estimates are more accurate.

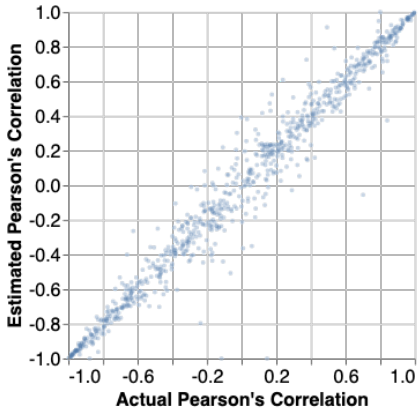
This provides evidence for the issue we discussed in Section 4.3: in large table collections, there are often many more poorly-correlated tables than well-correlated ones, and estimation errors may lead to a potentially large number of false positives. This underscores the importance of having effective ranking functions to help users focus their attention on tables that are more likely to be correlated. In this case, for example, a ranking function that prioritizes estimates computed from larger join samples is likely to be effective at pruning these false positive results.

4.4.3 Exploring Different Correlation Estimators

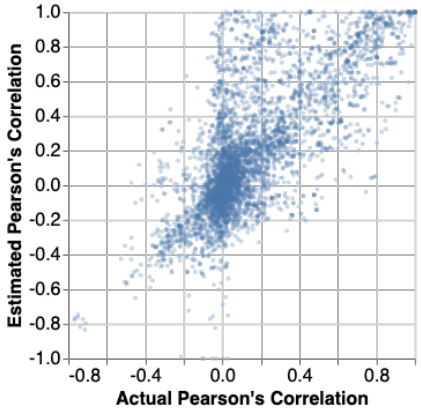
We studied the performance of different correlation estimators:

(1) Pearson's Sample Correlation: Computes the correlation using the formula defined in Equation 2.3.

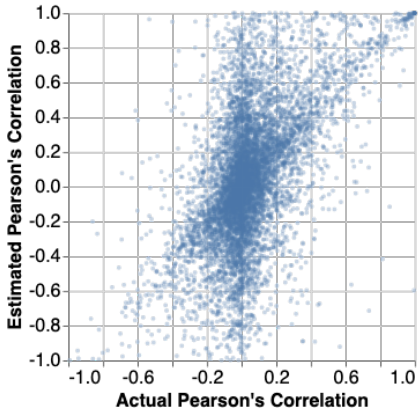
(2) Spearman's Rank Correlation: Let $r(x) = r_x$ where $r_x = 1$ for the smallest x, $r_x = 2$ for the second smallest x, and so on. The numeric column values are transformed using $r(x)$ and then the Pearson's correlation over the transformed values is computed.



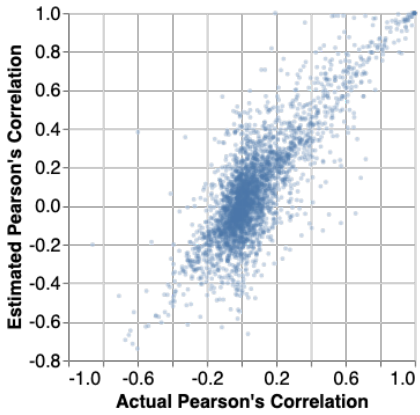
(a) SBN dataset, $n \geq 3$.



(b) WBF dataset, $n \geq 3$.



(c) NYC dataset, $n \geq 3$



(d) NYC dataset, $n \geq 20$

Figure 4.3: Estimation errors significantly vary for different sample sizes and different datasets with different data distributions. (a), (b), and (c) show the deviations of all column pairs for 3 different datasets. (d) shows the estimates from (c) after filtering out estimates that use fewer than 20 samples.

(3) Rank-based Inverse Normal (RIN): Similarly to Spearman's, a transformation function is applied before computing the Pearson's correlation. Following [18], we employ the *rankit* function [21] which is defined as: $h(x) = \Phi^{-1}\left(\frac{r(x)-1/2}{n}\right)$, where Φ is the inverse normal cumulative distribution function.

(4) Q_n correlation: Computes the correlation using as modified formula of Pearson's correlation and the robust Q_n scale estimator. For more details, see [144].

(5) PM1 Bootstrap: Performs repeated re-sampling with replacement of the data and recomputes the correlation using the Pearson's sample correlation estimator. The average of all re-samples is then used as an estimate. Instead of drawing a fixed large number of re-samples (say 10.000), we stop the re-sampling when the probability of changing the mean by more than 0.01 falls below 0.05%.

Most computed correlation estimates using sketches were compared to their corresponding population correlations (i.e., including the transformations of the population data when applicable). The only exception was the *PM1 Bootstrap* estimator, which is compared to the population's Pearson's correlation that it intends to estimate.

Correlation estimators differ in their sample size requirements and sensitivity to the data distribution [23]. To better understand how the correlation estimation accuracy changes when we vary the estimator and the amount of storage space used by the sketch (determined by the maximum sketch size), we plot the root mean squared error (RMSE) for different choices of correlation estimators and maximum sketch sizes in Figure 4.4. We can see a trend: for all estimators and maximum sketch sizes, the RMSE decreases as the intersection size between the sketches increases. The RMSE stabilizes roughly at 0.1. While the different estimators display similar trends, the plot also shows that some estimators are less robust (see e.g., the spikes in the line for Q_n).

4.4.4 Correlated Column Ranking

To better understand the effectiveness of the proposed scoring functions, we use the NYC data collection which contains the largest number of tables. For each pair of columns $\langle K_X, X\rangle$ in the collection, we retrieved all other joinable columns $\langle K_Y, Y\rangle$.

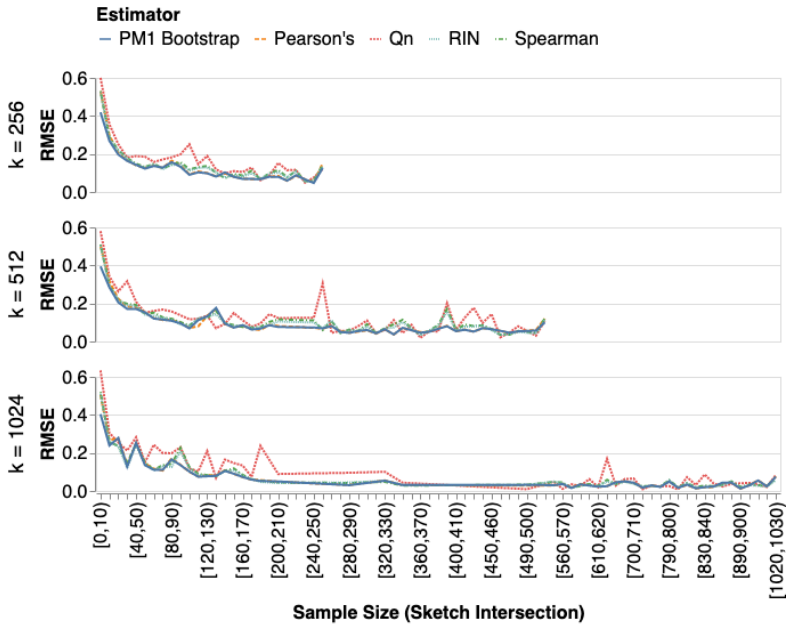


Figure 4.4: The sample size (sketch intersection) has an impact on RMSE. As the sketch intersection size increases, the RMSE decreases in the NYC dataset. Here, the k parameter (row) denotes the maximum sketch size (number of minimum values kept in the sketch).

Then, we ranked the list of retrieved columns using the scoring functions described in Section 4.3. As baselines we use: 1) a *random* scoring function, which assigns random scores in the range $[0, 1]$ drawn from a uniform distribution; 2) the *exact* Jaccard Containment (jc) similarity computed using the complete data after the join; and 3) the JC similarity estimated ($\hat{jc}$) using our correlation sketches. Even though these baselines do not take correlation into account, we use them as a point of comparison, since they represent existing methods for retrieving joinable tables.

We use two well-known measures to compare the scoring functions: mean average precision (MAP) and normalized discounted cumulative gain (nDCG) [97]. nDCG supports graded relevance judgments, i.e., each retrieved column can receive different scores depending on their relevance (the absolute value of the correlation, in our case)

and position on the list. In contrast, MAP supports binary judgment, and relies on thresholds to decide which columns are relevant (say, $r > .5$).

To evaluate different aspects of the ranking, we compute MAP for the whole ranked list using different thresholds for correlation relevance: $r > 0.50$ and $r > 0.75$. For nDCG, which has a tendency to assign higher scores, we compute the metrics only over the top-k search results, where $k = 5$ and $k = 10$. Table 4.1 summarizes the results. The first trend we observe is that the ranking produced using the JC similarity scores attains scores similar to random ordering. This confirms our hypothesis that JC is not well suited for ranking the results of join-correlation queries and also suggests that highly correlated columns can have different levels of JC similarity.

Note that all correlation-based ranking functions (Section 4.3.4) show significant improvements over the baselines. The ranking functions based on our Hoeffding bounds ($r_p * ci_h$) attain scores that are either better or very close to the bootstrap-based ranking function. This is especially interesting due to the fact that the Hoeffding-based CI can be computed in constant time, in contrast to the bootstrap method that requires many re-samples and the computation of correlations for these samples (typically, around between 1,000 and 10,000 iterations are used [20, 19]). In other words, we derive rankings that are comparable to those produced by the bootstrapping at a fraction of the cost. Moreover, the MAP scores suggest that the Hoeffding-based scoring is particularly effective at avoiding false positives with correlation above $r > 0.75$.

Note that Table 4.1 shows only average scores over all queries. To get a better sense of the improvements over all queries, we plot the distribution of evaluation scores in Figure 4.5. The histograms show the number of queries for different metric score values. Rows display the scoring functions: JC similarity (first row) and the Hoeffding-based scoring (second row). We can clearly see that, for all metrics, the score distributions

shift from left (bad scores) to right (good scores), indicating improvements in the whole metric range. It is also clear that, especially when considering the nDCG metric, most queries have very good scores (i.e., close to the optimal).

ranker	score	%
$r_p * ci_h$	0.529	193.2%
$r_b * ci_b$	0.516	185.9%
r_p	0.507	180.9%
$r_p * se_z$	0.420	133.1%
jc	0.180	0.0%
$\hat{jc}$	0.172	-4.8%
random	0.161	-10.8%

(a) MAP ($r > .75$)

ranker	score	%
$r_p * se_z$	0.472	102.1%
$r_p * ci_h$	0.467	99.8%
r_p	0.452	93.2%
$r_b * ci_b$	0.428	83.2%
$\hat{jc}$	0.239	2.3%
jc	0.234	0.0%
random	0.202	-13.7%

(b) MAP ($r > .50$)

ranker	score	%
$r_b * ci_b$	0.714	51.5%
$r_p * ci_h$	0.705	49.5%
r_p	0.699	48.4%
$r_p * se_z$	0.689	46.2%
random	0.481	2.1%
$\hat{jc}$	0.480	1.8%
jc	0.471	0.0%

(c) nDCG@5

ranker	score	%
$r_b * ci_b$	0.845	17.7%
r_p	0.843	17.5%
$r_p * ci_h$	0.841	17.3%
$r_p * se_z$	0.832	15.9%
$\hat{jc}$	0.726	1.2%
random	0.724	0.9%
jc	0.717	0.0%

(d) nDCG@10

Table 4.1: Ranking evaluation scores in terms of MAP and nDCG. The "%" column denotes relative improvement over jc.

4.4.5 Runtime Performance

Join-Correlation Estimation. To assess the efficiency of our approach and suitability for use in dataset search engines, we compare the running times of joins and correlation estimation computations using the sketches against the times required to execute the same computation over the full data. Since CSK sketches creates fixed-sized sketches for arbitrarily sized datasets, it is natural to expect a big performance impact due to

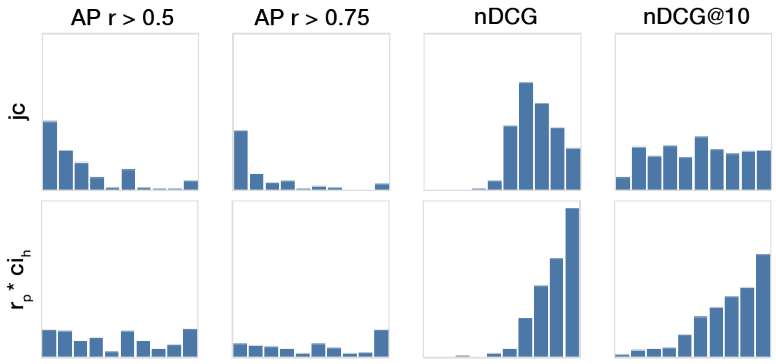


Figure 4.5: Distribution of the evaluation metric scores for different scoring functions. x-axis shows slices of the metric range $[0, 1]$. Each bar corresponds to a slice of width 0.1. The y-axis shows the number of queries that fall in each slice.

the reduction in the complexity of the problem – from potentially large table sizes to a small constant factor (sketch size). The results are summarized in Table 4.2, which include the times for joins and correlation computations using Pearson's and Spearman's coefficients. The results confirm that, with sketches, queries can be evaluated orders of magnitude faster than by using the full data, and given that the sketch size is fixed, their running time is more predictable. Note that, for this comparison, we assume the data is loaded in memory. However, for large datasets, the costs associated with using the full data would be even higher due to the cost of reading data from disk or transferring it over a network.

Query Evaluation. We also assessed the performance of evaluating join-correlation queries. For this experiment, we extracted all column pairs $\langle K_X, X \rangle$ from all 1,505 tables in the NYC dataset and randomly split them into two distinct sets, which we denote as *query set* and *corpus set*. Next, we set the maximum sketch size to 1024 and built an inverted index for all tables of the corpus set using a standard indexing library [2]. Finally, we used all pairs from the *query set* to issue queries against the index and measured the query execution time. We observed that the running times for

	Full data			Sketch		
percentiles	join	r_s	r_p	join	r_p	r_s
mean	42.219	8.494	0.240	0.026	0.000	0.004
std. dev.	367.696	134.357	9.314	5.618	0.042	0.279
75%	0.231	0.141	0.005	0.003	0.000	0.002
90%	7.038	0.154	0.011	0.006	0.001	0.004
99%	1360.605	29.583	0.385	0.012	0.003	0.013
99.9%	4021.838	2731.154	51.278	0.021	0.007	0.033

Table 4.2: Running times (in milliseconds) for computing joins and correlations using full data and sketches. Using the full data, queries can take orders of magnitude more time than when using the sketches. r_s denotes the Spearman's estimator and r_p denotes Pearson's estimator.

94% of queries are under 100 ms and approximately 98.5% of queries take under 200 ms. These times include retrieving the top 100 columns by key overlap, reading sketches from the index, and re-sorting all columns by estimated correlation. These preliminary results are promising and suggest that our approach can provide interactive response times for join-correlation queries over large data collections. In Chapter 6 we provide a more detailed performance evaluation of this and other methods.

Chapter 5

Tuple-based Sketches & Mutual Information Estimation

In Chapter 4, we introduced the idea of correlation sketches and a new method to build them, which we referred to as CSK. 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). Moreover, the implementation and experiments in Chapter 4 only considered traditional correlation measures, which may fail to identify non-monotonic relationships and are only applicable to numerical attributes.

Chapter Contributions. In this chapter, 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 join keys. We propose TUPSK, a new method for building correlation sketches 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 it guarantees a fixed-size sketch while keeping an unbiased uniform sample of the join. Due to the latter property, our sketches 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 MI with 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 identifying differences in the behavior of various combinations of sketches and MI estimators, and 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. Our experimental findings have uncovered useful observations that help guide the implementation and deployment of MI-based sketches for data augmentation.

5.1 MI Estimation over Joins with Repeated Keys

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.

5.1.1 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 4 (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 1.1. 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.

[1] While we limit joins to having high containment, our method does not prevent using existing strategies for handling `NULL`s in MI estimation [63, 93]. Evaluating these approaches is beyond the scope of this dissertation.

5.1.2 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** $\mathcal{T}_{aug}$

ON $\mathcal{T}_{train}[K_Y] = \mathcal{T}_{aug}[K_X]$;

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 5.2, 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 1.1 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 1.1(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 3. 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., $\{a \to [1], b \to [2, 2, 5], c \to [0, 3, 3]\}$. Next, the aggregate function `AVG` generates an intermediate aggregate table T_{aug} with the mappings $[a \to 1, b \to 3, c \to 2]$. 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 [102] and its derivatives could be used (see Section 2.3). 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.

