# A Sketch-based Index for Correlated Dataset Search

Aécio Santos, Aline Bessa, Christopher Musco, Juliana Freire
*New York University*
{aecio.santos, aline.bessa, cmusco, juliana.freire}@nyu.edu

2022 IEEE 38th International Conference on Data Engineering (ICDE) | 978-1-6654-0883-7/22/$31.00 ©2022 IEEE | DOI: 10.1109/ICDE53745.2022.00264

***Abstract*—Dataset search is emerging as a critical capability in both research and industry: it has spurred many novel applications, ranging from the enrichment of analyses of real-world phenomena to the improvement of machine learning models. Recent research in this field has explored a new class of data-driven queries: queries consist of datasets and retrieve, from a large collection, related datasets. In this paper, we study a specific type of data-driven query that supports relational data augmentation through numerical data relationships: given an input query table, find the top-$k$ tables that are both joinable with it and contain columns that are correlated with a column in the query. We propose a novel hashing scheme that allows the construction of a sketch-based index to support efficient correlated table search. We show that our proposed approach is effective and efficient, and achieves better trade-offs that significantly improve both the ranking accuracy and recall compared to the state-of-the-art solutions.**

***Index Terms*—dataset search, table search, sketching**

## I. Introduction

There is an ever-increasing number of datasets available on the Web and within enterprises, which typically come in different structured formats that range from tables embedded on web pages to CSV and JSON files. Dataset search engines, repositories, and data catalogs [1]–[7] allow users to search large collections of datasets through simple keyword-based queries and faceted search over dataset metadata. These, however, have limited expressiveness, making it difficult (and sometimes impossible) to convey specific information needs. In this paper we study a specific type of query that enables discovery of datasets that are correlated to an input query table. To motivate the importance of correlations and correlated dataset search, we start with an example that shows the relationship between correlations and linear regression.

**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 [8]–[11] to improving machine learning models through feature selection [12], [13]. To illustrate this, we use the example of *simple linear regression* [14], one most fundamental and widely used methods for data analysis.

Linear regression is widely used to estimate the parameters in a linear equation that predict values of a variable $Y$ based on another variable $X$. Formally, we can say that $Y = b_0 + b_1 X$, 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, 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 $Y$ [15].

In other words, the stronger the correlation is, the higher is the predictive power of a variable. This illustrates 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.

**Scalability Challenges in Dataset Discovery.** Methods have been proposed to support table-driven queries. Given a table $\mathcal{T}_Q$ and a collection of tabular datasets $\mathcal{D}$, these queries can, for example, retrieve tables in $\mathcal{D}$ that are semantically similar to $\mathcal{T}_Q$ [16], or that can be merged with $\mathcal{T}_Q$ through relational join [17]–[19] or union [20] operations. These queries have been used in data augmentation to improve machine learning model performance [21], [22] and to enable novel applications in domains such as political violence modeling [8].

Evaluating these queries over large, heterogeneous data collections is challenging. For example, joinability queries can be answered using inverted indexes and query processing techniques [23], but they are orders of magnitude more expensive than typical web search queries: joinability queries require the computation of the overlap between the values of columns in the input table and the values of columns in the data collection. As a point of comparison, while web search engine queries are typically short and have an average of only 2.2 terms per query [24], modern data lakes store tables with thousands of unique values per column [23]. Recent studies showed that state-of-the-art query evaluation algorithms are efficient in the web search setting, but they also reported that mean and tail latencies quickly increase as the number of terms in the query grows [25]. For joinability queries, query evaluation becomes extremely expensive due to the large number of inverted index look-ups and large posting lists. This increased cost leads to query times that take up to multiple seconds [23] for retrieving top-20 joinable tables, while typical document retrieval queries take only a few milliseconds [25].

**Scaling Correlated Dataset Search.** One important class of joinability queries are *join-correlation queries* [19] which, given an input query table $\mathcal{T}_Q$ and a tabular dataset collection $\mathcal{D}$, retrieves tables in $\mathcal{D}$ that are both joinable with $\mathcal{T}_Q$ and

2375-026X/22/$31.00 ©2022 IEEE
DOI 10.1109/ICDE53745.2022.00264

contain columns correlated with one or more columns in $\mathcal{T}_Q$.

