Efficient Attribute Recommendation with Probabilistic Guarantee

Chi Wang
Microsoft Research, Redmond WA
wang.chi@microsoft.com

Kaushik Chakrabarti
Microsoft Research, Redmond WA
kaushik@microsoft.com

ABSTRACT

We study how to efficiently solve a primitive data exploration problem: Given two ad-hoc predicates which define two subsets of a relational table, find the top-K attributes whose distributions in the two subsets deviate most from each other. The deviation is measured by $\ell 1$ or $\ell 2$ distance. The exact approach is to query the full table to calculate the deviation for each attribute and then sort them. It is too expensive for large tables. Researchers have proposed heuristic sampling solutions to avoid accessing the entire table for all attributes. However, these solutions have no theoretical guarantee of correctness and their speedup over the exact approach is limited. In this paper, we develop an adaptive querying solution with probabilistic guarantee of correctness and near-optimal sample complexity. We perform experiments in both synthetic and real-world datasets. Compared to the exact approach implemented with a commercial DBMS, previous sampling solutions achieve up to 2× speedup with erroneous answers. Our solution can produce 25× speedup with near-zero error in the answer.

KEYWORDS

Multi-dimensional data; Exploratory analysis; Sampling

ACM Reference Format:
Chi Wang and Kaushik Chakrabarti. 2018. Efficient Attribute Recommendation with Probabilistic Guarantee. In *Proceedings of ACM SIGKDD Conference (KDD'18).* ACM, New York, NY, USA, 10 pages. https://doi.org/10.1145/3219819.3219984

1. INTRODUCTION

Data analysts and data scientists explore and analyze relational datasets to answer business questions, spot insights and find patterns. To analyze a dataset with many attributes, one of the most important initial steps is to identify a few attributes worthy of exploration. This step requires significant manual effort from a skilled data scientist [23, 24]. It is of tremendous value to sharply reduce that effort via automated techniques [3, 13, 14, 20].

In this paper, we study an *attribute recommendation problem*: find the top-K attributes ranked by their distribution's deviation in two ad-hoc subsets of a relational table. We present an example scenario hereby.

KDD'18, August 19–23, 2018, London, United Kindom

ACM ISBN 978-1-4503-5552-0/18/08…$15.00
https://doi.org/10.1145/3219819.3219984

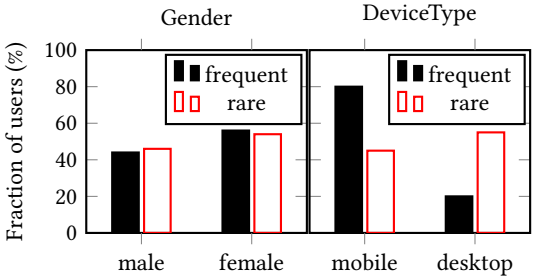

Fig. 1. The distribution of Gender and DeviceType among frequent and rare users of S1 in Example 1

Example 1. (*Deviation-based Attribute Recommendation*) Consider a simple database that stores customer information in a single table with hundreds of columns: (CustomerId, Gender, AgeGroup, DeviceType, S1MinsUsed, S2StorageUsed…) Each row corresponds to a customer. It stores the customer's demographic information (e.g., gender) as well as her product usage behavior (e.g., S1MinsUsed where S1 is a service/product of the company).

Suppose a marketer wants to conduct a marketing campaign to drive up the usage of S1. She wants to understand the difference between the frequent users of S1 (say, 'S1MinsUsed > 1000') and the rare users ('S1MinsUsed < 10'), so that she can target the campaign more specifically. To achieve this, she looks for attributes of which the distribution among the frequent users deviates from the distribution among the rare users. For example, Figure 1 shows the distribution of Gender and DeviceType in the two subsets. The distributions of Gender across the two subsets deviate less than DeviceType. Thus the latter is more interesting in this example. Manually checking each attribute's distribution and comparing them is a burden. A system to recommend top-K attributes is helpful for users to quickly identify a small number of interesting attributes. □

Several deviation-based attribute ranking functions (referred to as *utility functions*) have been validated in visual data exploration systems [10, 19, 22]. One example of such utility function is the Euclidean distance: Given an attribute A and two subsets defined by predicate P_1 and P_2, the Euclidean distance can be computed from the result of the following two SQL queries:

SELECT A, COUNT(∗) FROM T WHERE P_1 GROUP BY A
SELECT A, COUNT(∗) FROM T WHERE P_2 GROUP BY A

as $\|\mathbf{p}_1 - \mathbf{p}_2\|_2$, where $\mathbf{p}_1$ and $\mathbf{p}_2$ are normalized count vectors from the two group-by queries. Given the utility function, our focus is to find the top-K attributes *efficiently*.