5.2 Sketches for Joins on Repeated Keys

The task of estimating quantities over join results without materializing the join is a long-standing problem in the literature [1, 92]. 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 [1] which results in poor accuracy.

To address this issue, we use *coordinated sampling* [92, 156, 38, 13, 16, 136, 137, 42, 43]. 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 [92, 13]. 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 [48, 70], these estimators are tightly coupled with the measures being estimated and, thus, are not directly applicable to estimating MI (see Section 2.3). 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 [136, 13, 156]. As discussed in Section 5.1, 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 [156]. However, this is known to increase the variance of estimators and results in unbounded size [38]. Recent approaches introduce multiple sampling-rate parameters to select a fraction of the repeated entries [38, 92]. 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 [92, 38] that takes only a single parameter as input and provides a hard bound

on sketch size (Section 5.2.1). 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 5.3.2.3. To address this issue, we propose TUPSK (Section 5.2.2), 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 5.3.2.3, 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 [101].

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 2.3.

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.

5.2.1 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 $\texttt{AGG}$ 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 [158]. 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.

5.2.2 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 $N = 100$. Moreover, 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 5.3.2).

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 chapter have been proven to be consistent estimators [125, 79, 125] under some assumptions, such as i.i.d samples. While TUPSK does not guarantee sample independence (due to coordination), our experiments (Section 5.3.2.1) 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 [159] (and subsequently improved in [37]) 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.

5.3 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 5.3.2.2); (Q2) How does join-key distribution affect the MI estimation accuracy? (Sections 5.3.2.3 and 5.3.2.4); (Q3) How does accuracy vary depending on target and feature data types and, thus, the applied MI estimator? (Section 5.3.2); (Q4) How do sketches behave when estimating MI on real data collections? (Section 5.3.3).

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 [79]. 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 [133] that handles this case, referred to here as **DC-KSG**.

Sketching Methods. We evaluate LV2SK and TUPSK (Section 5.2). We also implemented another two-level approach that performs weighted sampling based on key frequencies (using priority sampling [66]) 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 CSK sketches (Chapter 4) 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).

5.3.1 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 [80, 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 $-\frac{1}{2}\ln(1 - r^2)$ 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 $r = \sqrt{1 - \exp(-2 \cdot I_{true})}$; 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 [160].

As done in [79], 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 [102]. 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 [133]. Additionally, MixedKSG [79] 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 [79] for mixture-mixture, and DC-KSG [133] 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 [79] and DC-KSG [133], 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, 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).

5.3.2 Experiments Using Synthetic Data

5.3.2.1 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 5.3.3, 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

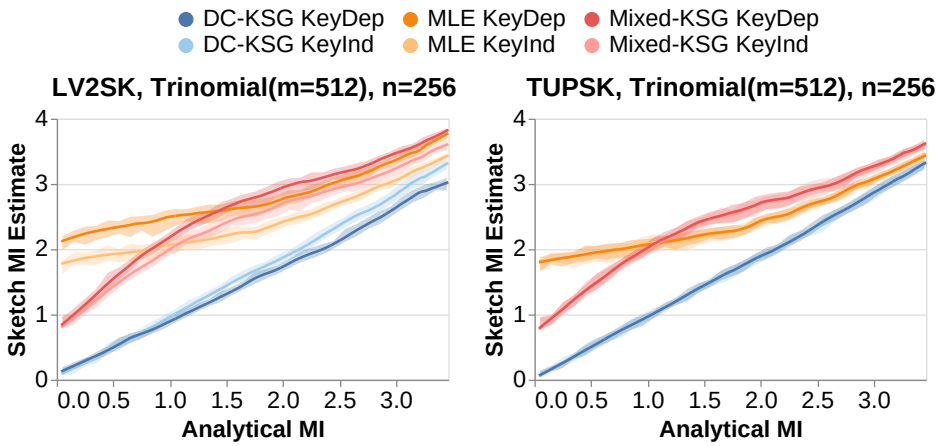


Figure 5.1: True MI vs MI estimates computed using sketches of size $n = 256$. Each plot shows a different sketching method (LV2SK on the left and TUPSK on the right) and each line shows results for different data types/estimators and join key generation processes. TUPSK is more robust to the join key distribution.

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.

5.3.2.2 Assessing Sketch Estimation Accuracy

Figure 5.1 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 5.3.2.1, 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 5.3.3, we further discuss this issue based on our results on data sourced from real-world open data repositories.

5.3.2.3 Effect of the Join Key Distribution

In Section 5.2, we described two different methods to select samples to include in the sketch. In Figure 5.1, 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 5.1(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 5.1(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.

5.3.2.4 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 to $I(X, Y) \approx 4.85$. As

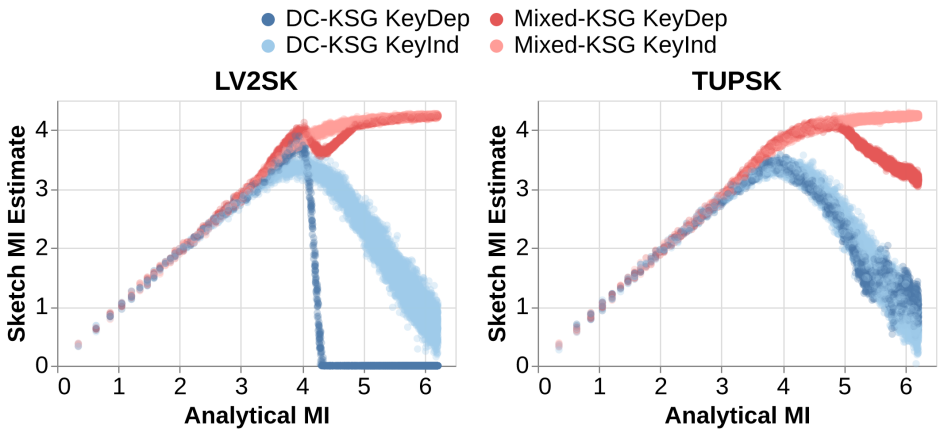


Figure 5.2: True MI vs MI estimates computed using sketches of size $n = 256$ for `CDUnif`. Each plot shows a different sketching method while each line shows results for different data types/estimators and join key generation processes.

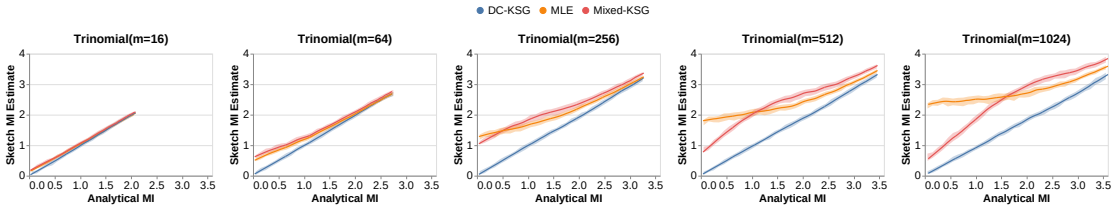


Figure 5.3: Sketch MI estimate versus the true MI computed using distribution parameters. Sketch size is $n = 256$ for all plots.

Figure 5.2 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 5.3 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 MixedKSG (●). 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 5.3.1), 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. 5.3.

5.3.2.5 Comparison to Other Baselines

In Table 5.1, 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 5.1: 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.

5.3.3 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) [164] and the NYC Open Data (NYC) [121]. Our experimental data consists of snapshots of these repositories collected in September 2019 using Socrata's REST API [147].

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

[2]We used the Tablesaw library [150] to perform type inference.

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 5.3.2.1, 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 [159].

5.3.3.1 Approximation Accuracy

Table 5.2 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 5.2.1), tends to generate a larger average join size. However, despite 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.

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 5.2: 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.

5.3.3.2 Effect of Sketch Join Size

In Figure 5.4, 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.

5.3.3.3 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

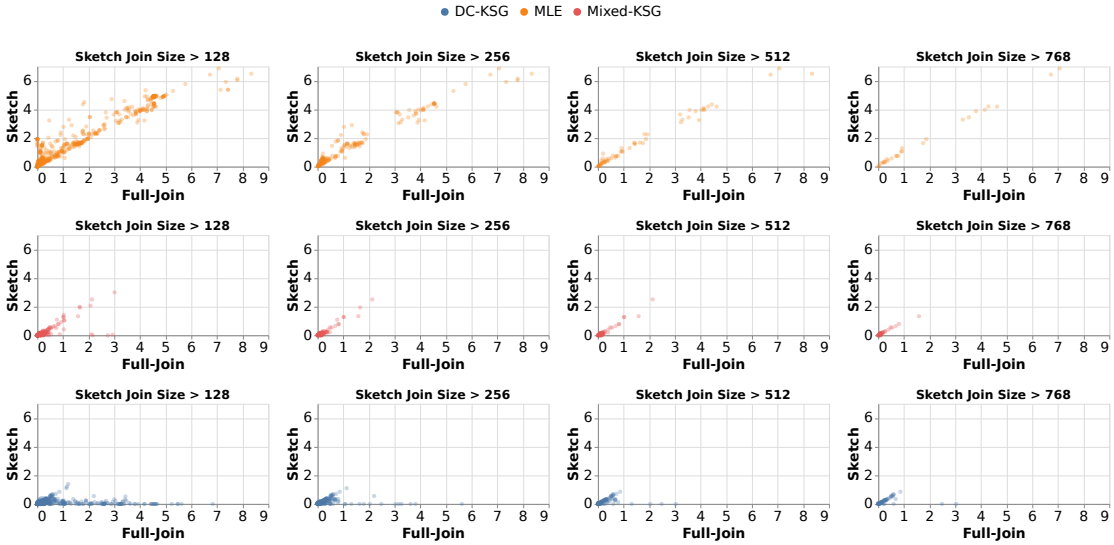


Figure 5.4: Sketch MI estimate versus the MI estimate computed using the full join output for tables from the WBF collection. Sketches are created using TUPSK with size $n = 1024$ for all plots.

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).

5.3.4 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 [136, 137]. 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.

Chapter 6

Weighted Join-Correlation Queries and QCR Indexes

In Chapter 4, we introduced Join-Correlation Queries: given an input query table $\mathcal{T}_Q$ and a tabular dataset collection $\mathcal{D}$, they retrieve tables in $\mathcal{D}$ that are both joinable with $\mathcal{T}_Q$ and contain columns correlated with one or more columns in $\mathcal{T}_Q$. Answering these queries poses significant scalability challenges, and our approach achieves scalability by providing approximate answers at the cost of precision and recall. In particular, we used a two-stage approach that relies on inverted indexes and randomized sketching algorithms: it first retrieves the top k most joinable (candidate) tables using indexes and then re-ranks them using approximate correlation estimates computed efficiently with data sketches.

While this sketching-based approach is effective at estimating correlations, typically there are many more joinable tables than tables that are both highly joinable *and* contain a correlated column. Therefore, by retrieving only the top k most joinable tables in the first stage, this approach may miss highly correlated tables that are not as highly

joinable as other uncorrelated tables. Additionally, the user has no fine-grained control over the level of joinability and correlation desired. For instance, given a table $\mathcal{T}_A$ with joinability (j) and correlation (r) scores $r_A = 0.95$ and $j_A = 0.92$, and a table $\mathcal{T}_B$ with scores $r_B = .92$ and $j_B = 0.95$, it is not possible to specify which one should be preferred using a standard join-correlation query.

Chapter Contributions. In this chapter, we propose a new approach to correlated dataset search. First, we define a more general version of join-correlation queries that allows users to specify the balance between joinability and correlation that is required for specific applications. Moreover, we propose a new hashing scheme that enables the retrieval of columns that are both joinable and correlated in a *single step*. This leads both to an increase in recall and overall ranking quality at a smaller storage cost compared to existing approaches.

Our method is based on a partition of the numerical plane into quadrants, inspired by a simple correlation estimator known as Quadrant Count Ratio [89]. This partitioning scheme allows us to apply an additional hashing step that takes into account both the join and the numerical attributes of interest. By doing so, we reduce the problem of finding join-correlated tables to the simpler problem of set overlap search between hashes, which our method generates for the query and for the tables in the data collection.

The remainder of this chapter is organized as follows:

- In Section 6.1, we define the new class of weighted join-correlation queries, of which join-correlation queries are a special case;
- In Section 6.2, we describe a novel hashing scheme and new sketch-based index, the *QCR index*, that supports efficient correlated table search over large collections;

- In Section 6.3, we describe an experimental evaluation using synthetic and real-world dataset collections which shows that: 1) Our QCR-based retrieval approach attains higher precision and recall and a better balance between ranking accuracy and joinability, when compared to existing approaches; 2) The QCR index achieves a better space-accuracy trade-off: for the same level of recall and ranking accuracy, the QCR index needs sketch sizes that require only about $1/4$ of the storage needed by our CSK-based approach (from Chapter 4). Consequently, the QCR index reduces the number of terms required per table, which leads to better query processing times when compared to retrieval strategies that attain a similar retrieval quality.

6.1 Weighted Join-Correlation Queries

Let $\mathcal{T}_Q$ be a query table comprised of a categorical column K_Q and a numerical column Q, and $\mathcal{D}$ be a dataset collection containing multiple tables $\mathcal{T}_C$, such that each table has a categorical column K_C and a numerical column C. Columns K_Q and K_C are the join attributes of tables $\mathcal{T}_Q$ and $\mathcal{T}_C$ respectively, and they may have overlapping sets of values that can be used to join $\mathcal{T}_Q$ and $\mathcal{T}_C$, resulting in a new table $\mathcal{T}_{Q\bowtie C}$. Using the relational algebra notation, we say that $\mathcal{T}_{Q\bowtie C} = \pi_{k,q_k,c_k}(\mathcal{T}_Q \bowtie_{K_Q=K_C} \mathcal{T}_C) = \{\langle k, q_k, c_k\rangle : k \in K_Q \cap K_C\}$. Finally, the values of numerical columns Q and C associated with each key $k \in K_Q \cap K_C$ are denoted as q_k and c_k respectively. Note that the above definition assumes that k uniquely identifies one row in the table. However, real-world data often contain repeated categorical values (as illustrated in column K_C from table $\mathcal{T}_C$ of Figure 6.1). In this example, the repeated key "a" is associated with the set of values $\{5.5, 4.5\}$. In such cases, we are interested in the table generated after applying an aggregate function (e.g., AVG, SUM, MAX, etc.) over the values associated

$\mathcal{T}_Q$

K_Q	Q
a	6.0
b	4.0
c	2.0
d	3.0
e	0.5
f	4.0
g	2.0

$\mathcal{T}_C$

K_C	C
a	5.5
a	4.5
b	4.0
b	2.0
c	2.5
d	4.0

$\mathcal{T}_{Q \bowtie C}$

$K_{Q \bowtie C}$	$Q_{Q \bowtie C}$	$C_{Q \bowtie C}$
a	6.0	5.0
b	4.0	3.0
c	2.0	2.5
d	3.0	4.0

Figure 6.1: An example of a query table $\mathcal{T}_Q$, a candidate table $\mathcal{T}_C$, and the joined table $\mathcal{T}_{Q \bowtie C}$ created after joining $\mathcal{T}_Q$ with $\mathcal{T}_C$, and aggregating repeated keys using the AVG aggregate function. We are interested in finding candidate tables $\mathcal{T}_C$ in a collection $\mathcal{D}$ such that the after-join correlation between attributes $Q_{Q \bowtie C}$ and $C_{Q \bowtie C}$ is high.

with the repeated values of k, e.g., $c_{\text{a}} = \texttt{AVG}(\{5.5, 4.5\}) = 5.0$. This is the desired behavior for applications such as data augmentation for data analysis and machine learning models, where the goal is to add new columns (features) to an existing training dataset while maintaining the same number of rows. Figure 6.1 shows a complete example of query and candidate tables, along with their corresponding join table after value aggregations.