A typical solution to overcome scalability challenges in joinability queries is to provide approximate answers [17]–[19], [26] at cost of precision and recall. For correlated dataset search, there is an additional challenge: besides identifying possible joins, it is also necessary to compute correlation. The state-of-the-art solution for join-correlation queries [19] proposes a two-stage approach that relies on randomized sketching algorithms: it first retrieves joinable candidate tables based on join key overlap, and then re-ranks candidates using approximate correlation estimates computed efficiently with data sketches. While the 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.

**Our Contributions.** In this paper, 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. 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 [27]. 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.

In summary, our main contributions are:

- We define the new class of weighted join-correlation queries, of which join-correlation queries are a special case (Section III);
- We propose a novel hashing scheme and new sketch-based index – the *QCR index* – to efficiently support correlated table search over large collections (Section IV);
- We perform a detailed experimental evaluation (Section V) 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 existing approaches. 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.

## II. Related Work

This paper studies a problem that lies in the intersection of multiples research areas, including dataset discovery [28], table search [29], data sketching [30], and efficient query processing for web search [31].

**Dataset Discovery.** One of the lines of works that are closest to ours is the research on dataset discovery. Recent works have proposed dataset discovery methods that, given a query dataset $D_Q$ as input, find datasets that can be merged with $D_Q$ through relational operations such as *join* [17], [18], [23], [26] and *union* [20]. 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 [23] provide an exact solution to this problem, others such as LSH Ensemble [18], GB-KMV [26] and Lazo [17] propose approximate approaches.

In previous work [19], we formalized the problem of *join-correlation estimation*: how to efficiently estimate the *after-join* correlation between numerical columns from a pair of distinct tables without performing a join between them. We proposed a sketch data structure that approximates join-correlations efficiently. In order to search tables in a dataset collection using sketches, a two-step approach is required: tables retrieved using an approach for set overlap search (such as the ones described above [17], [18], [23]) need to be re-ordered according to correlations estimates computed using the sketches. In this paper, we introduce a new hashing scheme that allows searching for joinable and correlated tables in a single step. In addition, our indexing method is complementary to approach in [19] as it can replace the set overlap search to improve ranking quality (as shown in Section V).

At a higher level, there are works that propose end-to-end dataset search systems. For instance, the Data Civilizer [32] and Aurum [33] use linkage graphs to help identify relevant data for a given user task; JUNEAU [34] describes different data search tasks and table relatedness measures (e.g., column and row overlap, provenance, textual similarity), and proposes a method to combine these measures for dataset search in the Jupyter Notebook platform. Auctus [22] provides dataset search over tables collected from multiple open-data portals by automatically profiling and indexing tables using methods for joinable table search [17].

Some works focus on applying dataset discovery to solve machine learning problems. ARDA [21], for example, is a system that focuses on relational data augmentation, i.e., it automatically joins tables and selects the best features from tables discovered by dataset search systems to augment an initial query dataset. Another related system is Visus [35], which integrates dataset search with an interactive machine learning model-building workflow guided by humans. Our work is complementary to these: our methods for *efficient* correlated table search could be used to improve these systems.

Specifically, our method can be used by the dataset search engines that powers these systems [21], [22], [32]–[34], [36] to retrieve correlated tables that are more likely to improve machine-learning models [12], [13], [37].

Finally, our method is also loosely related to the work of Kumar et. al. [38], [39], where they propose decision rules to predict when the features obtained through a join can improve the error of a classification model. While their approach was developed for classification over categorical attributes, our methods are designed for numerical attributes. Besides supporting discovery of tables with correlated numeric attributes, our techniques could potentially be used to implement correlation-based feature selection methods [12], [13], [37] that do not require performing full table joins. We focus on the fundamental problem of efficient retrieval of correlated columns which has many applications beyond improving machine learning models (see e.g., [8]–[10]). Exploring these applications is out of scope for this work and is a direction we intend to pursue in future research.

**Web Tables and Ad-hoc Table Search.** Another related line of work focuses on discovering and performing automatic extension of web tables that contain entities in textual form [29], [40]. Zhang et al. [16], [41] 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 [42]–[45]. We refer the reader to [31] 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 [43] 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 [31], [46]. As we discuss in Section IV-A, this property prevents the use of these algorithms for correlated table retrieval. In this paper, we show that by decomposing the retrieval into two smaller queries, we can leverage these algorithmic optimizations. Our approach also uses a technique that has been described as cascading [31]. 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.