Deficiency in the state-of-the-art: A straightforward approach is to execute the SQL queries for each attribute, and compute and

sort the utility of them. For large tables with many records, this approach (which we will refer to as *exact approach*) has high latency.

SeeDB [22] explores the idea of reducing data access by sampling. It leverages the Successive-Accept-Reject (SAR) algorithm designed for the best arm identification from a multi-armed bandit (MAB) [5]. In a MAB, an arm is defined to be a sampler from an unknown distribution over real values. One can 'pull an arm', i.e., acquire a sample, for unlimited times. The samples are called observed rewards, and the distribution is called reward distribution. Given multiple arms with unknown reward distributions, SAR can iteratively pull the arms to sample rewards, and eventually return top-K arms ranked by the expectation of reward distribution, with high probability. The attribute recommendation problem cannot be exactly reduced to a MAB problem, but SeeDB executes the SAR algorithm in the following heuristic way. It creates an arm for each attribute, and partitions the rows of a table randomly into a few equal sized parts. Whenever the SAR algorithm pulls an arm, it calculates the utility of the corresponding attribute in one unused partition as an 'observed reward' of the arm. When all partitions are used, it terminates the algorithm and returns the top-K attributes ranked by the mean of 'observed rewards.' This heuristic MAB modeling suffers two issues.

• No guarantee of the correctness, even probabilistically. The expectation of the utilities calculated from each partition is not equal to the utility calculated from the entire table. Therefore, the returned K attributes can deviate significantly from the actual top-K with unbounded chance, which makes the results untrustworthy.

• High sampling rate. The number of data partitions has to be small in order to limit the overhead of querying partitions. Applying SAR in this situation does not reduce the amount of data access significantly from the exact approach.

An alternative solution in [22] is to prune the arms using confidence intervals based on the same MAB modeling. It suffers the same issues. The high sampling rate issue is because the number of 'observed rewards' is too small to yield useful confidence interval. These deficiencies suggest that a deeper exploration of the sampling idea is required than the heuristic MAB modeling.

Our solution: We develop a rigorous and efficient adaptive querying solution. First, we propose a more accurate approach to estimate the utility. As opposed to averaging the utility calculated from each partition, we first aggregate the SQL query results from each partition, and then calculate the utility from the aggregated result. It is ensured that when all partitions are used for an attribute, the utility estimator is unbiased. The new estimator also allows new modeling of the sampling process, so that each row can be modeled as a sample unit rather than each partition. Based on that, we invent new analytic techniques for calculating confidence intervals. Our derived interval is 3-4 orders of magnitude shorter than the previous result when a partition has 10^6 to 10^8 rows. That makes the confidence intervals informative during the adaptive querying. From the confidence intervals, we can identify 'competing attributes' (some but not all of them are plausible top-K). We adaptively choose the most uncertain attribute among them to query until there is no competing attribute. Our algorithm is simple to implement but difficult to analyze. We prove both correctness and sample complexity of our algorithm with probabilistic guarantee. Furthermore, we prove the sample complexity is near optimal.

Example 2. (*Illustration of Adaptive Querying*)
Suppose the table in Example 1 has 100 attributes and 100M rows. The goal is to find top-5 attributes with largest Euclidean distance utility. Suppose the table is randomly partitioned into 10 equal sized parts, each containing 10M random rows. When two ad-hoc predicates are given as input, we first execute the 200 group-by queries (corresponding to 100 attributes) on partition 1. We compute the utility's confidence interval for attributes $A_1 - A_{100}$, as depicted in the left plot of Figure 3 ($A_6 - A_{100}$ have similar confidence intervals). Next, instead of querying all attributes on partition 2, our adaptive querying algorithm will iteratively select one attribute to query. In this particular case, A_5 and A_6 are competing attributes (the exact definition is in Section 3.1) and A_5 has longer confidence interval. So it selects A_5 as the first attribute to query using partition 2. The group-by results from partition 1 and 2 for A_5 are summed up group-wise, and the estimation of its utility is updated. If it finds no overlap between the confidence intervals of $A_1 - A_5$ and $A_6 - A_{100}$, it stops querying more attributes. That saves 99% queries to partition 2. Note that the result of the query is unknown when the attribute is selected, and the above situation happens only probabilistically. So the analysis of the algorithm is difficult. □

Our contributions can be summarized as follows:

• We present the first adaptive querying algorithm that is guaranteed to return correct top-K attributes with high probability. Its sample complexity is near optimal. (Section 3)

• We present novel analytic techniques to derive confidence interval for two common deviation-based utility functions. The results are the first of its kind. (Section 4)

• We conduct an extensive empirical study on real-life and synthetic datasets. Our approach demonstrates 25× speedup with almost no error. We also provide ablation study to prove that both the new adaptive querying algorithm and the new confidence interval calculation are critical for the superior performance. (Section 5)

2. PROBLEM STATEMENT

We first define the attribute recommendation problem (Section 2.1). We then describe the adaptive-querying system architecture and isolate the technical problem we solve in this paper (Section 2.2).

2.1. Attribute Recommendation

Exploration model. As is standard in OLAP and in visual analytic tools, we consider databases with a snowflake schema. The user conducts data exploration over either a single table or the result of join of tables involved in the schema. For simplicity, we explain our techniques in the context of a single table T. The user can invoke the recommendation functionality for two ad-hoc subsets of the rows in T. The subsets are defined by predicates containing any expression except nested queries. So the two subsets can overlap, and one of them can be the full table (with empty predicate). The recommendation system returns a small number of attributes, and users study one individual attribute at a time. These assumptions are similar to previous work and suffice for many data exploration tasks [22, 24]. Furthermore, the user can also apply transformation to attributes (e.g., binning for numeric attributes), and choose a subset of C attributes for consideration. We refer to the set as candidate attributes and denote as $\mathcal{A} = \{A_1, \ldots, A_C\}$.

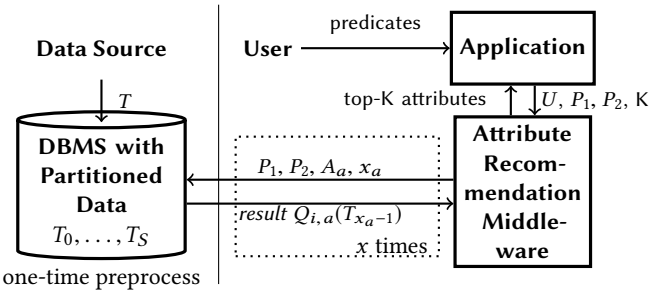

Fig. 2. Attribute recommendation based on querying randomly partitioned data. The partition is done offline

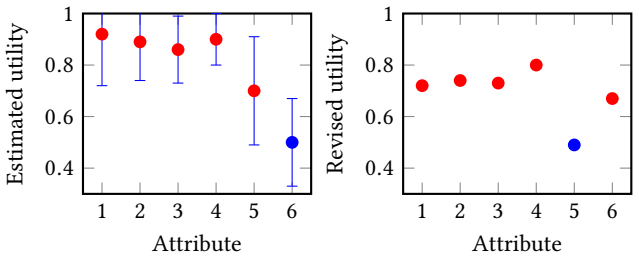

Fig. 3. Illustration of competing attributes. Red dots represent estimated top-K and alternative top-K in the two plots. Dots corresponding to competing attributes (A_5 and A_6) change color

Utility function for attribute ranking. Given two predicates P_1 and P_2, the goal is to find attributes whose distribution in the two subsets filtered by P_1 and P_2 have largest deviation. Normalized $\ell 1$ (a.k.a. total variation or earth mover distance) and $\ell 2$ (a.k.a. Euclidean distance) are two effective metrics of deviation [22]:

$$\ell 1 = \frac{1}{2}\|\mathbf{p}_1 - \mathbf{p}_2\|_1, \ell 2 = \frac{1}{\sqrt{2}}\|\mathbf{p}_1 - \mathbf{p}_2\|_2 \in [0, 1] \quad (1)$$

where $\mathbf{p}_i \triangleq \left(\frac{m_{ij}}{m_i}\right)_{j=1}^{c_a}$, m_{ij} denotes the number of tuples with j-th value in attribute A_a satisfying P_i, $m_i = \sum_{j=1}^{c_1} m_{ij}$ the total number of tuples satisfying P_i, and c_a the number of distinct values in A_a (or the number of bins if A_a is a binned numeric attribute).

The main cost of computing the utility for an attribute is in collecting its distribution under the two predicates, which can be done in two group-by queries. To unify both utility functions in the same form, we can define the utility of attribute A_a as $U_a = U(Q_{1,a}(T), Q_{2,a}(T))$, where:

• $Q_{i,a}(T), i = 1, 2$ is the result table from the query

SELECT A_a, COUNT(*) FROM T WHERE P_i GROUP BY A_a

• U is $\ell 1$ or $\ell 2$ as defined in Eq. (1).