Our goal is to query a dataset collection $\mathcal{D}$ with a query table $\mathcal{T}_Q$ to find other tables $\mathcal{T}_C$ that are not only joinable with $\mathcal{T}_Q$, but that also contain the top-k numerical columns correlated with Q after a join. We recall the query definition from Section 4.3:

Definition 3 (Top-k Join-Correlation Query)**.** *Given an integer $k > 0$ and query table $\mathcal{T}_Q$ (containing a column Q and a join column K_Q), find the top-k tables $\mathcal{T}_C$ in a dataset collection such that $\mathcal{T}_C$ is joinable with $\mathcal{T}_Q$ on K_Q and has the strongest (after-join) correlations between the columns $Q_{Q \bowtie C}$ and $C_{Q \bowtie C}$ from the joined table $\mathcal{T}_{Q \bowtie C}$.*

Note that while this definition only focuses on high correlation, many applications might also require a high degree of joinability. Consider, for instance, the problem of relational data augmentation [39]: the goal is to find new, relevant candidate columns

to be included as features in a machine learning model, augmenting the model's initial feature table with such columns. When searching large dataset collections for joinable tables, it is likely that we will encounter large tables that have only a few coincidental overlapping values (e.g., $|K_Q|$ and $|K_C|$ could have both over a thousand rows, but their overlap could be only a few values, say $|K_Q \cap K_C| = 3$). Yet, a few samples can yield very high correlations even tough they may not be significant and have very low p-values. In this case, a new column that is highly correlated with the model's target may not improve the quality of the initial model if the overlap between its initial feature table's keys and the new column's keys is too low. Moreover, this would lead to many missing data entries in the resulting joined table, and deciding how to handle them (e.g., through a missing data imputation strategy, or by simply removing them) is not trivial. Therefore, given two tables with similar correlation levels, the table with the highest join key overlap is preferred for retrieval.

To take into account the user preferences regarding joinability and correlation, we define a more general class of join-correlation queries:

Definition 5 (Weighted Top-k Join-Correlation Query). *Given $\mathcal{T}_Q$, an integer $k > 0$, and a user preference weighting function $W(j, r)$ that combines a joinability score j and correlation coefficient r, find the top-k joinable tables $\mathcal{T}_C \in \mathcal{D}$ with the highest (after-join) scores assigned by W based on the correlation r between numerical columns $Q_{Q \bowtie C}$ and $C_{Q \bowtie C}$ and joinability score j between $\mathcal{T}_Q$ and $\mathcal{T}_C$.*

We can define W to express a user's preferences regarding the trade-off between joinability and correlation, as follows:

$$W(j,r) = \begin{cases} w & \text{if } j > 0 \text{ (i.e., the tables are joinable)} \\ 0 & \text{otherwise} \end{cases} \tag{6.1}$$

where r is the absolute value of a correlation coefficient such as Pearson's correlation,[1] j is a joinability score such as the Jaccard Containment (JC), and w is a combination of j and r.

A natural choice for w is a weighted mean of correlation or joinability. For example, the weighted geometric mean of a set of real numbers $X = \{x_0, ..., x_n\}$ is defined as $(\prod_{i=1}^{n} x_i^{\alpha_i})^{1/\sum_{i=1}^{n} \alpha_i}$, where α_i is the weight associated with each number x_i. Applying it to our setting, we get that:

$$w = (j^{\alpha_j} r^{\alpha_r})^{1/(\alpha_j + \alpha_r)},$$

where α_j is the weight for joinability j and α_r is the weight for correlation r.

Note that we can express the join-correlation query from Definition 3 using Definition 5 and Equation 6.1 by defining $\alpha_r = 1$ and $\alpha_j = 0$, in which case $w = r$. Similarly, we can express the "pure" joinable table search objective by defining $\alpha_r = 0$ and $\alpha_j = 1$, in which case $w = j$. In the equal-weights special case where $\alpha_j = 1$ and $\alpha_r = 1$, w becomes $w = \sqrt{j * r}$.

Defining the weights and combination functions between correlation and joinability is application-specific and beyond the scope of this dissertation. Here, we focus on the

[1] We use the absolute value due to the assumption that both positive and negative correlations are of interest, but $W(j,r)$ and w can be adjusted accordingly if this is not the case. As we show next, this simplifies the problem.

version of correlation queries proposed in Chapter 4 (Definition 3), where $w = r$, and on an equal-weights case where high correlation and joinability are equally desirable. Specifically, we propose a new hashing and indexing scheme that allows us to retrieve tables that maximize an equal-weight weighting function (details in Section 6.2).

As we show in Section 6.3, this retrieval method enables not only the discovery of tables with both high correlation and joinability but *also* significantly improves the precision and recall when the objective is high correlation only ($w = r$), the objective from Definition 3 discussed in Chapter 4.

6.2 Single-Stage Correlated Table Retrieval

Our approach to efficiently answer weighted top-k join-correlation queries is based on a combination of sketching and query processing algorithms for fast top-k document retrieval [153]. We propose a new hashing scheme that derives terms to be indexed in an inverted index, which can then be used to retrieve the correlated tables. Using this hashing scheme, the task of retrieving correlated tables is reduced to finding the top-k candidate tables that have the most hashed terms in common with the query table. This allows us to apply existing query processing algorithms for document retrieval [26, 60, 153] to retrieve correlated tables.

6.2.1 The QCR Hashing Scheme

A challenge in applying document retrieval algorithms to correlated table search is that these algorithms are based on term matching (discrete values), whereas correlations are derived from real numbers. To workaround this problem, our previous approaches from Chapter 4 proposed to retrieve joinable tables by matching tables using only the

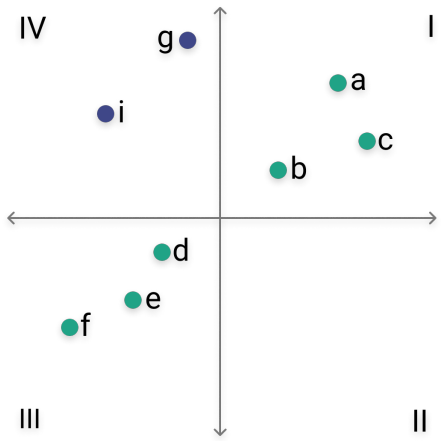


Figure 6.2: An example of the four quadrants, where green points (●) contribute to positive correlation and blue points (●) contribute to negative correlation.

values from the join columns. This approach forces the introduction of an additional re-ranking step that detects tables that are joinable but do not contain correlated columns.

To overcome this problem, we propose a new hashing scheme that allows considering both the join keys and the numeric values associated with each key. Our hashing scheme is based on the correlation estimator known as Quadrant Count Ratio (QCR) [89]. The QCR estimator is defined as:

$$r_{QCR} = \frac{n(I) + n(III) - n(II) - n(IV)}{N},$$

where $n(i)$ is the number of samples located in the i^{th}quadrant, and N is the total number of samples. This is illustrated in Figure 6.2.

The QCR estimator is simple and has multiple properties in common with popular correlation coefficients (e.g., Pearson's, Spearman's and Kendall's Tau) [89]. For instance, it yields numbers in the range $[-1, 1]$; uncorrelated data has correlation close to 0; and data with perfect positive or negative correlations according to these measures also have perfect QCR correlations. Therefore, the QCR not only provides a good basis to

develop a retrieval strategy for these correlation coefficients but also provides a simple intuition for how to partition the continuous space of a set of numeric values $\{x\}$ into a binary space (e.g., $x > 0$ and $x < 0$).

For our problem, finding tables that are both correlated and joinable, it is not necessary to accurately estimate the r_{QCR} coefficient for all joinable tables. Instead, to find large positive correlations, it suffices to find tables that maximize the number of points in quadrants I and III, while to find large negative correlations, points need to be in quadrants II and IV.

The absolute r_{QCR} correlation for the joined table $\mathcal{T}_{Q \bowtie C}$ is equal to:

$$|r_{QCR}| = \frac{|N^+ - N^-|}{N^\cap}, \tag{6.2}$$

where $N^\cap$ is the number of points after the join (i.e., the number of rows in the joined table), $N^+ = n(I) + n(III)$ is the number of points in the positive quadrants, $N^- = n(II) + n(IV)$ is the number of points in the negative quadrants.

Note that Equation 6.2 violates the non-negative monotonicity property required by the top-k query processing algorithms [153, 71]: encountering a point that lies in the quadrants II and IV could lead to a decrease of the current top-k table scores. However, we can show that to maximize $|r_{QCR}|$, it suffices to individually maximize $\frac{N^+}{N^\cap}$ or $\frac{N^-}{N^\cap}$.

To see this, first observe that, by definition, $N^\cap = N^+ + N^-$. Thus, $N^- = N - (N^+)$ and so by maximizing N^+ we are also minimizing N^-. Substituting $N^- = N^\cap - N^+$ (or $N^+ = N^\cap - N^-$) in the equation for r_{QCR}, we get that:

$$r_{QCR} = 2\frac{N^+}{N^\cap} - 1 \qquad \text{and} \qquad r_{QCR} = -\left(2\frac{N^-}{N^\cap} - 1\right).$$

In these equations, it is clear that $r_{QRC} > 0$ when $\frac{N^+}{N^\cap} > \frac{1}{2}$ and that the absolute correlation $|r_{QCR}|$ assumes maximum value when $\frac{N^+}{N^\cap}$ is maximized or minimized. Moreover, maximizing $|r_{QCR}|$ is equivalent to maximizing $max(\frac{N^+}{N^\cap}, \frac{N^-}{N^\cap})$.

In other words, we can split the problem of finding tables with the highest correlations into two sub-problems of finding the top-k tables that have the maximum between $\frac{N^+}{N^\cap}$ and $\frac{N^-}{N^\cap}$, which are monotone functions. In what follows, we propose a hashing scheme that allows us to achieve this goal.

The QCR hashing scheme builds on the sketching strategy proposed for join-correlation estimation in Chapter 4. For each table $\mathcal{T}_C = \langle K_C, C \rangle$, we first build a correlation sketch $\mathcal{S}_C$. The sketch is then used to derive a set of terms T_C to represent the table in the inverted index. To make this chapter self-contained, we will first briefly describe how to build these sketches (Section 6.2.2), and then we will describe how to compute the QCR index terms from sketches (Section 6.2.3). Here, we will use the CSK sketches described in Chapter 4, but one could also use TUPSK (described in Chapter 5) when the application does not allow aggregating values associated with repeated join keys (e.g., when performing left joins for data augmentation with repeated join keys on the left table). We refer the reader to Chapter 5 for details on TUPSK sketches and the effect of repeated values in join keys.

6.2.2 Building the Correlation Sketches

A key idea behind correlation sketches is to use hashing techniques to consistently choose the keys from each table that are included in the sketch. We use two different hashing functions, h and h_u, to create a sketch $\mathcal{S}_C$. The first of them, h, is a collision-free hash function that maps the key values $k \in K_C$ onto distinct integers, uniformly at random. Given that the hashed keys $h(k)$ are unique, they are used as tuple identifiers

in the sketch. The second function, h_u, maps the hashed keys $h(k)$ uniformly and randomly onto the unit range $[0, 1]$. This allows for the selection of a small sample of n tuples $\langle h(k), c_k \rangle$ from a table $\mathcal{T}_C = \langle K_C, C \rangle$. In other words, $\mathcal{S}_C$ includes the n tuples $\langle h(k), c_k \rangle$ with the minimum values of $h_u(k)$, i.e., $\mathcal{S}_C = \{\langle h(k), c_k \rangle : k \in min(k, h_u(k))\}$, where min is a function that returns a set containing the keys k with the n smallest values $h_u(k)$. In the case where the set $\{k\}$ contains repeated keys, the values c_k can simply be aggregated using aggregate functions as described in Section 6.1.

Building sketches effectively reduces large tables to a sample that contains only n tuples, while guaranteeing with high probability that different sketches will have similar sets of hashed keys $h(k)$ if their original tables are joinable. Moreover, a pair of sketches $\mathcal{S}_Q$ and $\mathcal{S}_C$ can be used to compute correlations by creating joined sketch $\mathcal{S}_{Q \bowtie C}$ and applying any correlation estimator (see Chapter 4).

6.2.3 Building the QCR Index Terms

Given the sketch for table $\mathcal{T}_C$, we derive a set of terms $T_C = \{t_k\}$ to represent $\mathcal{T}_C$ in the inverted index. Since we aim to estimate the ratio of points that fall into each quadrant after the join between tables $\mathcal{T}_Q$ and $\mathcal{T}_C$, our term hashing scheme must satisfy two constraints. Given two sets of hashed terms T_Q and T_C, derived from tables $\mathcal{T}_Q$ and $\mathcal{T}_C$ respectively, any pair of hashed terms $t_i \in T_Q$ and $t_j \in T_C$ must be equal only if (1) their original join key values are the same, *and* (2) their numerical values belong to the same quadrant. To satisfy these constraints, we derive index terms t_k as a function of the set of numerical values $\{c_k\} \in \mathcal{S}_C$ and of the hashed key $h(k)$. Specifically, we

compute t_k as:

$$t_k = \begin{cases} h(h(k) \oplus +1) & \text{if } c_k - \mu_c > 0 \\ h(h(k) \oplus -1) & \text{if } c_k - \mu_c < 0 \end{cases} \tag{6.3}$$

where μ_c is the average of $\{c_k\}$, and $\oplus$ denotes concatenation of a hash $h(k)$ and the quadrant ID. Points which $c_k - \mu_c = 0$ do not contribute to correlation and are ignored.

Note that μ_c is a reference point used to partition the quadrants. As illustrated in Figure 6.3, this operation can be seen as a translation of the coordinate system to be centered at zero (mean centering), and most correlation measures are not affected by this transformation [136].

In the example of Figure 6.3, our hashing scheme assigns the same hash value to $\langle b, c_b \rangle \in C$ and $\langle b, q_b \rangle \in Q$ because their join key is `b`, and both c_b and q_b are greater than μ_c and μ_q, respectively. However, it would not assign the same hash value to c_b and q_b if, for example, c_b were less than μ_q. Moreover, note that the terms generated for `j` and `h` would not match the terms for any other point because their key values are different, and we assume that h is collision-free.

It is worth noting that we estimate μ_c using the data from the sketches (i.e., before the join), which assumes that the mean does not change significantly after the join. While this assumption may not always hold, it is a required assumption because we do not have access to the necessary data to compute the after-join mean at indexing time (as it depends on the query table). A shift in the mean after the join may cause some terms that belong in the same quadrant to not match. However, this affects tables with high and low correlations equally. Moreover, by using correlation estimates computed using the sketches, we can correct possible errors in a reranking phase. In practice, our

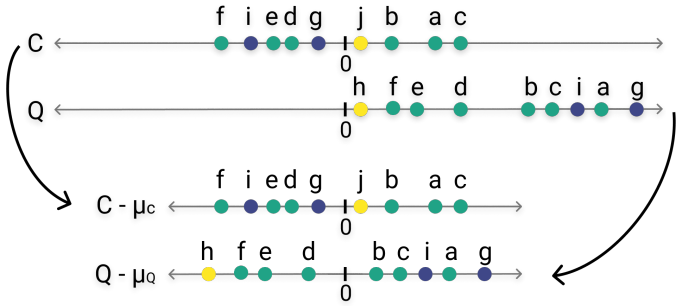


Figure 6.3: An example of mean centering for two tables $\mathcal{T}_C$ and $\mathcal{T}_Q$. Letters represent join keys k and the positions along the lines represent values c_k. Yellow circles (●) mean that the keys do not join. Green circles (●) denote that the keys join and are located in "positive quadrants" (I and III), and Dark blue circles (●) mean that the rows join and are in "negative quadrants" (II and IV). A projection of the table generated after the join onto the plane is shown in Figure 6.2.

experimental evaluation (Section 6.3) shows that this approach works well and attains high recall values.

In summary, to index a table $\mathcal{T}_C$ we first compute its sketch $\mathcal{S}_C$ and then use Equation 6.3 to compute terms T_C, which are ultimately used to represent the table $\mathcal{T}_C$ in the QCR inverted index.

6.2.4 Querying the QCR Index

When a table $\mathcal{T}_Q$ is provided as a query, we apply a process similar to the one described in Section 6.2.3 to query the inverted index. We start by constructing a sketch $\mathcal{S}_Q$ and then generate terms to query the index. Note that querying the index using terms T_Q returns only positive correlations ($r > 0$). To see this, consider the mean-centered example of C and Q at the bottom of Figure 6.3. If a user queries the index using the terms T_C generated for $\mathcal{T}_C$, a comparison with terms T_Q would only match the hashes for the (green) join keys `a`, `b`, `c`, `d`, `e`, and `f`, which have the same join keys and are on the same side (positive/negative) of the mean-centered line. The

points of a negatively correlated column ($r < 0$) projected onto the plane would lie mostly on quadrants II and IV, the opposite of our query terms T_Q.