## III. Preliminaries and Problem Definition

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 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 with the repeated values of $k$, e.g., $c_{\mathtt{a}} = \mathtt{AVG}(\{5.5, 4.5\}) = 5.0$. 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 1 shows a complete example of query and candidate tables, along with their corresponding join table after value aggregations.

$\mathcal{T}_Q$

| $K_Q$ | $Q$ |
|---|---|
| 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 | 3.9 |
| 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 |

Fig. 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.

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.

**Definition 1 (Top-$k$ Join-Correlation Query [19]).** *Given $\mathcal{T}_Q$ and an integer $k > 0$, find the top-$k$ joinable tables $\mathcal{T}_C \in \mathcal{D}$ with the highest (after-join) correlations between numerical columns $Q_{Q\bowtie C}$ and $C_{Q\bowtie C}$.*

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 [21]: 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 2 (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:

$$W(j,r) = \begin{cases} w & \text{if } j > 0 \text{ (i.e., the tables are joinable)} \\ 0 & \text{otherwise} \end{cases} \quad (1)$$

where $r$ is the absolute value of a correlation coefficient such as Pearson's correlation,[1] $j$ is a joinability score such 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 $\alpha_i$ 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 $\alpha_j$ is the weight for joinability $j$ and $\alpha_r$ is the weight for correlation $r$.

Note that we can express the join-correlation query from Definition 1 using Definition 2 and Equation 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 of the scope this paper. Here, we focus on the version of correlation queries proposed in [19] (Definition 1), 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 maximizes an equal-weight weighting function (details in Section IV). As we show in Section V, this retrieval method enables not only 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 1 discussed in [19].

[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 case. As we show next, this simplifies the problem.

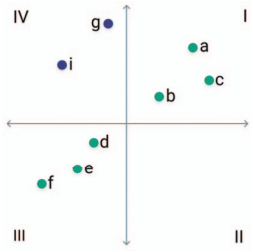


Fig. 2. An example of the four quadrants, where green points (●) contribute to positive correlation and blue points (●) contribute to negative correlation.

## IV. Our Approach

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 [31]. 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 [31], [42], [43] to retrieve correlated tables.

### A. 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, previous approaches [19] proposed to retrieve joinable tables by matching tables using only the 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 keys. Our hashing scheme is based on the correlation estimator known as Quadrant Count Ratio (QCR) [27]. 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 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 ) [27]. 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:

$$|r_{QCR}| = \frac{|N^+ - N^-|}{N^\cap}, \tag{2}$$

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 2 violates the non-negative monotonicity property required by the top-$k$ query processing algorithms [31], [46]: 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:

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

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 highest correlation 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 [19]. For each table $\mathcal{T}_C = \langle K_C, C\rangle$, we first build a correlation sketch $L_{\langle K_C, C\rangle}$. The sketch is then used to derive a set of terms $T_C$ to represent the table in the inverted index. To make the paper self-contained, we will first briefly describe how to build these sketches (Section IV-B), and then we will describe how to compute the QCR index terms from sketches (Section IV-C).

### B. 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 sketch $L_{\langle K_C, C\rangle}$. 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, $L_{\langle K_C, C\rangle}$ includes the $n$ tuples $\langle h(k), c_k\rangle$ with the minimum values of $h_u(k)$, i.e., $L_{\langle K_C, C\rangle} = \{\langle h(k), c_k\rangle : k \in min(k, h_u(k))\}$, 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 III.

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 $L_{\langle K_Q, Q\rangle}$ and $L_{\langle K_C, C\rangle}$ can be used to compute correlations by creating joined sketch $L_{Q \bowtie C}$ and applying any correlation estimator [19].

### C. 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 $\{c_k\} \in L_{\langle K_C, C\rangle}$ and of the hashed key $h(k)$. Specifically, we compute $t_k$ as:

$$t_k = \begin{cases} h(h(k) \oplus +1) & \text{if } c_k - \mu_c > 0 \\ h(h(k) \oplus -1) & \text{if } c_k - \mu_c < 0 \end{cases} \tag{3}$$

where $\mu_c$ 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 $\mu_c$ is a reference point used to partition the quadrants. As illustrated in Figure 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 [19].