Definition 1. (*Attribute Recommendation*)
Given the utility function U, P_1, P_2 and a positive integer K, return $\mathcal{I} \subset [C]$ s.t. $|\mathcal{I}| = K$ and $\forall a \in \mathcal{I}, b \in [C] \setminus \mathcal{I}, U_a \geq U_b$. □

2.2. System Architecture

The exact approach executes the queries $Q_{i,a}$ on T for each attribute and then ranks them by U. The major cost is spent in executing the group-by queries on the data of size $\Omega(CN)$, where N is the total number of rows in T and C the number of candidate attributes. We propose an adaptive querying solution to reduce the data access but return correct answers with high probability.

Figure 2 depicts the system architecture. The system is implemented as a middleware to run on top of any relational DBMS. The system has two components, preprocessing and online recommendation.

Preprocessing: At preprocessing time, the user points to the table T on which she wants to conduct data exploration. The table can reside in any platform (e.g., data warehouse, Hadoop). The system reads the rows in T and randomly partitions them into $S+1$ horizontal fragments $T_0, T_1, \ldots, T_S$. The first fragment is small (e.g., 1%) in order for making a quick initial utility estimate of all attributes. The remaining S fragments have equal expected size. We create a table in the DBMS for each fragment (for joins on snowflake schema, only the fact table needs to be materialized). More partitions imply more flexibility to prune attributes early, but incur more roundtrip overhead. To prevent performance regression in the worst case (i.e., all partitions need to be queried), S should be small enough (empirically, around 10) to keep the total overhead ignorable compared with querying the full table.

Online recommendation: The user specifies the predicates via a data analytic application's front end, like Tableau[1] or Power BI[4]. The application invokes the attribute recommendation algorithm with a utility function U, predicates P_1, P_2 and a positive integer K. The attribute recommendation algorithm iteratively executes queries until it finds the top-K attributes with high probability. The application can then present the results to the user. For example, a visualization tool might present a chart for each returned attribute explaining why that attribute is ranked high, like the bar chart shown in Figure 1.

Technical problem: Let x denote the number of requests sent to the DBMS by the attribute recommendation algorithm. Our goal is to return true top-K attributes with probability at least $1 - \delta$ (δ is a positive constant between 0 and 1), while minimizing x.

3. ALGORITHM AND ANALYSIS

Our solution consists of an adaptive querying algorithm and a technique for calculating confidence intervals. This section focuses on the adaptive querying algorithm, named TopKAttr. Section 3.1 presents the algorithm, and Section 3.2 provides formal guarantee of the algorithm's correctness and sample complexity.

3.1. Algorithm

We introduce several definitions for the algorithm.

• Utility estimator. We define the utility estimator $\widehat{U}_a(x_a)$ for attribute A_a after the first x_a partitions are queried for A_a as:

$$\widehat{U}_a(x_a) = U\left(\sum_{i=0}^{x_a-1} Q_{1,a}(T_i), \sum_{i=0}^{x_a-1} Q_{2,a}(T_i)\right) \quad (2)$$

[1] http://www.tableau.com
[4] https://powerbi.microsoft.com

After x DB requests	$x=1$	$x=2$	$x=3$	$x=4$	$x=5$	$x=6$	$x=7$	$x=8$
A_1's estimation $\widehat{U}_1 \pm r_1$	0.50 ± 0.2	0.50 ± 0.2	0.50 ± 0.2	0.46 ± 0.12	0.46 ± 0.12	0.40 ± 0.08	0.40 ± 0.08	0.37 ± 0.05
A_2's estimation $\widehat{U}_2 \pm r_2$	-	0.45 ± 0.2	0.45 ± 0.2	0.45 ± 0.20	0.49 ± 0.12	0.49 ± 0.12	0.51 ± 0.08	0.51 ± 0.08
A_3's estimation $\widehat{U}_3 \pm r_3$	-	-	0.70 ± 0.2	0.70 ± 0.20	0.70 ± 0.20	0.70 ± 0.20	0.70 ± 0.20	0.70 ± 0.20
Top-K set $\mathcal{I}$	-	-	{1,3}	{1,3}	{2,3}	{2,3}	{2,3}	{2,3}
Alternative top-K set $\mathcal{I}'$	-	-	{2,3}	{2,3}	{1,3}	{1,3}	{1,3}	{2,3}
Next attribute a^*	-	-	1	2	1	2	1	Terminate