If one is also interested in retrieving negative correlations, an additional step is therefore necessary. In this case, we generate two sets of terms: T_Q^+ is used to retrieve positive correlations, and T_Q^-, retrieves negative correlations. We set T_Q^+ to be equal to T_Q, and we compute T_Q^- using the additive inverse of numerical values $\{q_k\} \in \mathcal{S}_Q$ — i.e., we apply a transformation to $\{q_k\}$ that multiplies all of its elements by -1. We show in our theoretical analysis (Section 6.2.6) that the size of the overlaps $s^+ = |T_Q^+ \cap T_C|$ and $s^- = |T_Q^- \cap T_C|$ are roughly proportional to $\frac{N^+}{N^\cap}$ and $\frac{N^-}{N^\cap}$, respectively, within some error bounds. Therefore, to retrieve positively and negatively correlated tables, we can issue a disjunction of two queries — one for each set of terms — and keep the top-k tables with highest scores of either s^+ or s^-. In other words, we find the top-k tables that maximize the score $s = max(s^+, s^-)$.

6.2.5 Implementation Details

We implemented our algorithms in Java, and we used the Apache Lucene library [2] to build the indexes and queries. We construct two queries of type `BooleanQuery`, one for T_Q^- and another for T_Q^+. They are then combined in a single `DisjunctionMaxQuery` [111], which picks the top-k tables with maximum value of either T_Q^- or T_Q^+. This allows both queries to be efficiently processed together using Lucene's implementation of the Block-Max WAND algorithm [60].

6.2.6 Theoretical Analysis

We provide a theoretical analysis to justify our heuristic and clarify the assumptions behind it. We analyze the s^+ and s^- scores and show that the estimator s^+/n provides a reasonable estimate for $N^+/N^\cap$. More formally, we show that:

Lemma 2. *The following bounds hold for a score s^+ computed using the QCR hashing scheme and a sketch of size n:*

$$\frac{N^+}{N^\cap}\rho \leq \mathbb{E}\left[\frac{s^+}{n}\right] \leq 2\frac{N^+}{N^\cap}\rho \tag{6.4}$$

where $\rho = \frac{N^\cap}{N^\cup}$ is the Jaccard similarity between K_Q and K_C.

To compare QCR with the indexing approach from Chapter 4, which only uses CSK sketches, we also extend our analysis to the scores produced by the CSK method (as both scores are counts of hash collisions). Before presenting our analysis, we describe CSK.

The CSK indexing scheme. While the QCR approach uses the set of hashes computed using Equation 6.3, CSK uses hashes derived only from the join keys. More specifically, it recommends constructing correlation sketches for candidate tables, and then indexing them using their set of hashed keys $T_C = \{h(k) \in \mathcal{S}_C\}$. At query time, it constructs a sketch $\mathcal{S}_Q$ for the query table, which is used to create the query term set $T_Q = \{h(k) \in \mathcal{S}_Q\}$. The queries find the top-k tables with the highest overlap of hashed keys $s^\cap = |T_Q \cap T_C|$. This step retrieves highly joinable tables. The final step re-ranks the retrieved tables using the correlation estimate obtained using the complete sketches $\mathcal{S}_Q$ and $\mathcal{S}_C$, which then places correlated tables at the top of the ranked list. This indexing approach based on the hashed keys of CSK sketches was already presented informally in Chapter 4. In what follows, we present a more detailed analysis of $s^\cap$, and compare it with the score s^+ derived from QCR terms.

Proof of Lemma 2. Let $\mathcal{S}_A$ and $\mathcal{S}_B$ be the sketches computed for the tables $\mathcal{T}_A$ and $\mathcal{T}_B$, respectively. Moreover, let $U = \{k_i : h(k_i) \in \mathcal{S}_A \cup \mathcal{S}_B\} = \{k_1, k_2, ..., k_n, ..., k_{|\mathcal{S}_A \cup \mathcal{S}_B|}\}$ where i denotes the index of k_i in the order induced by the hashing function h. Let also T_A and T_B denote the set of hashes computed using either the QCR or the CSK strategies from $\mathcal{S}_A$ and $\mathcal{S}_B$, respectively. Finally, $T = \{t_1, ..., t_{|\mathcal{S}_A \cup \mathcal{S}_B|}\}$ denotes the set of hashes computed for each k_i. We can now define a collection of Bernoulli random variables that represent hash collisions between T_A and T_B:

$$u_i = \begin{cases} 1 & \text{if } t_i \in T_A \cap T_B \\ 0 & \text{otherwise.} \end{cases}$$

We will first analyze the expected value of each u_i independently. Then, we will calculate the sum $s = \sum_{i=1}^{|\mathcal{S}_A \cup \mathcal{S}_B|} u_i$ and analyze its expected value $\mathbb{E}[s]$. For convenience, we will also denote $s(a, b) = \sum_{i=a}^{b} u_i$.

Both scores s^+ and $s^\cap$ are summations as in s, with their only difference being their probability of collisions. Note that $t_i \in T_A \cap T_B$ if and only if the keys used to compute the hashes collide in their original sets *and* if the hashes are selected for inclusion in both sketches. For instance, in CSK the strategy, $t_i \in T_A \cap T_B$ iff $k_i \in K_A \cap K_B$ and $k_i \in \mathcal{S}_A$ and $k_i \in \mathcal{S}_B$.

The probabilities of the keys being included in the sketches (i.e., $\mathbb{P}\{k_i \in \mathcal{S}_A$ and $k_i \in \mathcal{S}_B\}$) depend on the sketch size n and are not uniform for the set of all i's. In fact, as proven in [16], $\mathbb{P}\{k_i \in \mathcal{S}_A$ and $k_i \in \mathcal{S}_B\} = 1$ for all $i \leq n$ when sketches have size n, given that $\mathbb{P}\{k_i \in \mathcal{S}_A\} = 1$ and $\mathbb{P}\{k_i \in \mathcal{S}_B\} = 1$. Thus, we divide the analysis in two cases: $\{u_1, ..., u_n\}$ and $\{u_{n+1}, ..., u_{|\mathcal{S}_A \cup \mathcal{S}_B|}\}$. First, note that s can be decomposed into the sum of the parts: $s = s(1, n) + s(n + 1, |\mathcal{S}_A \cup \mathcal{S}_B|)$. Then, due to linearity of

expectation, we have that:

$$\begin{aligned}
\mathbb{E}[s] &= \mathbb{E}[s(1,n) + s(n+1, |\mathcal{S}_A \cup \mathcal{S}_B|)] \\
&= \mathbb{E}[s(1,n)] + \mathbb{E}[s(n+1, |\mathcal{S}_A \cup \mathcal{S}_B|)] \\
&= \mathbb{E}\left[\sum_{i=1}^{n} u_i\right] + \mathbb{E}\left[\sum_{i=n+1}^{|\mathcal{S}_A \cup \mathcal{S}_B|} u_i\right] \\
&= \sum_{i=1}^{n} \mathbb{E}[u_i] + \sum_{i=n+1}^{|\mathcal{S}_A \cup \mathcal{S}_B|} \mathbb{E}[u_i]
\end{aligned}$$

Each u_i is a Bernoulli random variable, thus we know that $\mathbb{E}[u_i] = \mathbb{P}\{u_i = 1\}$. Let I denote the event that k_i is *included* in both sketches and C denote the event that a hash *collision* happens. Then, we have that: $\mathbb{P}\{u_i = 1\} = \mathbb{P}\{I \mid C\} \cdot \mathbb{P}\{C\}$. Using the facts that the probability of inclusion is 1 for $i \leq n$ and $\mathbb{P}\{C\}$ is constant for all i, we get:

$$\begin{aligned}
\mathbb{E}[s(1,n)] &= \sum_{i=1}^{n} \mathbb{P}\{I \mid C\} \cdot \mathbb{P}\{C\} \\
&= \sum_{i=1}^{n} 1 \cdot \mathbb{P}\{C\} \\
&= n\mathbb{P}\{C\}
\end{aligned} \tag{6.5}$$

For the remaining $i \in \{n+1, |\mathcal{S}_A \cup \mathcal{S}_B|\}$, we use the the fact that $\mathbb{P}\{I|C\}$ can be at most 1 to get the upper bound:

$$\begin{aligned}
\mathbb{E}[s(n+1, |\mathcal{S}_A \cup \mathcal{S}_B|)] &= \sum_{i=n+1}^{|\mathcal{S}_A \cup \mathcal{S}_B|} \mathbb{P}\{I \mid C\} \cdot \mathbb{P}\{C\} \\
&\leq \sum_{i=n+1}^{|\mathcal{S}_A \cup \mathcal{S}_B|} 1 \cdot \mathbb{P}\{C\} \\
&\leq n\mathbb{P}\{C\}
\end{aligned} \tag{6.6}$$

Combining Equations 6.5 and 6.6, we get that:

$$n\mathbb{P}\{C\} \leq \ \mathbb{E}[s] \ \leq n\mathbb{P}\{C\} + n\mathbb{P}\{C\}$$

$$n\mathbb{P}\{C\} \leq \ \mathbb{E}[s] \ \leq 2n\mathbb{P}\{C\} \tag{6.7}$$

$$\mathbb{P}\{C\} \leq \mathbb{E}\left[\frac{s}{n}\right] \leq 2\mathbb{P}\{C\} \tag{6.8}$$

The final step is to obtain the collision probabilities for each hashing scheme. For CSK, we have that $\mathbb{P}\{C\} = \frac{N^{\cap}}{N^{\cup}}$ and whereas for QCR, $\mathbb{P}\{C\} = \frac{N^{+}}{N^{\cup}}$. Combining these with Equation 6.8, we obtain the following bounds:

$$\frac{N^{\cap}}{N^{\cup}} \leq \mathbb{E}\left[\frac{s^{\cap}}{n}\right] \leq 2\frac{N^{\cap}}{N^{\cup}} \quad \text{for CSK, and} \tag{6.9}$$

$$\frac{N^{+}}{N^{\cup}} \leq \mathbb{E}\left[\frac{s^{+}}{n}\right] \leq 2\frac{N^{+}}{N^{\cup}} \quad \text{for QCR.} \tag{6.10}$$

To get the results from Lemma 2, note that the following equality holds: $\frac{N^{+}}{N^{\cup}} = \frac{N^{+}}{N^{\cap}}\frac{N^{\cap}}{N^{\cup}}$.

□

6.2.7 Discussion

As shown earlier, using a QCR index is sufficient to retrieve correlated columns in a single step. In contrast to CSK, which retrieves tables that roughly maximize the Jaccard similarity (Equation 6.9), our QCR maximizes both the Jaccard similarity and the ratio $\frac{N^{+}}{N^{\cap}}$. Therefore, our method is a heuristic that uses the Jaccard similarity as a proxy to Jaccard containment.

Note, however, that our hashing scheme is complementary to the sketches proposed in (Chapter 4) and they can be used in a cascading fashion: one can first retrieve tables

using the QCR index, as described in Section 6.2.3, and then pass the retrieved candidates to another layer that re-ranks the candidate tables using estimates produced by CSK sketches (Chapter 4). At this re-ranking stage, one can estimate any correlation measure (e.g., Pearson's, Spearman's, and others) and the Jaccard Containment. Therefore, the re-ranking can optimize any weight combination as discussed in our Definition 5.

In addition to re-ranking using direct correlation estimates computed from sketches, additional information stored at indexing time can be used to further improve the ranking. For instance, scoring functions that take into account the risk of estimation error could be used to avoid placing false positives at the top of the ranked list (as done in Section 4.3.1). Note, however, that these improvements are only applicable to the re-ranking phase (i.e., they are orthogonal to the approach and experimental results we present in this chapter), and while they may improve final ranking, they cannot impact the retrieval recall (which is one to the main benefits of QCR indexes).

In our experiments (Section 6.3), we consider both the single-stage approach as well as the cascading approach. We show that QCR indexes improve both the ranking accuracy and recall compared to the CSK approach. Moreover, we show that the QCR-based hashing works well for retrieving candidate correlated tables according to a different correlation estimator such as Pearson's.

6.3 Experimental Evaluation

6.3.1 Experimental Setup

Dataset Collections. Our evaluation uses one synthetic and one real-world collection. Each of them is composed of two distinct sets of tables which we refer to as *query set* and *corpus set*. We describe these collections in more detail below.

(1) Synthetic Table Corpus (STC). In order to have more control of the data properties, we automatically generated a corpus containing tables with varying degrees of joinability and correlation. Our corpus generation method proceeds as follows. We first generate 1,000 table queries by generating unique keys and drawing numbers from a Gaussian distribution with parameters $\mathcal{N}(0, 1)$. Next, for each query, we synthetically generate 100 candidate tables with high correlation and 400 with low correlation with varying levels of Jaccard containment between their keys and the keys of the query table. Specifically, we draw the correlation level r, uniformly at random, from the the range $[1.0, 0.25]$ or $[-1.0, -0.25]$ for high-correlation tables, and from the range $(0.25, -0.25)$ for low-correlation tables. The Jaccard Containment is drawn from random and uniformly from $[0.1, 1.0]$. The final table collection includes 1,000 queries and 500,000 candidate tables, totaling approximately 250 GB of storage.

(2) NYC Open Data (NYC). The tables from this dataset contain data published by New York City agencies and their partners [121]. We used a snapshot that included 1,505 different CSV files with a varying number of columns. From each file, we generated 289,487 two-column tables (i.e., $\langle K_C, C \rangle$) by extracting all pairs of categorical and numerical data columns from each file. A brute-force approach to estimate join-correlations between all pairs of these tables would require over 41 billion join and correlation computations. To generate a query set, we randomly selected 1,000 tables. The remaining tables are assigned to the corpus set.

Evaluation Metrics. Ideally, table retrieval approaches should be able to find the largest number possible of correlated tables and, at the same time, place highly correlated and highly joinable tables at the top of the list. To measure different aspects of retrieval quality, we use the following evaluation metrics:

(1) Normalized Cumulative Gain (nDCG) [97]: measures the ability to place highly

relevant items at the top of the ranking. As a relevance measure, we use the *actual* absolute Pearson's correlation ($|r|$) between the numerical columns of the query and the candidate tables after a full table join. Therefore, nDCG values assess the retrieval quality with respect to the correlation-only objective ($\alpha_r = 1$ and $\alpha_j = 0$). We report the nDGC at positions 5, 10, and 50 of the ranked list (referred to as nDCG@5, nDCG@10, nDCG@50, respectively).

(2) Recall: measures the percentage of relevant tables retrieved relative to the total of relevant tables. Recall is also computed with respect to correlation $|r|$ only (i.e., $\alpha_j = 0$, $\alpha_r = 1$)). We report the recall considering different correlation levels as relevant: $|r| > 0.25$, $|r| > 0.50$, and $|r| > 0.75$. In order to compute the recall, we pooled all retrieved candidate tables by merging the lists associated with all retrieval strategies, and then reported the fraction of relevant tables retrieved by each specific retrieval strategy compared to all pooled tables.

(3) Average Jaccard Containment (Avg. JC) Similarity: to measure the ability to prioritize highly joinable tables, we report the average JC similarity at different positions of the ranked list (i.e., $\alpha_j = 1$, $\alpha_r = 0$). The JC similarity is computed over the join keys of the complete tables, i.e., $JC(K_Q, K_C) = |K_Q \cap K_C|/|K_Q|$, where K_Q and K_C are the sets of join keys of the original tables $\mathcal{T}_Q$ and $\mathcal{T}_C$ respectively. We report the average JC at positions 5, 10, and 50.

(4) Average Weighted Means (AAM, AGM, AHM): To evaluate queries with respect to weight preferences (Definition 5), we compute the weight w for the candidate table at each position in the ranked list, and then calculate the average from the first position up to a maximum position i. We use as combination functions the arithmetic, geometric, and harmonic means (denoted as AAM@i, AGM@i, and AHM@i, respectively).

(5) Harmonic Mean (HM): We also use the harmonic mean to evaluate how good each

retrieval method is with respect to multiple metrics. The HM is defined as $HM(a,b) = \frac{2ab}{a+b}$, where a and b are scores for two different evaluation metrics (e.g., nDCG, Recall, or Avg. JC).

All these metrics are in the interval $[0,1]$, and larger values are preferred over smaller values. We use the HM to combine metrics because it is intuitive and equivalent to the well-known F_1-*score*, which also uses the HM to quantify the trade-offs between precision and recall in binary classification tasks. When compared to the arithmetic (AM) and the geometric (GM) means, the HM is strictly smaller. This means that HM scores are "harsher" than if we were using the GM to evaluate an equal-weight linear combination of Avg. JC and nDCG.