In the example of Figure 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 $\flat$, and both $c_b$ and $q_b$ are greater than $\mu_c$ and $\mu_q$, respectively. However, it would not assign the same

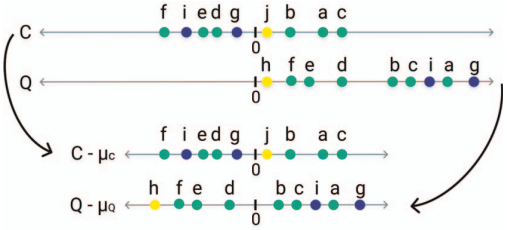


Fig. 3. An example of mean centering for two tables $\mathcal{T}_C$ and $\mathcal{T}_Q$. Letters represent join keys $k$ and the positions along the lines represent values $c_k$. Yellow circles (●) mean that the keys do not join. Green circles (●) denote that the keys join and are located in a "positive 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 2.

hash value to $c_b$ and $q_b$ if, for example, $c_b$ were less than $\mu_q$. 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 $\mu_c$ 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 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 correlation equally. Moreover, by using correlation estimates computed using the sketches, we can correct possible errors in a reranking phase. In practice, our experimental evaluation (Section V) 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 $L_{\langle K_C,C\rangle}$ and then use Equation 3 to compute terms $T_C$, which are ultimately used to represent the table $\mathcal{T}_C$ in the QCR inverted index.

### D. 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 IV-C to query the inverted index. We start by constructing a sketch $L_{\langle K_Q,Q\rangle}$ 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 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 $\{q_k\} \in L_{\langle K_Q,Q\rangle}$ — i.e., we apply a transformation to $\{q_k\}$ that multiplies all of its elements by $-1$. We show in our theoretical analysis (Section IV-F) 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 $s = max(s^+, s^-)$.

### E. Implementation Details

We implemented our algorithms in Java, and we used the Apache Lucene library [47] 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` [48], 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 [43].

### F. 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 1.** *The following bounds hold for a score $s^+$ computed using the QCR hashing scheme and a sketch of size $n$:*

$$\frac{N^+}{N^\cap}\rho \leq \mathbb{E}\left[\frac{s^+}{n}\right] \leq 2\frac{N^+}{N^\cap}\rho \tag{4}$$

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

To compare our approach with the approach from [19], which we refer to as CSK, we also extend our analysis to the scores produced by their method (as both scores are counts of hash collisions). Before presenting our analysis, we describe CSK. **The CSK indexing scheme.** While our QCR approach uses the set of hashes computed using Equation 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 $T_C = \{h(k) \in L_{\langle K_C,C\rangle}\}$. At query time, it constructs a sketch $L_{\langle K_Q,Q\rangle}$ for the query table, which is used to create the query term set $T_Q = \{h(k) \in L_{\langle K_Q,Q\rangle}\}$. The queries find the top-$k$ tables with 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 $L_{\langle K_C,C\rangle}$ and $L_{\langle K_C,C\rangle}$, which then places correlated tables at the top of the ranked list. This indexing approach based on the hashed keys of sketches (CSK) was presented informally in [19]. In what follows, we present an analysis of $s^\cap$, which is part of the contributions of this paper.

**Proof of Lemma 1.** Let $L_A$ and $L_B$ be the sketches computed for the tables $\mathcal{T}_A$ and $\mathcal{T}_B$, respectively. Moreover, let $U = \{k_i : h(k_i) \in L_A \cup L_B\} = \{k_1, k_2, ..., k_n, ..., k_{|L_A \cup L_B|}\}$ 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 $L_A$ and $L_B$, respectively. Finally, $T = \{t_1, ..., t_{|L_A \cup L_B|}\}$ 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 $s = \sum_{i=1}^{|L_A \cup L_B|} u_i$ 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 $k_i \in L_A$ and $k_i \in L_B$.