Baselines and Parameter Settings. We compare our approach against the CSK approach from [136] under multiple parameter settings. As discussed in Section 6.2.6, this type of index can yield two baselines: the first, which retrieves and ranks tables according to the score $s^{\cap}$ (referred to as *CSK-Overlap*), roughly optimizes for finding tables with high joinability ($\alpha_r = 0$, $\alpha_j = 1$); the second, which ranks retrieved tables using sketch estimates (referred to as *CSK-Correlation*), optimizes the final ranking for correlations ($\alpha_r = 1$, $\alpha_j = 0$).

Similarly, we consider two possible ranking strategies for our QCR indexes: ranking tables based on the overlap of the QCR keys (namely, *QCR-Overlap*), which simultaneously prioritizes joinability and correlations ($\alpha_r = 1$, $\alpha_j = 1$); and a second strategy that re-ranks the retrieved tables using estimates, computed using the sketches (namely, *QCR-Correlation*). While *QCR-Correlation* roughly optimizes for both correlation and joinability in the first step, the re-ranking step optimizes for correlation ($\alpha_r = 1$, $\alpha_j = 0$).

Note that the re-ranking strategies (*CSK-Correlation* and *QCR-Correlation*) incur a small additional computational overhead, as they need to load the sketches from the storage, compute estimates, and then re-order the table candidates using the Pearson's correlation estimates. They also require an additional storage overhead for storing the sketches. For a collection with d documents and sketches of size n, they need $d * n * (\texttt{sizeof}(h(k)) + \texttt{sizeof}(c))$ bytes in addition to storage required for the inverted index.

Besides the aforementioned index types and ranking strategies, we also evaluate the effect of various parameters such as sketch size (the larger the sketch size, the larger the amount of storage space needed and the larger the index size), and the number of candidate tables retrieved (top-k). These parameters are applicable to both QCR and CSK indexes.

6.3.2 Retrieval of Highly Correlated Tables

Our first experiment focuses on retrieval quality with respect to correlations ($\alpha_r = 1$, $\alpha_j = 0$) (same as in [136]). We built an inverted index for several combinations of index parameter settings and data collections, and then used all tables in the *query sets* to issue queries against the index. We report the evaluation metric scores for the NYC and STC collections in Tables 6.1 and 6.2 respectively.

6.3.2.1 Ranking Accuracy

The results show that our QCR-based methods significantly improve over the baseline methods. The top performing method, QCR-Correlation, substantially increases ranking quality in terms of nDCG scores across all possible parameter settings, with particularly good results at the top-5. We note also that the QCR-Overlap method achieves

Table 6.1: Ranking scores for different index and ranking parameters on the NYC Open Data (NYC) collection.

Parameters				**nDCG**			**Recall**			**Harmonic Mean (nDCG, Recall)**			
n	top-k	index	ranking	@5	@10	@50	$r > .25$	$r > .50$	$r > .75$	$r > .5$ @10	$r > .5$ @50	$r > .75$ @10	$r > .75$ @50
256	50	CSK	Overlap	0.386	0.401	0.474	0.378	0.357	0.353	0.330	0.368	0.308	0.344
			Correlation	0.766	0.734	0.582	0.378	0.357	0.353	0.412	0.400	0.388	0.375
		QCR	Overlap	0.754	0.732	0.743	0.487	0.590	0.672	0.630	0.637	0.680	0.686
			Correlation	0.853	0.845	0.780	0.487	0.590	0.672	0.663	0.649	0.714	0.697
	100	CSK	Overlap	0.386	0.401	0.474	0.606	0.540	0.496	0.422	0.476	0.385	0.433
			Correlation	0.794	0.776	0.724	0.606	0.540	0.496	0.568	0.573	0.509	0.516
		QCR	Overlap	0.754	0.732	0.743	0.769	0.851	0.897	0.781	0.788	0.805	0.812
			Correlation	0.851	0.853	0.865	0.769	0.851	0.897	0.837	0.850	0.855	0.870
512	50	CSK	Overlap	0.380	0.397	0.472	0.378	0.357	0.353	0.326	0.367	0.305	0.343
			Correlation	0.776	0.743	0.585	0.378	0.357	0.353	0.413	0.400	0.390	0.375
		QCR	Overlap	0.759	0.737	0.747	0.488	0.596	0.678	0.636	0.642	0.685	0.691
			Correlation	0.866	0.858	0.787	0.488	0.596	0.678	0.671	0.655	0.723	0.704
	100	CSK	Overlap	0.380	0.397	0.472	0.603	0.542	0.493	0.417	0.475	0.379	0.430
			Correlation	0.808	0.790	0.732	0.603	0.542	0.493	0.573	0.575	0.510	0.515
		QCR	Overlap	0.759	0.737	0.747	0.770	0.862	0.905	0.788	0.795	0.810	0.817
			Correlation	0.868	0.870	0.876	0.770	0.862	0.905	0.849	0.860	0.866	0.879
1024	50	CSK	Overlap	0.381	0.397	0.472	0.380	0.356	0.354	0.326	0.366	0.306	0.343
			Correlation	0.781	0.749	0.586	0.380	0.356	0.354	0.413	0.399	0.392	0.376
		QCR	Overlap	0.763	0.742	0.749	0.490	0.594	0.680	0.637	0.642	0.689	0.694
			Correlation	0.870	0.860	0.788	0.490	0.594	0.680	0.670	0.654	0.723	0.706
	100	CSK	Overlap	0.381	0.397	0.472	0.603	0.540	0.496	0.416	0.473	0.379	0.430
			Correlation	0.815	0.797	0.736	0.603	0.540	0.496	0.571	0.573	0.513	0.517
		QCR	Overlap	0.763	0.742	0.749	0.772	0.863	0.909	0.791	0.797	0.813	0.820
			Correlation	0.873	0.874	0.879	0.772	0.863	0.909	0.852	0.862	0.870	0.883

nDCG scores that are very close to the best-performing approach (QCR-Correlation), suggesting that our indexing and retrieval approach provides scores that are well-correlated with the Pearson's correlation, even though the QCR estimator is only a crude estimator of this coefficient.

The improvements of QCR-Correlation over CSK-Correlation are due to the base QCR retrieval strategy, which makes more correlated tables available to the re-ranking

Table 6.2: Ranking scores for different index and ranking parameters on the Synthetic Table Corpus (STC) collection.

Parameters				**nDCG**			**Recall**			**Harmonic Mean (nDCG, Recall)**			
n	top-k	index	ranking	@5	@10	@50	$r > 0.25$	$r > 0.50$	$r > 0.75$	$r > .50$ @10	$r > .50$ @50	$r > .75$ @10	$r > .75$ @50
256	50	CSK	Overlap	0.133	0.137	0.196	0.234	0.208	0.188	0.149	0.199	0.138	0.184
			Correlation	0.839	0.708	0.373	0.234	0.208	0.188	0.319	0.265	0.290	0.242
		QCR	Overlap	0.921	0.900	0.815	0.747	0.788	0.822	0.839	0.801	0.857	0.817
			Correlation	0.994	0.987	0.847	0.747	0.788	0.822	0.875	0.816	0.895	0.833
	100	CSK	Overlap	0.133	0.137	0.196	0.471	0.420	0.379	0.190	0.263	0.182	0.252
			Correlation	0.936	0.887	0.582	0.471	0.420	0.379	0.567	0.486	0.523	0.453
		QCR	Overlap	0.921	0.900	0.815	0.931	0.941	0.951	0.919	0.873	0.924	0.877
			Correlation	0.998	0.996	0.957	0.931	0.941	0.951	0.967	0.949	0.972	0.954
512	50	CSK	Overlap	0.133	0.140	0.197	0.235	0.208	0.188	0.151	0.200	0.140	0.184
			Correlation	0.840	0.709	0.374	0.235	0.208	0.188	0.319	0.265	0.290	0.242
		QCR	Overlap	0.925	0.903	0.835	0.776	0.814	0.842	0.854	0.823	0.869	0.837
			Correlation	0.995	0.989	0.865	0.776	0.814	0.842	0.891	0.838	0.907	0.852
	100	CSK	Overlap	0.133	0.140	0.197	0.470	0.419	0.377	0.193	0.264	0.186	0.252
			Correlation	0.936	0.887	0.582	0.470	0.419	0.377	0.566	0.486	0.521	0.452
		QCR	Overlap	0.925	0.903	0.835	0.957	0.964	0.969	0.932	0.894	0.935	0.896
			Correlation	0.999	0.998	0.973	0.957	0.964	0.969	0.980	0.969	0.983	0.971
1024	50	CSK	Overlap	0.135	0.141	0.197	0.235	0.209	0.188	0.152	0.201	0.140	0.185
			Correlation	0.841	0.710	0.374	0.235	0.209	0.188	0.320	0.266	0.290	0.242
		QCR	Overlap	0.927	0.906	0.849	0.798	0.832	0.860	0.866	0.840	0.881	0.853
			Correlation	0.995	0.991	0.879	0.798	0.832	0.860	0.903	0.854	0.919	0.868
	100	CSK	Overlap	0.135	0.141	0.197	0.471	0.420	0.377	0.194	0.264	0.186	0.253
			Correlation	0.936	0.888	0.583	0.471	0.420	0.377	0.567	0.487	0.522	0.452
		QCR	Overlap	0.927	0.906	0.849	0.976	0.981	0.986	0.942	0.910	0.944	0.912
			Correlation	0.999	0.999	0.985	0.976	0.981	0.986	0.990	0.983	0.992	0.985

strategy, allowing the re-ranking phase to place more relevant tables at the top of the ranked list. In addition, because the QCR index tends to return items that are highly joinable (as shown in Section 6.3.3), the accuracy of the correlation estimates produced by the sketches might also significantly improve: the more joinable the sketches, the larger the sample size for correlation estimation.

Note also that while it may seem that the gap between CSK and QCR is not very large for top-5 results, CSK is expected to lead to the discovery of correlated tables only *eventually*. The event of finding a correlated columns when optimizing for JC is close to random [136]. Thus, the probability of such events is highly dependent on the distribution of the number of correlated tables in the collection. In large collections with very few correlated tables, it will be much harder to find correlated tables using CSK index than in smaller collections.

For instance, notice the big change in recall between the STC and NYC collections, which have different underlying generating processes. In the STC collection which has approximately 100 tables with $r > 0.25$ and 400 tables with $r < 0.25$ for each query, QCR is able to retrieve twice as many tables as CSK (recall of 93.1% compared to 47.1%, respectively) for $k = 100$ and $n = 256$. For smaller $k = 50$, the difference is more than 3 times larger (74.7% compared to 23.4%).

6.3.2.2 Recall

The results also show that QCR indexes dramatically improve the recall of correlated columns. A particularly interesting trend is that, while retrieving columns by overlap of correlation sketch keys (CSK-Overlap and CSK-Correlation), the recall progressively decreases as we increase the correlation level. The opposite happens for the QCR index: the recall becomes better for higher correlation levels. This confirms that *QCR indexes are particularly good at retrieving highly-correlated tables* ($r > 0.75$).

Another interesting result is that retrieving a longer list of candidate tables yields better results than increasing the sketch sizes. We can see this, for instance, by comparing the Recall scores obtained by the QCR index in the NYC collection: retrieving the top-100 tables using a sketch size of $n = 256$ leads to scores in the range $[0.769, 0.897]$, while

retrieving top-50 tables with $n = 1024$ only leads to scores in the range $[0.490, 0.680]$. Note that this is a two-fold increase in the top-k compared to a four-fold increase in the sketch size n. This suggests that *increasing the number top-k retrieved tables has a higher impact on recall than increasing the sketch size n.* As we will show in our efficiency evaluation, this is particularly good because an increase in the sketch size results in a larger number of query terms, which has a bigger impact on query processing times than increasing the number of top-k results.

The results also indicate that *QCR indexes are more space-efficient than CSK indexes*: a comparison of different sketch sizes ($n = 256$ vs. $n = 1024$) for the same number of top-k tables (top-k=100) shows that QCR attains better recall with smaller sketches. For example, the best recall for $r > 0.75$ attained by CSK is 0.496 (with the settings $n = 1024$ and $k = 100$), whereas QCR is able to attain a higher recall of 0.897 (with $n = 256$ and $k = 100$). This suggests that *QCR indexes need less than $1/4$ of the storage size needed by CSK indexes* (due to smaller sketches) to achieve the same recall.

6.3.2.3 Overall Ranking

Besides nDCG and Recall, we also report the harmonic mean of nDCG and Recall scores for different ranked list positions and correlation levels in Tables 6.1 and 6.2. These results, along with results from Sections 6.3.2.1 and 6.3.2.2, confirm that the QCR-based retrieval strategies are able to achieve a better overall ranking quality using smaller sketch sizes.

6.3.3 Balanced Retrieval of Correlated & Joinable Tables

So far, we discussed ranking quality in terms of accuracy and recall. We now consider the ability of our QCR index to place tables that are simultaneously highly correlated

Table 6.3: Average weighted means at different rank positions for different parameters on the NYC Open Data (NYC) collection.

n	top-k	index	ranking	AAM@5	AAM@10	AAM@k	AGM@5	AGM@10	AGM@k	AHM@5	AHM@10	AHM@k
512	50	CSK	Overlap	0.185	0.181	0.159	0.093	0.090	0.075	0.069	0.066	0.053
			Correlation	0.296	0.266	0.159	0.135	0.123	0.075	0.099	0.089	0.053
		QCR	Overlap	0.306	0.280	0.226	0.139	0.125	0.094	0.101	0.089	0.063
			Correlation	0.326	0.306	0.226	0.140	0.130	0.094	0.099	0.091	0.063
	100	CSK	Overlap	0.185	0.181	0.146	0.093	0.090	0.066	0.069	0.066	0.045
			Correlation	0.301	0.276	0.146	0.129	0.119	0.066	0.091	0.084	0.045
		QCR	Overlap	0.306	0.280	0.201	0.139	0.125	0.081	0.101	0.089	0.052
			Correlation	0.320	0.303	0.201	0.128	0.121	0.081	0.087	0.082	0.052

and joinable at the top of the ranked list. For this evaluation, we use as w the weighted mean of the Jaccard Containment (j) and the absolute Pearson's correlation (r). We compute w for the candidate table at each position in the ranked list and then calculate the average from the first position up to a maximum position i. We denote the arithmetic, geometric, and harmonic means as AAM@i, AGM@i, and AHM@i, respectively. The absence of @i means that the average is over all top-k retrieved tables. The results are reported in Table 6.3 (we only show $n = 512$ because other settings are similar). They confirm that QCR indexes lead to better performance than CSK indexes, specially when the size of the ranked list grows. Also, the correlation sketch estimates improve the performance regardless of the index, but the best results are achieved when QCR index is used because it makes more correlated tables available for the re-ranking step.

We also computed the Average Jaccard Containment (Avg. JC) attained at different positions of the ranked list, as well as the harmonic mean between the Avg. JC and nDCG. The results are reported in Tables 6.4 and 6.5 (we also include nDCG values for an easier comparison). We only report scores for sketch size $n = 512$ and for queries that retrieve the top-100 candidate tables. However, other settings led to similar results.

Table 6.4: Avg. JC scores on the STC table collection.

	nDCG		Avg. JC		HM(nDCG, Avg. JC)	
	@5	@50	@5	@50	@5	@50
CSK-Overlap	0.133	0.197	0.994	0.954	0.215	0.322
CSK-Correlation	0.936	0.582	0.908	0.908	0.921	0.707
QCR-Overlap	0.925	0.835	0.925	0.882	0.924	0.857
QCR-Correlation	0.999	0.973	0.749	0.827	0.854	0.894

Table 6.5: Avg. JC scores on the NYC table collection.

	nDCG		Avg. JC		HM(nDCG, Avg. JC)	
	@5	@50	@5	@50	@5	@50
CSK-Overlap	0.303	0.374	0.223	0.190	0.173	0.185
CSK-Correlation	0.622	0.570	0.180	0.175	0.216	0.209
QCR-Overlap	0.582	0.570	0.218	0.185	0.239	0.213
QCR-Correlation	0.658	0.666	0.170	0.168	0.208	0.210

As expected, the CSK-Overlap strategy is the best performing in terms of Avg. JC in both table collections. Moreover, while the QCR-Correlation is the best strategy in terms of nDCG, its Avg. JC is considerably lower than other methods. This is not surprising as this strategy only uses the correlation estimate in the re-ranking stage. In contrast, QCR-Overlap is able to maintain a good balance between correlation and joinability: its Avg. JC scores are not too far below when compared to the CSK-Overlap strategy, nor are its nDCG scores. As a result, *QCR-Overlap is able to obtain the best balance between the two metrics* as confirmed by their harmonic mean.

Note also the difference between the overall Average JC scores for different table collections: Avg. JC tends to be higher for the synthetic collection (STC) compared to smaller values in the NYC collection. This difference is due to the underlying data generation process of the collections. In the STC collection, we generate the tables in such a way that there are always joinable candidates for every query. In contrast, in

Table 6.6: Running time for different parameter settings along with their ranking scores on the NYC collection.

					nDCG/Rec	**nDCG/JC**	**Time**
	index	ranking	top-k	n	@10, $r > 0.75$	@10	$Avg.$
1	QCR	Correlation	100	1024	0.870	0.136	28.554
2	QCR	Correlation	100	512	0.866	0.132	19.704
3	QCR	Correlation	100	256	0.855	0.127	13.944
4	QCR	Overlap	100	1024	0.813	0.155	27.763
5	QCR	Overlap	100	512	0.810	0.153	19.492
6	QCR	Overlap	100	256	0.805	0.151	13.568
7	QCR	Correlation	50	512	0.723	0.148	18.431
8	QCR	Correlation	50	1024	0.723	0.150	26.963
9	QCR	Correlation	50	256	0.714	0.144	12.940
10	QCR	Overlap	50	1024	0.689	0.155	26.689
11	QCR	Overlap	50	512	0.685	0.153	18.102
12	QCR	Overlap	50	256	0.680	0.151	12.721
13	CSK	Correlation	100	1024	0.513	0.139	17.033
14	CSK	Correlation	100	512	0.510	0.136	12.488
15	CSK	Correlation	100	256	0.509	0.131	9.837
16	CSK	Correlation	50	1024	0.392	0.148	14.132
17	CSK	Correlation	50	512	0.390	0.146	10.700
18	CSK	Correlation	50	256	0.388	0.143	8.655

the NYC collection, where we select query tables randomly from the data, we notice a significant skew in the distribution of JC for the retrieved tables, with some query tables having few joinable columns. Nonetheless, we can see that the general trends in the results are still the same, with QCR-Overlap attaining the best scores in terms of HM(nDCG, Avg. JC).

6.3.4 Runtime Performance

To evaluate performance, we executed all queries in the query set and measured their total execution time, including the time to create sketches (and terms T_Q^+ and

T_Q^-) for the query table, processing the query, reading the candidate tables' sketches and re-ranking the results based on sketch correlation estimates (when applicable). Given that Lucene's implementation makes heavy use of caching and memory mapping mechanisms to speed up query execution, we ran all queries 5 times and discarded the first execution. We omit query times for the synthetic collection. While their query times are higher due to their query distribution size, we observed similar results.

In Table 6.6, we report the average query time for all queries in the NYC collection along with the ranking metric scores obtained by the same parameter settings. Results are sorted by the harmonic mean between nDCG@10 and Recall at $r > 0.75$, in decreasing order, to make it easier to find the running time of the best-performing settings. In general, we can see that the higher the sketch size n and the number of retrieved candidates, the higher the running time. Moreover, we can see that, for a fixed top-k and sketch size n, the query times for QCR methods are roughly twice as high as for their CSK counterparts. This is not surprising, since QCR queries need to process twice as many terms (in order to retrieve positive and negative correlations) as CSK queries. Another interesting result is that even the settings of the QCR methods that use the smallest sketch sizes, which are as good as the best CSK methods, have running times that are either lower or comparable to the best CSK approaches. This suggests that *QCR strategies are more efficient than CSK's for a fixed retrieval quality level.*

To better visualize the trends in these results, consider the plot in Figure 6.4. In this plot, the best methods are the ones located closer to the top-left corner, i.e., the region of better metric scores (top) and lower running time (left). Here, it is easy to see that increasing the number of retrieved candidate tables from 50 (○) to 100 (□) significantly improves retrieval quality, at only a small runtime cost. Conversely, increasing the sketch size incurs a significant runtime penalty. This suggests that in order to improve

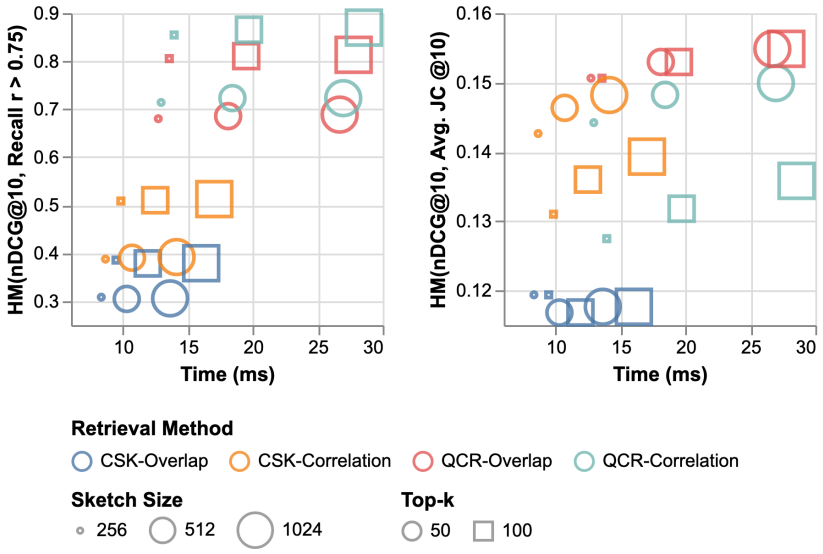


Figure 6.4: Runtime versus retrieval quality scores for different parameter settings.

the results, it is more resource-efficient to increase the length of the retrieved list than the sketch size.

Chapter 7

Applications to Machine Learning Model Improvement

In this chapter, we present two use cases that demonstrate the application of our methods for discovering useful ML model features efficiently. The first use case (Section 7.1) shows an application of indexing and querying large data lakes for the discovery of features across many different tables for improving machine learning models. In the second use case (Section 7.2), we illustrate an application to real-world biomedical data of searching ML features in wide tables (i.e., tables with a large number of columns) for predicting gene mutations relevant to cancer prognosis.

7.1 Feature Discovery on Large Data Lakes

In this use case, we evaluate different techniques introduced in this dissertation in the task of feature discovery for improving the performance of machine learning models. To do so, we use the YADL datalake [30], which has been specifically designed for benchmarking this task. YADL is based on the YAGO knowledge base [112] and

provides a controlled environment for testing different methods. For this experiment, we use the *YADL Wordnet* data lake, which contains 32,103 tables with an average of 287,134.33 rows per table [30]. As machine learning problems, we considered the following three regression tasks that are included in YADL: (1) *Company Employees*: predict the number of employees in a company; (2) *US Elections*: predict the fraction of votes by party in each US county during the 2020 elections; (3) *US Accidents*: predict the number of accidents by US county in the year 2019.

We start the experiments with versions of these tables that only contain the *target* and *join key* columns, i.e., all features used to train the model are discovered from the data lake. As indexing methods, we use both indexes based on CSK sketches join key values as well as QCR index terms (see Chapter 6). For simplicity, we only indexed features with numerical data, used the PM1 bootstrap estimator for Pearson's correlation, and used the aggregate function `FIRST`, which only stores the first element seen of each repeated join key (but it would be useful to evaluate the impact of other variables such as different data types, and aggregate functions, and correlation/MI estimators in future work). To discover features, we issue queries against an index to retrieve the top 500 columns, which are then used to generate the final ranking using one of the following ranking approaches:

- **KEY:** This approach uses original scores computed during the set overlap search phase. For CSK indexes, this means the overlap between the join keys; for QCR indexes, this means the overlap between QCR index terms, which maximizes both correlation and joinability jointly (see Section 6 for details);
- **BAL:** This approach is a "balanced" approach that prioritizes equally correlation and joinability by reranking the retrieved features using the formula $j * r$ where j is the Jaccard containment estimate computed using the k-minimum values

included in the correlation sketch and r is the correlation estimate computed using a correlation estimator applied over the sketch join.

- **GRD:** This approach is a simple heuristic ranking diversification method that uses a greedy algorithm to reorder [140] the ranking taking into account the features that were selected previously. The goal is to place strongly correlated features at the top of the ranking while avoiding redundancy, i.e., avoiding features that are correlated with features that have been previously selected.

Finally, we train regression models using the CatBoost library [33] over a varying number of the top-ranked features discovered using each method. To evaluate model quality, we use standard random train-test splits and the MSE and R-squared metrics.

Figures 7.1 and 7.2 show the MSE and R-squared scores, respectively, attained by each combination of index and ranking methods. The results show that methods that combine the QCR or CSK indexes, along with the greedy diversification (*GRD*) approach often lead to the best results. This happens because, given that the YADL Wordnet data lake contains a small number of highly joinable tables, both of these indexing techniques can retrieve the best features in the initial retrieval phase.

We also note that the basic *BAL* approach usually does not place the best features in the top 5-10 positions because of the large amount of duplication in the YADL data lake. With *BAL* ranking, a highly correlated feature is usually followed by other features that are similar to the previously selected features. While they could be relevant in isolation, the presence of similar features earlier in the ranking makes them redundant. As seen in the plots, this issue is easily solved by a diversification approach that looks ahead for less redundant features in the ranked list.

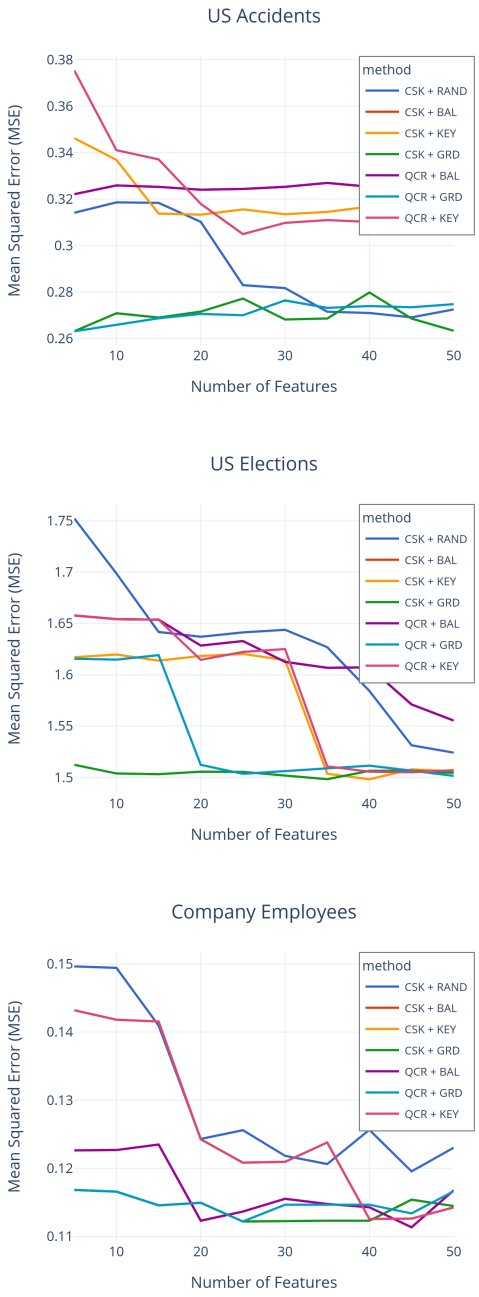


Figure 7.1: Mean Squared Error achieved by each combination of retrieval and ranking methods in different ML problems.

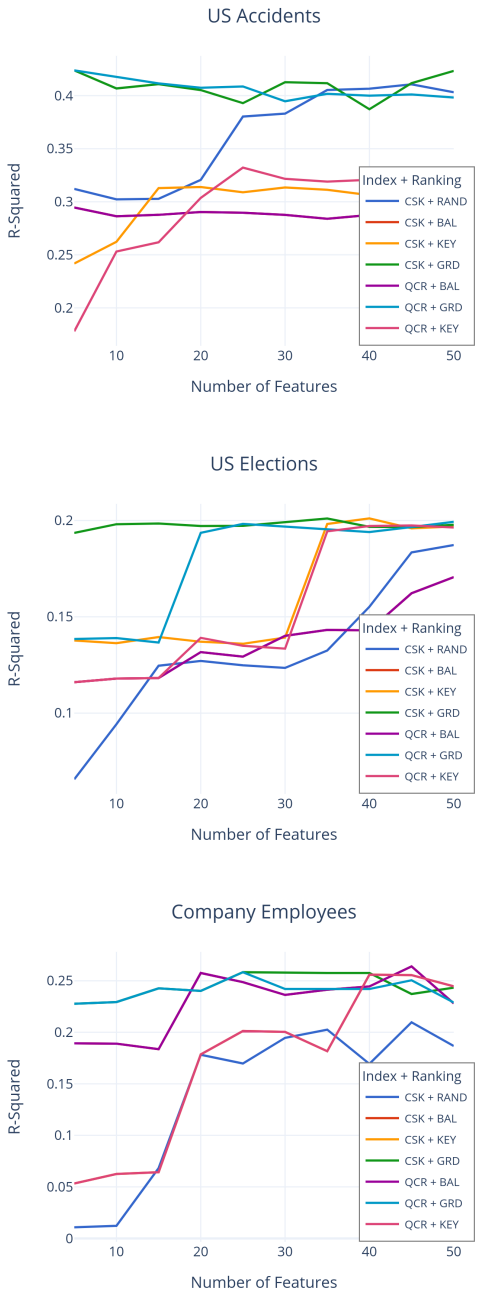


Figure 7.2: R-Squared scores achieved by each combination of retrieval and ranking methods in different ML problems.

7.2 Efficient Feature Ranking on Wide Tables

Even though not designed with this exact scenario in mind, here we show that a QCR index may also be useful in ranking features in wide tables. In this experiment, we aim to reproduce one of the results from Dou et al. [65], where the researchers built models that can predict gene mutations relevant to the prognosis of endometrial cancer. In this problem, the researchers already know the tables that contain the relevant features, however, there are over 10,000 different columns to search from.

Mutations in the CTNNB1 gene are associated with the risk of cancer recurrence [118, 104], and therefore useful for making treatment decisions. Gene sequencing is currently used to identify these mutations, however, insurance providers do not always cover its costs for low-grade tumors. Therefore, a more accessible and reliable diagnostic tool is needed for these cases. In Dou et al. [65], the researchers address this problem by building highly accurate predictive models that use as input only proteomics data (measurements of protein levels in a tissue sample) that are more readily available. Next, we investigate whether researchers could use the methods proposed in this dissertation to discover features efficiently and build models of comparable accuracy.

For this experiment, we use as input a query table containing 95 different individual samples (rows) and two columns that contain (1) a sample ID that can be used for joining tables, and (2) a *target column* storing binary data that indicates the mutation status of the CTNNB1 gene. As candidate features, we use a separate table that contains over 11.000 columns, which contain measurements for each protein detected in a tissue, in addition to an ID column that links these records to the same 95 samples from our target variable. The goal is to efficiently retrieve useful features from this table using a QCR index. To evaluate the accuracy of the built models, we use similar tables containing an

additional set of 138 samples from an independent cohort of patients. As in Section 7.1, we train a machine learning classification model using the CatBoost library [33]. For feature indexing and retrieval, we use a method that retrieves only the top 500 features using a QCR index and reranks them using Pearson's correlation estimates computed using the PM1 bootstrapping estimator.

Figure 7.3 shows the accuracy of different models trained with the top-k ranked features, varying k from 5 to 100. The results show that models trained with the $k = 10$ and $k = 25$ features are sufficient to achieve good performance. The ROC curves have a similar shape to the models presented in Dou et al.[65], and the area under the curve (AUC) scores obtained by these models are similar. In particular, one of our best-performing models that uses only 25 features, achieves AUC=0.95 which is only 0.02 points away from the best-performing model (AUC=0.97) from Dou et al.[65]. The exact source of this difference could be due to many reasons, including the choice of features, learning algorithms, hyper-parameters, and random variance of the models.

When closely analyzing the selected features, we notice a significant overlap between the features selected by our method and the features used by the models of Dou et al. [65]. Specifically, when we consider the top 10 features of the ranking, we notice that 6 of them were also used by at least two of the models from Dou et al. [65], and 7 are used by at least one model. When we look at the top 25 features, the number of features used by at least one model grows to 9. This suggests that our approach can find the most relevant features in a dataset with a large number of features.