The probabilities the keys being included in the sketches (i.e., $\mathbb{P}\{k_i \in L_A \text{ and } k_i \in L_B\}$) depend on the sketch size $n$ and are not uniform for the set of all $i$'s. In fact, as proven in [49], $\mathbb{P}\{k_i \in L_A \text{ and } k_i \in L_B\} = 1$ for all $i \leq n$ when sketches have size $n$, given that $\mathbb{P}\{k_i \in L_A\} = 1$ and $\mathbb{P}\{k_i \in L_B\} = 1$. Thus, we divide the analysis in two cases: $\{u_1, ..., u_n\}$ and $\{u_{n+1}, ..., u_{|L_A \cup L_B|}\}$. First, note that $s$ can be decomposed into the sum of the parts: $s = s(1, n) + s(n + 1, |L_A \cup L_B|)$. Then, due to linearity of expectation, we have that:

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

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:

$$\begin{aligned}
\mathbb{E}[s(1, n)] &= \sum_{i=1}^{n} \mathbb{P}\{I \mid C\} \cdot \mathbb{P}\{C\} \\
&= \sum_{i=1}^{n} 1 \cdot \mathbb{P}\{C\} \\
&= n\mathbb{P}\{C\} \qquad (5)
\end{aligned}$$

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

$$\begin{aligned}
\mathbb{E}[s(n + 1, |L_A \cup L_B|)] &= \sum_{i=n+1}^{|L_A \cup L_B|} \mathbb{P}\{I \mid C\} \cdot \mathbb{P}\{C\} \\
&\leq \sum_{i=n+1}^{|L_A \cup L_B|} 1 \cdot \mathbb{P}\{C\} \\
&\leq n\mathbb{P}\{C\} \qquad (6)
\end{aligned}$$

Combining Equations 5 and 6, we get that:

$$\begin{aligned}
n\mathbb{P}\{C\} \leq\ & \mathbb{E}[s] \leq n\mathbb{P}\{C\} + n\mathbb{P}\{C\} \\
n\mathbb{P}\{C\} \leq\ & \mathbb{E}[s] \leq 2n\mathbb{P}\{C\} \qquad (7) \\
\mathbb{P}\{C\} \leq\ & \mathbb{E}\left[\frac{s}{n}\right] \leq 2\mathbb{P}\{C\} \qquad (8)
\end{aligned}$$

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 8, we obtain the following bounds:

$$\frac{N^\cap}{N^\cup} \leq \mathbb{E}\left[\frac{s^\cap}{n}\right] \leq 2\frac{N^\cap}{N^\cup} \quad \text{for CSK, and} \qquad (9)$$

$$\frac{N^+}{N^\cup} \leq \mathbb{E}\left[\frac{s^+}{n}\right] \leq 2\frac{N^+}{N^\cup} \quad \text{for QCR.} \qquad (10)$$

To get the results from Lemma 1, note that the following equality holds: $\frac{N^+}{N^\cup} = \frac{N^+}{N^\cap}\frac{N^\cap}{N^\cup}$. □

### G. 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 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 [19] and they can be used in a cascading fashion: one can first retrieve tables using the QCR index, as described in Section IV-C, and then pass the retrieved candidates to another layer that re-ranks the candidate tables using estimates produced by correlation sketches [19]. 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 2.

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 intro account the risk of estimation error could be used to avoid placing false-positives in the top of the ranked list (as done in [19]). Note, however, that these improvement are only applicable to the re-ranking phase (i.e., they are orthogonal to the approach and experimental results we present in this paper), 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 V), 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 [19]. 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.

## V. Experimental Evaluation

### A. 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 [50]. We used a snapshot that includes 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 list. To measure different aspects of retrieval quality, we use the following evaluation metrics:

*(1) Normalized Cumulative Gain (nDCG)* [51]: measures the ability of placing 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 to 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 of prioritizing 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 2), 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 [19] under multiple parameter settings. As discussed in Section IV-F, 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 ($\alpha_r = 1$, $\alpha_j = 0$).

TABLE I
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 |

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.

### B. 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 [19]). 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 I and II respectively.

*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 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 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 V-C), the accuracy of the correlation

TABLE II
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 |
| 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 |

TABLE III
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 |

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 [19]. 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%).

*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

TABLE IV
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 V
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 |

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.

*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 I and II. These results, along with results from Sections V-B1 and V-B2, confirm that the QCR-based retrieval strategies are able to achieve a better overall ranking quality using smaller sketch sizes.

### C. 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 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 III (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 to 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 IV and V (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.

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 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).