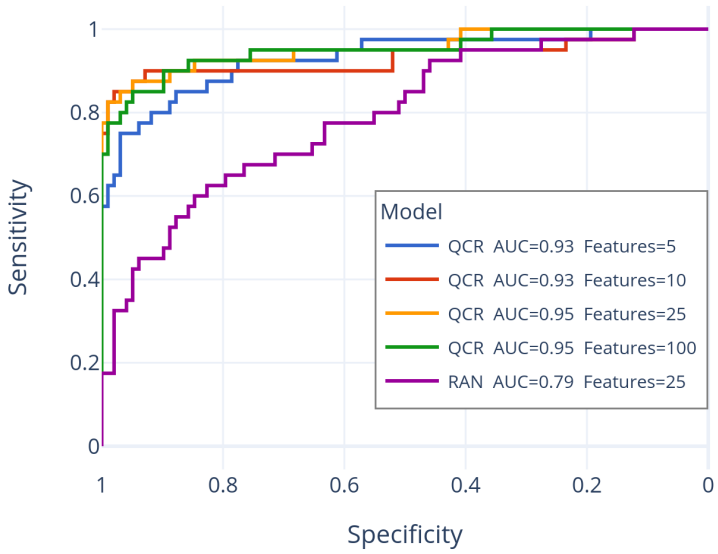


Figure 7.3: ROC curves for different models trained using features automatically selected using a QCR index and a random choice of features (RAN). A random choice of 25 features achieves only AUC=0.79. In contrast, our best QCR-based models achieve an AUC score of up to 0.95. This score matches some models from Dou et al. [65] and is only 0.02 points away from their best-performing model, which achieves AUC=0.97. The exact score difference could be attributed to the choice features or other variables such as a different choice of learning algorithm and its hyper-parameters (since we did not test multiple algorithms or perform hyperparameter tuning).

Chapter 8

Conclusion

In this dissertation we proposed multiple methods and algorithms, including the idea of (weighted) join-correlation queries, multiple sampling-based sketches for estimating statistics over table joins, and new indexing techniques for correlated data retrieval. These methods open opportunities for more effective data discovery, analysis, and integration, with implications spanning diverse applications, from automatic feature selection over large data repositories (e.g., data lakes) for improving machine learning model accuracy to enhancing dataset search engines and data catalogs. In the remainder of this chapter, we summarize our technical contributions (Section 8.1) and outline some limitations and future research directions (Section 8.2).

8.1 Summary of Contributions

In Chapter 4, we proposed join-correlation queries, an approach for efficiently answering this new type of query that discovers correlated tables, and a sampling algorithm to build data sketches (which we refer to as CSK sketches) for estimating correlations over unjoined tables without the need of a full join computation. We also

developed confidence interval bounds for the accuracy of these correlation estimates and used them to design functions to rank the discovered columns. The applicability of these sketches extends beyond correlations, having broader utility in estimating various statistics and data relationships over table joins.

In Chapter 5, we extended CSK sketches in two other directions: (1) we introduced a novel tuple-based sampling strategy that allows building sketches (namely TUPSK sketches) that properly handle join keys with repeated values and (2) carried out an experimental study of these sketches along with different estimators to efficiently estimate mutual information across unjoined tables. Our results show the effectiveness of TUPSK sketches for approximating mutual information values accurately across different data types. Finally, we believe this sampling approach is of independent interest and may have applications to other problems, such as join size estimation.

In Chapter 6, we generalized join-correlation queries to support user-defined weights and introduced another approach for indexing tables (based on a novel QCR hashing scheme) that allows answering (weighted) join-correlation queries even more efficiently. Our results show that this approach leads to enhanced retrieval accuracy and recall.

8.2 Future Directions

This dissertation leaves many open research questions that can be explored in future work. Next, we point out some directions we believe are worth being pursued.

- *Applications:* Since the publication of Auctus, many follow-up papers have addressed the problem of relational data augmentation for improving machine learning models [39, 110, 96]. However, to the best of our knowledge, none of these systems employ sketching techniques similar to the ones proposed in this

dissertation. It would be interesting to integrate and evaluate the impact of our algorithms in these systems. Moreover, it would be interesting to explore the possible application of sketches to new problems, such as federated feature selection [75] or monitoring of the quality of machine learning models in production.

- *Feature Selection*: In Chapter 7 we provided a preliminary evaluation of our algorithm for the task of feature selection. However, our experiments are far from exhaustive and there are many improvements to be pursued. For example, the diversification algorithm is expensive to compute, even when using sketches, due to its high computational complexity. In the area of web search engines, similar diversification algorithms have been proven to be NP-hard via reduction to the classic maximum coverage problem [140]. Therefore, it is worth investigating efficient and effective heuristics to address this problem in the context of data discovery. One possible direction is to explore the use of "conditional correlation measures" (e.g., conditional mutual information) that take into account the features that have been previously selected.

- *Indexing Methods for Categorical Data:* Our QCR indexing method is only applicable to numerical data. However, as discussed in Chapter 5, other data types like categorical variables also need to be supported as well.

- *Better Bounds for the Sketch Estimates*: From a theoretical point of view, there are still some open problems. For instance, the correlation bounds proposed in Chapter 4 are not tight and could be improved (e.g., using concentration inequalities for sampling without replacement [6, 152]). Moreover, we do not provide bounds for the MI estimates computed using sketches in Chapter 5. While there are known bounds for the empirical entropy computed using subsampling

and the MLE estimator [159, 37], it is unclear if their assumptions hold in our setting. Additionally, we are not aware of similar bounds for KSG-like estimators that are used for numerical data.

- *Different Types of Joins:* A common assumption throughout this dissertation is that the tables can be joined using an equi-join. However, in practice, there are other join types that need to be supported. For instance, joins can be done over spatial data attributes (which are often represented as a pair of latitude and longitude values). In this setting, joining can be done not only by comparing equal values but also by finding records with a maximum distance (e.g., using the Euclidean distance). How to build sketches and indexes that support efficient correlated data search in such settings is an open question.

Bibliography

[1] S. Acharya, P. B. Gibbons, V. Poosala, and S. Ramaswamy. Join synopses for approximate query answering. *SIGMOD Rec.*, 28(2):275–286, June 1999.

[2] Apache lucene. `https://lucene.apache.org/index.html`.

[3] M. Baak, R. Koopman, H. Snoek, and S. Klous. A new correlation coefficient between categorical, ordinal and interval variables with pearson characteristics. *Computational Statistics & Data Analysis*, page 107043, 2020.

[4] 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].

[5] Z. Bar-Yossef, T. S. Jayram, R. Kumar, D. Sivakumar, and L. Trevisan. Counting distinct elements in a data stream. In J. D. P. Rolim and S. Vadhan, editors, *Randomization and Approximation Techniques in Computer Science*, pages 1–10, Berlin, Heidelberg, 2002. Springer Berlin Heidelberg.

[6] R. Bardenet, O.-A. Maillard, et al. Concentration inequalities for sampling without replacement. *Bernoulli*, 21(3):1361–1385, 2015.

[7] O. Benjelloun, S. Chen, and N. Noy, editors. *Google Dataset Search by the Numbers*, 2020.

[8] 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)*, pages 1–9. IEEE, 2019.

[9] M. Bermudez-Edo, P. Barnaghi, and K. Moessner. Analysing real world data streams with spatio-temporal correlations: Entropy vs. pearson correlation. *Automation in Construction*, 88:87–100, 2018.

[10] K. J. Berry and P. W. Mielke Jr. A monte carlo investigation of the fisher z transformation for normal and nonnormal distributions. *Psychological Reports*, 87(3_suppl):1101–1114, 2000.

[11] A. Bessa, S. Castelo, R. Rampin, A. Santos, M. Shoemate, V. D'Orazio, and J. Freire. An ecosystem of applications for modeling political violence. In *Proceedings of the 2021 International Conference on Management of Data*, pages 2384–2388, 2021.

[12] 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*, pages 2384–2388, 2021.

[13] 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.

[14] A. Bessa, J. Freire, T. Dasu, and D. Srivastava. Effective discovery of meaningful outlier relationships. *ACM Transactions on Data Science*, 1(2):1–33, 2020.

[15] K. Beyer, R. Gemulla, P. J. Haas, B. Reinwald, and Y. Sismanis. Distinct-value synopses for multiset operations. *Commun. ACM*, 52(10):87–95, Oct. 2009.

[16] 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*, SIGMOD '07, pages 199–210, New York, NY, USA, 2007. ACM.

[17] A. J. Bishara and J. B. Hittner. Testing the significance of a correlation with non-normal data: Comparison of Pearson, Spearman, transformation, and resampling approaches. *Psychological Methods*, 2012.

[18] A. J. Bishara and J. B. Hittner. Reducing bias and error in the correlation coefficient due to nonnormality. *Educational and Psychological Measurement*, 75(5):785–804, 2015.

[19] A. J. Bishara and J. B. Hittner. Confidence intervals for correlations when data are not normal. *Behavior Research Methods*, 49(1):294–309, 2017.

[20] A. J. Bishara, J. Li, and T. Nash. Asymptotic confidence intervals for the pearson correlation via skewness and kurtosis. *British Journal of Mathematical and Statistical Psychology*, 71(1):167–185, 2018.

[21] C. I. Bliss et al. Statistics in biology. statistical methods for research in the natural sciences. *Statistics in biology. Statistical methods for research in the natural sciences.*, 1967.

[22] J. Boidol and A. Hapfelmeier. Fast mutual information computation for dependency-monitoring on data streams. In *Proceedings of the Symposium on*

Applied Computing, SAC '17, page 830–835, New York, NY, USA, 2017. Association for Computing Machinery.

[23] D. Bonett and T. A. Wright. Sample size requirements for estimating pearson, kendall and spearman correlations. *Psychometrika*, 65(1):23–28, 2000.

[24] A. Bowley. The standard deviation of the correlation coefficient. *Journal of the American Statistical Association*, 23(161):31–34, 1928.

[25] 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*, WWW '19, pages 1365–1375, New York, NY, USA, 2019. ACM.

[26] A. Z. Broder, D. Carmel, M. Herscovici, A. Soffer, and J. Zien. Efficient query evaluation using a two-level retrieval process. In *Proceedings of the twelfth international conference on Information and knowledge management*, pages 426–434, 2003.

[27] 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*, 13:27–66, 2012.

[28] K. E. Brown and D. A. Talbert. Heuristically reducing the cost of correlation-based feature selection. In *Proceedings of the 2019 ACM Southeast Conference*, ACM SE '19, page 24–30, New York, NY, USA, 2019. Association for Computing Machinery.

[29] P. Brown, P. J. Haas, J. Myllymaki, H. Pirahesh, B. Reinwald, and Y. Sismanis. Toward automated large-scale information integration and discovery. In *Data Management in a Connected World*, volume 3551 of *Lecture Notes in Computer Science*, pages 161–180. Springer, 2005.

[30] R. Cappuzzo, G. Varoquaux, A. Coelho, and P. Papotti. Retrieve, merge, predict: Augmenting tables with data lakes. *arXiv preprint arXiv:2402.06282*, 2024.

[31] 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*, 14(12):2791–2794, 2021.

[32] 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)*, pages 1190–1201, April 2019.

[33] Catboost – open-source gradient boosting library. `https://catboost.ai/`.

[34] G. Chandrashekar and F. Sahin. A survey on feature selection methods. *Computers & Electrical Engineering*, 40(1):16–28, 2014. 40th-year commemorative issue.

[35] A. Chapman, E. Simperl, L. Koesten, G. Konstantinidis, L.-D. Ibáñez, E. Kacprzak, and P. Groth. Dataset search: a survey. *The VLDB Journal*, 29(1):251–272, 2020.

[36] M. Charikar, S. Chaudhuri, R. Motwani, and V. Narasayya. Towards estimation error guarantees for distinct values. In *Proceedings of the Nineteenth ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems*, PODS '00, page 268–279, New York, NY, USA, 2000. Association for Computing Machinery.

[37] 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*, pages 274–286, 2021.

[38] Y. Chen and K. Yi. Two-level sampling for join size estimation. In *Proceedings of the 2017 ACM International Conference on Management of Data*, SIGMOD '17, page 759–774, New York, NY, USA, 2017. Association for Computing Machinery.

[39] 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*, 13(9), 2020.

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

[41] CKAN. The open source data portal software. `https://ckan.org/`.

[42] E. Cohen. Coordinated sampling. In *Encyclopedia of Algorithms*, pages 449–454. Springer New York, New York, NY, 2016.

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

[44] E. Cohen and H. Kaplan. Tighter estimation using bottom k sketches. *Proc. VLDB Endow.*, 1(1):213–224, Aug. 2008.

[45] R. Cohen, L. Katzir, and A. Yehezkel. A minimal variance estimator for the cardinality of big data set intersection. In *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '17, page 95–103, New York, NY, USA, 2017. Association for Computing Machinery.

[46] G. Cormode, M. N. Garofalakis, P. J. Haas, and C. Jermaine. Synopses for massive data: Samples, histograms, wavelets, sketches. *Foundations and Trends in Databases*, 4(1-3):1–294, 2012.

[47] S. Dahlgaard, M. B. T. Knudsen, and M. Thorup. Practical hash functions for similarity estimation and dimensionality reduction. In *Proceedings of the 31st International Conference on Neural Information Processing Systems*, NIPS'17, page 6618–6628, Red Hook, NY, USA, 2017. Curran Associates Inc.

[48] M. Daliri, J. Freire, C. Musco, A. Santos, and H. Zhang. Sampling methods for inner product sketching. *Proceedings of the VLDB Endowment*, 2024. (To appear).

[49] A. Dasgupta, K. J. Lang, L. Rhodes, and J. Thaler. A Framework for Estimating Stream Expression Cardinalities. In W. Martens and T. Zeume, editors, *19th International Conference on Database Theory (ICDT 2016)*, volume 48 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 6:1–6:17, Dagstuhl, Germany, 2016. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik.

[50] A. Dasgupta, K. J. Lang, L. Rhodes, and J. Thaler. A Framework for Estimating Stream Expression Cardinalities. In W. Martens and T. Zeume, editors, *19th International Conference on Database Theory (ICDT 2016)*, volume 48 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 6:1–6:17, Dagstuhl, Germany, 2016. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik.

[51] `https://www.darpa.mil/program/data-driven-discovery-of-models`, 2019.

[52] Data.gov: U.S. Government's open data. `https://www.data.gov/`.

[53] `https://pypi.org/project/datamart-profiler/`, 2021.

[54] The Dataverse Project. https://dataverse.org.

[55] Harvard Dataverse. https://dataverse.harvard.edu/.

[56] 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*, 5(1):1–12, 2004.

[57] J. C. de Winter, S. D. Gosling, and J. Potter. Comparing the pearson and spearman correlation coefficients across distributions and sample sizes: A tutorial using simulations and empirical data. *Psychological Methods*, 2016.

[58] D. Deng, R. C. Fernandez, Z. Abedjan, S. Wang, M. Stonebraker, A. K. Elmagarmid, I. F. Ilyas, S. Madden, M. Ouzzani, and N. Tang. The data civilizer system. In *8th Biennial Conference on Innovative Data Systems Research, CIDR 2017, Chaminade, CA, USA, January 8-11, 2017, Online Proceedings*. www.cidrdb.org, 2017.

[59] S. J. Devlin, R. Gnanadesikan, and J. R. Kettenring. Robust estimation and outlier detection with correlation coefficients. *Biometrika*, 62(3):531–545, 12 1975.

[60] S. Ding and T. Suel. Faster top-k document retrieval using block-max indexes. In *Proceedings of the 34th international ACM SIGIR conference on Research and development in Information Retrieval*, pages 993–1002, 2011.

[61] https://www.docker.com/, 2021.

[62] 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*, pages 3267–3271, 2022.

[63] G. Doquire and M. Verleysen. Feature selection with missing data using mutual information estimators. *Neurocomputing*, 90:3–11, 2012. Advances in artificial neural networks, machine learning, and computational intelligence (ESANN 2011).

[64] V. D'Orazio. Conflict forecasting and prediction. In *Oxford Research Encyclopedia of International Studies*. 2020.

[65] Y. Dou, L. Katsnelson, M. A. Gritsenko, Y. Hu, B. Reva, R. Hong, Y.-T. Wang, I. Kolodziejczak, R. J.-H. Lu, C.-F. Tsai, et al. Proteogenomic insights suggest druggable pathways in endometrial carcinoma. *Cancer cell*, 41(9):1586–1605, 2023.

[66] N. Duffield, C. Lund, and M. Thorup. Priority sampling for estimation of arbitrary subset sums. *J. ACM*, 54(6):32–es, Dec. 2007.

[67] B. Efron and R. Tibshirani. *An Introduction to the Bootstrap*. Chapman & Hall/CRC Monographs on Statistics & Applied Probability. Taylor & Francis, 1994.

[68] `https://www.elastic.co/`, 2021.

[69] M. Esmailoghli, J.-A. Quiané-Ruiz, and Z. Abedjan. Mate: multi-attribute table extraction. *Proceedings of the VLDB Endowment*, 15(8):1684–1696, 2022.

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

[71] R. Fagin, A. Lotem, and M. Naor. Optimal aggregation algorithms for middleware. *Journal of computer and system sciences*, 66(4):614–656, 2003.

[72] M. Ferdosi, A. Gholamidavoodi, and H. Mohimani. Measuring mutual information between all pairs of variables in subquadratic complexity. In S. Chiappa and R. Calandra, editors, *Proceedings of the Twenty Third International Conference on Artificial Intelligence and Statistics*, volume 108 of *Proceedings of Machine Learning Research*, pages 4399–4409. PMLR, 26–28 Aug 2020.

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

[74] P. Flajolet, É. Fusy, O. Gandouet, and F. Meunier. HyperLogLog: the analysis of a near-optimal cardinality estimation algorithm. In P. Jacquet, editor, *AofA: Analysis of Algorithms*, volume DMTCS Proceedings vol. AH, 2007 Conference on Analysis of Algorithms (AofA 07) of *DMTCS Proceedings*, pages 137–156, Juan les Pins, France, June 2007. Discrete Mathematics and Theoretical Computer Science.

[75] R. Fu, Y. Wu, Q. Xu, and M. Zhang. Feast: A communication-efficient federated feature selection framework for relational data. *Proceedings of the ACM on Management of Data*, 1(1):1–28, 2023.

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

[77] S. Ganguly, P. B. Gibbons, Y. Matias, and A. Silberschatz. Bifocal sampling for skew-resistant join size estimation. *SIGMOD Rec.*, 25(2):271–281, June 1996.

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

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

[80] H.-O. Georgii. *Stochastics: Introduction to Probability and Statistics*. De Gruyter, Berlin, Boston, 2012.

[81] 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].

[82] P. J. Haas, J. F. Naughton, S. Seshadri, and L. Stokes. Sampling-based estimation of the number of distinct values of an attribute. In *Proceedings of the 21th International Conference on Very Large Data Bases*, VLDB '95, page 311–322, San Francisco, CA, USA, 1995. Morgan Kaufmann Publishers Inc.

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

[84] M. Hall and L. Smith. Feature subset selection: a correlation based filter approach. In *Proceedings of International Conference on Neural Information Processing and Intelligent Information Systems*, 1998.

[85] M. A. Hall and L. A. Smith. Feature selection for machine learning: comparing a correlation-based filter approach to the wrapper. In *FLAIRS conference*, volume 1999, pages 235–239, 1999.

[86] H. Harmouch and F. Naumann. Cardinality estimation: An experimental survey. *Proc. VLDB Endow.*, 11(4):499–512, Dec. 2017.

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

[88] W. Hoeffding. Probability inequalities for sums of bounded random variables. *Journal of the American Statistical Association*, 58(301):13–30, 1963.

[89] P. Holmes. Correlation: From picture to formula. *Teaching Statistics*, 23(3):67–71, 2001.

[90] S. R. Hong, S. Castelo, V. D'Orazio, C. Benthune, A. Santos, S. Langevin, D. Jonker, E. Bertini, and J. Freire. Towards evaluating exploratory model building process with automl systems. *arXiv preprint arXiv:2009.00449*, 2020.

[91] X. Hu, A. Jung, and G. Qin. Interval estimation for the correlation coefficient. *The American Statistician*, 74(1):29–36, 2020.

[92] D. Huang, D. Y. Yoon, S. Pettie, and B. Mozafari. Joins on samples: a theoretical guide for practitioners. *Proceedings of the VLDB Endowment*, 13(4):547–560, 2019.

[93] M. Hutter and M. Zaffalon. Distribution of mutual information from complete and incomplete data. *Computational Statistics & Data Analysis*, 48(3):633–657, 2005.

[94] P. Indyk and A. McGregor. Declaring independence via the sketching of sketches. In *Proceedings of the Nineteenth Annual ACM-SIAM Symposium on Discrete Algo-*

rithms, SODA '08, page 737–745, USA, 2008. Society for Industrial and Applied Mathematics.

[95] Y. E. Ioannidis. The history of histograms (abridged). In *VLDB*, pages 19–30, 2003.

[96] 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)*, pages 84–88, 2022.

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

[98] 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*, SSDBM '15, New York, NY, USA, 2015. Association for Computing Machinery.

[99] D. Y. Kenett, X. Huang, I. Vodenska, S. Havlin, and H. E. Stanley. Partial correlation analysis: Applications for financial markets. *Quantitative Finance*, 15(4):569–578, 2015.

[100] A. Kipf, T. Kipf, B. Radke, V. Leis, P. A. Boncz, and A. Kemper. Learned cardinalities: Estimating correlated joins with deep learning. In *CIDR 2019, 9th Biennial Conference on Innovative Data Systems Research, Asilomar, CA, USA, January 13-16, 2019, Online Proceedings*. www.cidrdb.org, 2019.

[101] D. Knuth, Addison-Wesley, and P. Education. *The Art of Computer Programming*. Number v. 3 in Addison-Wesley series in computer science and information processing. Addison-Wesley, 1997.

[102] A. Kraskov, H. Stögbauer, and P. Grassberger. Estimating mutual information. *Physical review E*, 69(6):066138, 2004.

[103] 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*, pages 19–34, 2016.

[104] K. C. Kurnit, G. N. Kim, B. M. Fellman, D. L. Urbauer, G. B. Mills, W. Zhang, and R. R. Broaddus. Ctnnb1 (beta-catenin) mutation identifies low grade, early stage endometrial cancer patients at increased risk of recurrence. *Modern Pathology*, 30(7):1032–1041, 2017.

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

[106] O. Lehmberg, D. Ritze, P. Ristoski, R. Meusel, H. Paulheim, and C. Bizer. The mannheim search join engine. *Journal of Web Semantics*, 35:159 – 166, 2015.

[107] V. Leis, B. Radke, A. Gubichev, A. Kemper, and T. Neumann. Cardinality estimation done right: Index-based join sampling. In *CIDR 2017, 8th Biennial Conference on Innovative Data Systems Research, Chaminade, CA, USA, January 8-11, 2017, Online Proceedings*. www.cidrdb.org, 2017.

[108] J. Li, K. Cheng, S. Wang, F. Morstatter, R. P. Trevino, J. Tang, and H. Liu. Feature selection: A data perspective. *ACM Comput. Surv.*, 50(6), dec 2017.

[109] R. J. Lipton, J. F. Naughton, and D. A. Schneider. Practical selectivity estimation through adaptive sampling. *SIGMOD Rec.*, 19(2):1–11, May 1990.

[110] 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)*, pages 3360–3372. IEEE, 2022.

[111] DisjunctionMaxQuery (Lucene 8.8.2 API). `https://lucene.apache.org/core/8_8_2/core/org/apache/lucene/search/DisjunctionMaxQuery.html`.

[112] F. Mahdisoltani, J. Biega, and F. M. Suchanek. Yago3: A knowledge base from multilingual wikipedias. In *CIDR*, 2013.

[113] 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*, pages 355–363, 2017.

[114] P. Mandros, M. Boley, and J. Vreeken. Discovering reliable correlations in categorical data. In *2019 IEEE International Conference on Data Mining (ICDM)*, pages 1252–1257. IEEE, 2019.

[115] 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*, pages 1404–1414, 2020.

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

[117] T. Micceri. The unicorn, the normal curve, and other improbable creatures. *Psychological bulletin*, 105(1):156, 1989.

[118] A. Myers, W. T. Barry, M. S. Hirsch, U. Matulonis, and L. Lee. β-catenin mutations in recurrent figo ia grade i endometrioid endometrial cancers. *Gynecologic oncology*, 134(2):426–427, 2014.

[119] F. Nargesian, E. Zhu, K. Q. Pu, and R. J. Miller. Table union search on open data. *Proceedings of the VLDB Endowment*, 11(7):813–825, 2018.

[120] A. D. Nobari and D. Rafiei. Efficiently transforming tables for joinability. In *2022 IEEE 38th International Conference on Data Engineering (ICDE)*, pages 1649–1661. IEEE, 2022.

[121] NYC OpenData. `https://opendata.cityofnewyork.us`.

[122] City of Chicago Data Portal. `https://data.cityofchicago.org`.

[123] United States Government Open Data. `https://www.data.gov`.

[124] S. Padmanabhan, B. Bhattacharjee, T. Malkemus, L. Cranston, and M. Huras. Multi-dimensional clustering: A new data layout scheme in DB2. In *SIGMOD*, pages 637–641, 2003.

[125] L. Paninski. Estimation of entropy and mutual information. *Neural computation*, 15(6):1191–1253, 2003.

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

[127] 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*, pages 1254–1264, 2020.

[128] https://www.prio.org/Data/PRIO-GRID/, 2021.

[129] C. R. Rao. *Linear Statistical Inference and Its Applications.* Wiley, New York, 1973.

[130] Reports and Data. Data Catalog Market | Size & Growth Report, 2020-2027. https://www.reportsanddata.com/report-detail/data-catalog-market, 2020. [Online; accessed 28-March-2021].

[131] L. M. A. Rocha, A. Bessa, F. Chirigati, E. OFriel, M. M. Moro, and J. Freire. Understanding spatio-temporal urban processes. In *International Conference on Big Data (Big Data)*, pages 563–572, 2019.

[132] J. L. Rodgers and W. A. Nicewander. Thirteen ways to look at the correlation coefficient. *The American Statistician*, 42(1):59–66, 1988.

[133] B. C. Ross. Mutual information between discrete and continuous data sets. *PloS one*, 9(2):e87357, 2014.

[134] M. S. Roulston. Estimating the errors on measured entropy and mutual information. *Physica D: Nonlinear Phenomena*, 125(3-4):285–294, 1999.

[135] F. Rusu and A. Dobra. Sketches for size of join estimation. *ACM Trans. Database Syst.*, 33(3), Sept. 2008.

[136] 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*, pages 1531–1544, 2021.

[137] 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)*, pages 2928–2941, 2022.

[138] 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*, pages 1–7, 2019.

[139] A. Santos, F. Korn, and J. Freire. Efficiently estimating mutual information between attributes across tables. In *2024 IEEE 40th International Conference on Data Engineering (ICDE)*, 2024.

[140] R. L. Santos, C. Macdonald, I. Ounis, et al. Search result diversification. *Foundations and Trends® in Information Retrieval*, 9(1):1–90, 2015.

[141] 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.

[142] V. Shah, A. Kumar, and X. Zhu. Are key-foreign key joins safe to avoid when learning high-capacity classifiers? *Proceedings of the VLDB Endowment*, 11(3), 2017.

[143] 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*, SIGMOD '21, page 1584–1596, New York, NY, USA, 2021. Association for Computing Machinery.

[144] G. L. Shevlyakov and H. Oja. *Robust correlation: Theory and applications*, volume 3. John Wiley & Sons, 2016.

[145] G. Shieh. Estimation of the simple correlation coefficient. *Behavior Research Methods*, 42(4):906–917, 2010.

[146] M. Siedlaczek, A. Mallia, and T. Suel. Using conjunctions for faster disjunctive top-k queries. In *Proceedings of the Fifteenth ACM International Conference on Web Search and Data Mining*, pages 917–927, 2022.

[147] The Socrata Open Data API. `https://dev.socrata.com`.

[148] V. Solo. On causality and mutual information. In *2008 47th IEEE Conference on Decision and Control*, pages 4939–4944, 2008.

[149] G. J. Székely, M. L. Rizzo, and N. K. Bakirov. Measuring and testing dependence by correlation of distances. *Ann. Statist.*, 35(6):2769–2794, 12 2007.

[150] The Tablesaw Library. `https://github.com/jtablesaw/tablesaw`.

[151] D. Ting. Towards optimal cardinality estimation of unions and intersections with sketches. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 1195–1204, 2016.

[152] I. Tolstikhin. Concentration inequalities for samples without replacement. *Theory of Probability & Its Applications*, 61(3):462–481, 2017.

[153] N. Tonellotto, C. Macdonald, and I. Ounis. Efficient query processing for scalable web search. *Foundations and Trends in Information Retrieval*, 12(4-5):319–500, 2018.

[154] H. Turtle and J. Flood. Query evaluation: strategies and optimizations. *Information Processing & Management*, 31(6):831–850, 1995.

[155] P. Venetis, Y. Sismanis, and B. Reinwald. Crsi: a compact randomized similarity index for set-valued features. In *Proceedings of the 15th International Conference on Extending Database Technology*, pages 384–395, 2012.

[156] D. Vengerov, A. C. Menck, M. Zait, and S. P. Chakkappen. Join size estimation subject to filter conditions. *Proc. VLDB Endow.*, 8(12):1530–1541, Aug. 2015.

[157] J. R. Vergara and P. A. Estévez. A review of feature selection methods based on mutual information. *Neural computing and applications*, 24(1):175–186, 2014.

[158] J. S. Vitter. Random sampling with a reservoir. *ACM Transactions on Mathematical Software (TOMS)*, 11(1):37–57, 1985.

[159] 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*, pages 658–667, 2019.

[160] Wikipedia contributors. Multinomial distribution — Wikipedia, the free encyclopedia, 2023. [Online; accessed 01-August-2023].

[161] R. R. Wilcox. Confidence intervals for the slope of a regression line when the error term has nonconstant variance. *Computational Statistics & Data Analysis*, 22(1):89–98, 1996.

[162] C. C. Williams. Democratizing Data at Airbnb. `https://medium.com/airbnb-engineering/democratizing-data-at-airbnb-852d76c51770`, 2017. [Online; accessed 22-June-2020].

[163] World Bank Open Data. `https://data.worldbank.org`.

[164] World Bank Group Finances. `https://finances.worldbank.org`.

[165] C. Xiao, W. Wang, X. Lin, J. X. Yu, and G. Wang. Efficient similarity joins for near-duplicate detection. *ACM Transactions on Database Systems (TODS)*, 36(3):1–41, 2011.

[166] 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)*, pages 458–469, April 2019.

[167] Z. Yang, E. Liang, A. Kamsetty, C. Wu, Y. Duan, X. Chen, P. Abbeel, J. M. Hellerstein, S. Krishnan, and I. Stoica. Deep unsupervised cardinality estimation. *Proc. VLDB Endow.*, 13(3):279–292, Nov. 2019.

[168] 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*, pages 171–174, 2023.

[169] 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.

[170] K.-H. Yuan and P. M. Bentler. Inferences on correlation coefficients in some classes of nonnormal distributions. *Journal of Multivariate Analysis*, 72(2):230 – 248, 2000.

[171] K.-H. Yuan, P. M. Bentler, and W. Zhang. The effect of skewness and kurtosis on mean and covariance structure analysis: The univariate case and its multivariate implication. *Sociological Methods & Research*, 34(2):240–258, 2005.

[172] `https://zenodo.org/`, 2021.

[173] S. Zhang and K. Balog. Ad hoc table retrieval using semantic similarity. In *Proceedings of the 2018 World Wide Web Conference*, WWW '18, pages 1553–1562, Republic and Canton of Geneva, Switzerland, 2018. International World Wide Web Conferences Steering Committee.

[174] S. Zhang and K. Balog. Web table extraction, retrieval, and augmentation: A survey. *ACM Transactions on Intelligent Systems and Technology (TIST)*, 11(2):1–35, 2020.

[175] S. Zhang and K. Balog. Semantic table retrieval using keyword and table queries. *ACM Transactions on the Web (TWEB)*, 15(3):1–33, 2021.

[176] 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*, pages 1951–1966, 2020.

[177] 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*, SIGMOD '19, pages 847–864, New York, NY, USA, 2019. ACM.

[178] E. Zhu, F. Nargesian, K. Q. Pu, and R. J. Miller. Lsh ensemble: Internet-scale domain search. *Proc. VLDB Endow.*, 9(12):1185–1196, Aug. 2016.