### D. Performance Evaluation

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

TABLE VI
RUNNING TIME FOR DIFFERENT PARAMETER SETTINGS ALONG WITH THEIR RANKING SCORES ON THE NYC COLLECTION.

| | index | ranking | top-$k$ | $n$ | nDCG/Rec @10, $r > 0.75$ | nDCG/JC @10 | Time $Avg.$ |
|---|---|---|---|---|---|---|---|
| 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 |

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 VI, 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 4. In this plot, the best methods are the ones located closer to top-left corner, i.e., the region of better metric

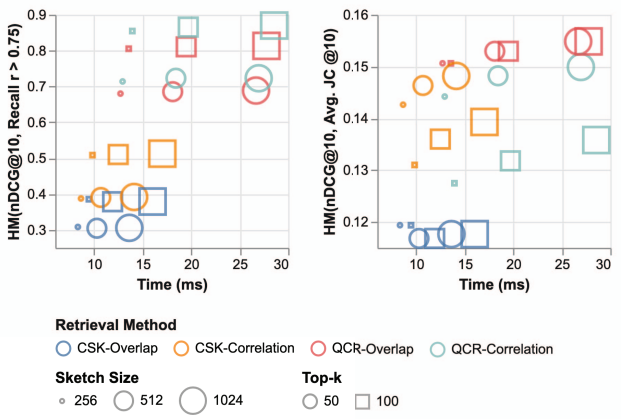


Fig. 4. Runtime versus retrieval quality scores for different parameter settings.

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

## VI. CONCLUSION

We proposed a generalization of join-correlation queries that consider both joinabilty and correlation. This new approach led to the development of an indexing and retrieval method for correlated table search based on our novel QCR hashing scheme. We provide theoretical analysis of algorithms and we show that our approach improves the retrieval of correlated tables in terms of both ranking accuracy and recall through an extensive experimental evaluation.

There are multiple open problems in this line of research that are worth studying in future work. For instance, a natural question that may arise is how to discover correlations between other types of data (e.g., images, text). A natural approach is to encode such data as numerical variables. For example, text could be encoded using techniques such as TF-IDF, topic modeling (LDA) [52], or deep-learning-based approaches (e.g., BERT [53]). A challenge here is that this would significantly increase the dimensionality of the data (as each text column would be converted into potentially hundreds to thousands of columns) and thus impact the performance. We would also like to explore the application of our methods to downstream applications [21], [35], dataset search engines, and data catalogs [6], [22], [32]. In terms of performance, it would be interesting to evaluate other fast algorithms for top-$k$ retrieval [25], [45], and set overlap search [23].

## ACKNOWLEDGMENTS

This work was partially supported by the DARPA D3M program and NSF award ISS-2106888. Any opinions, findings, conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of NSF and DARPA.

## REFERENCES


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

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

[3] CKAN, "The open source data portal software," https://ckan.org/.

[4] "The Dataverse Project," https://dataverse.org.

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

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

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

[8] 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*, 2021, pp. 2384–2388.

[9] D. Y. Kenett, X. Huang, I. Vodenska, S. Havlin, and H. E. Stanley, "Partial correlation analysis: Applications for financial markets," *Quantitative Finance*, vol. 15, no. 4, pp. 569–578, 2015.

[10] 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)*, 2019, pp. 563–572.

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

[12] 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.

[13] M. A. Hall and L. A. Smith, "Feature selection for machine learning: comparing a correlation-based filter approach to the wrapper." in *FLAIRS conference*, vol. 1999, 1999, pp. 235–239.

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

[15] J. L. Rodgers and W. A. Nicewander, "Thirteen ways to look at the correlation coefficient," *The American Statistician*, vol. 42, no. 1, pp. 59–66, 1988. [Online]. Available: https://doi.org/10.1080/00031305.1988.10475524

[16] S. Zhang and K. Balog, "Semantic table retrieval using keyword and table queries," *ACM Transactions on the Web (TWEB)*, vol. 15, no. 3, pp. 1–33, 2021.

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

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

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

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

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

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

[23] E. Zhu, D. Deng, F. Nargesian, and R. J. Miller, "Josie: Overlap set similarity search for finding joinable tables in data lakes," in *Proceedings of the 2019 International Conference on Management of Data*, ser. SIGMOD '19. New York, NY, USA: ACM, 2019, pp. 847–864. [Online]. Available: http://doi.acm.org/10.1145/3299869.3300065

[24] S. M. Beitzel, E. C. Jensen, A. Chowdhury, D. Grossman, and O. Frieder, "Hourly analysis of a very large topically categorized web query log," in *Proceedings of the 27th annual international ACM SIGIR conference on Research and development in information retrieval*, 2004.

[25] O. Khattab, M. Hammoud, and T. Elsayed, "Finding the best of both worlds: Faster and more robust top-k document retrieval," 2020. [Online]. Available: https://doi.org/10.1145/3397271.3401076

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

[27] P. Holmes, "Correlation: From picture to formula," *Teaching Statistics*, vol. 23, no. 3, pp. 67–71, 2001.

[28] A. Chapman, E. Simperl, L. Koesten, G. Konstantinidis, L.-D. Ibáñez, E. Kacprzak, and P. Groth, "Dataset search: a survey," *The VLDB Journal*, vol. 29, no. 1, pp. 251–272, 2020.

[29] S. Zhang and K. Balog, "Web table extraction, retrieval, and augmentation: A survey," *ACM Transactions on Intelligent Systems and Technology (TIST)*, vol. 11, no. 2, pp. 1–35, 2020.

[30] G. Cormode, M. N. Garofalakis, P. J. Haas, and C. Jermaine, "Synopses for massive data: Samples, histograms, wavelets, sketches," *Foundations and Trends in Databases*, vol. 4, no. 1-3, pp. 1–294, 2012.

[31] N. Tonellotto, C. Macdonald, and I. Ounis, "Efficient query processing for scalable web search," *Foundations and Trends in Information Retrieval*, vol. 12, no. 4-5, pp. 319–500, 2018.

[32] 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. [Online]. Available: http://cidrdb.org/cidr2017/papers/p44-deng-cidr17.pdf

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

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

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

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

[37] K. E. Brown and D. A. Talbert, "Heuristically reducing the cost of correlation-based feature selection," in *Proceedings of the 2019 ACM Southeast Conference*, ser. ACM SE '19. New York, NY, USA: Association for Computing Machinery, 2019, p. 24–30. [Online]. Available: https://doi.org/10.1145/3299815.3314428

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

[39] V. Shah, A. Kumar, and X. Zhu, "Are key-foreign key joins safe to avoid when learning high-capacity classifiers?" *arXiv preprint arXiv:1704.00485*, 2017.

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

[41] S. Zhang and K. Balog, "Ad hoc table retrieval using semantic similarity," in *Proceedings of the 2018 World Wide Web Conference*, ser. WWW '18. Republic and Canton of Geneva, Switzerland: International World Wide Web Conferences Steering Committee, 2018, pp. 1553–1562. [Online]. Available: https://doi.org/10.1145/3178876.3186067

[42] 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*, 2003, pp. 426–434.

[43] 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*, 2011, pp. 993–1002.

[44] H. Turtle and J. Flood, "Query evaluation: strategies and optimizations," *Information Processing & Management*, vol. 31, no. 6, pp. 831–850, 1995.
[45] 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*, 2022, pp. 917–927.
[46] R. Fagin, A. Lotem, and M. Naor, "Optimal aggregation algorithms for middleware," *Journal of computer and system sciences*, vol. 66, no. 4, pp. 614–656, 2003.
[47] "Apache lucene," https://lucene.apache.org/index.html.
[48] "DisjunctionMaxQuery (Lucene 8.8.2 API)," https://lucene.apache.org/core/8_8_2/core/org/apache/lucene/search/DisjunctionMaxQuery.html.
[49] K. Beyer, P. J. Haas, B. Reinwald, Y. Sismanis, and R. Gemulla, "On synopses for distinct-value estimation under multiset operations," in *Proceedings of the 2007 ACM SIGMOD International Conference on Management of Data*, ser. SIGMOD '07. New York, NY, USA: ACM, 2007, pp. 199–210. [Online]. Available: http://doi.acm.org/10.1145/1247480.1247504
[50] "NYC OpenData," https://opendata.cityofnewyork.us.
[51] K. Järvelin and J. Kekäläinen, "Cumulated gain-based evaluation of ir techniques," *ACM Transactions on Information Systems (TOIS)*, vol. 20, no. 4, pp. 422–446, 2002.
[52] D. M. Blei, A. Y. Ng, and M. I. Jordan, "Latent dirichlet allocation," *Journal of machine Learning research*, vol. 3, no. Jan, pp. 993–1022, 2003.
[53] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "Bert: Pre-training of deep bidirectional transformers for language understanding," *arXiv preprint arXiv:1810.04805*, 2018.