Table 1. Running TopKAttribute for Example 3

where $Q_{i,a}(T_1)+Q_{i,a}(T_2)$ is defined to be group-wise summation of counts for corresponding query results. For example, if $Q_{1,a}(T_1) = \{(g_1 : 2, g_2 : 3)\}, Q_{1,a}(T_2) = \{(g_1 : 4, g_3 : 1)\}$, then $Q_{1,a}(T_1) + Q_{1,a}(T_2) = \{(g_1 : 6, g_2 : 3, g_3 : 1)\}$. It is easy to see that $Q_{i,a}(T_1) + Q_{i,a}(T_2) = Q_{i,a}(T_1 \cup T_2)$, and $\widehat{U}_a(S+1) = U_a$. Note that we do not calculate the utility from each partition and then average them, because in general, $U_a \neq \frac{\sum_{i=0}^{S} U(Q_{1,a}(T_i), Q_{2,a}(T_i))}{S+1}$.

• Confidence interval. To bound the estimation error, we define $r_a(x_a)$ as the *confidence radius* of utility U_a such that

$$Pr\left[|U_a - \widehat{U}_a(x_a)| \geq r_a(x_a)\right] \leq \tilde{\delta} \triangleq \frac{\delta}{C(S+1)} \tag{3}$$

i.e., with confidence probability at least $1-\tilde{\delta}$, the true utility U_a falls into the confidence interval $[\widehat{U}_a(x_a) - r_a(x_a), \widehat{U}_a(x_a) + r_a(x_a)]$. Our calculation technique of confidence interval is presented in Section 4.

• Competing attributes. Consider a "bad luck" case for utility estimation: We have overestimated all the current top-K utilities and underestimated the others. We define *revised utility* under that bad luck case as:

$$\widetilde{U}_a \triangleq \begin{cases} \widehat{U}_a(x_a) - r_a(x_a) & a \in \mathcal{I} \\ \widehat{U}_a(x_a) + r_a(x_a) & a \notin \mathcal{I} \end{cases} \tag{4}$$

i.e., we revise the current top-K utilities to be their lower confidence bound, and others upper confidence bound. Then we define *alternative top-K set* $\mathcal{I}'$ as the set of top-K attributes ordered by revised utility (tie is broken by preferring smaller index). We further define $\mathcal{I} \oplus \mathcal{I}' \triangleq (\mathcal{I} \setminus \mathcal{I}') \cup (\mathcal{I}' \setminus \mathcal{I})$ as the set of competing attributes. Competing attributes are among top-K either under the current estimation, or in the "bad luck" case, but not both. For example, in Figure 3, $\mathcal{I}' = \{A_1, A_2, A_3, A_4, A_6\}$. The attributes in $\mathcal{I} \setminus \mathcal{I}'$ (A_5) and $\mathcal{I}' \setminus \mathcal{I}$ (A_6) are competing. Intuitively, shrinking the longest confidence interval among them (A_5 in this example) has good chance to reduce the competition. Section 3.2 will reveal that this simple intuition leads to surprisingly good sampling efficiency.

In Algorithm 1, lines 1-5 query each attribute once using the first partition. QueryDB(P_1, P_2, A_a, x_a) queries T_{x_a-1} for attribute A_a. The result for attribute A_a is stored (line 3) and used to estimate the utility (line 4). Line 5 calculates the confidence radius $r_a(x_a)$. Lines 6-14 iteratively query the database until there is no competing attribute. In each iteration, it first selects the attribute with the longest confidence interval among competing attributes (lines 7-9), and then updates its utility estimation (lines 10-13). In lines 7 and 8, tie is broken by preferring smaller index. In line 9, tie is broken arbitrarily. The algorithm aggregates the query results (line 11) before it updates the estimator (line 12). The specific formula in lines 5 and 13 is presented in Section 4 (ref. Eq. (11)).

Algorithm 1: TopKAttr

Input: U, P_1, P_2, K
Output: Top-K attribute indices $\mathcal{I}$

1 **for** $a \in [C]$ **do**
2 $x_a \leftarrow 1$;
3 $(Q_{1,a}, Q_{2,a}) \leftarrow$ QueryDB(P_1, P_2, A_a, x_a);
4 Estimate $\widehat{U}_a \leftarrow U(Q_{1,a}, Q_{2,a})$;
5 Calculate $r_a(x_a)$;
6 **repeat**
7 $\mathcal{I} \leftarrow$ top-K attribute indices for $\{\widehat{U}_a\}$;
8 $\mathcal{I}' \leftarrow$ alternative top-K indices for $\{\widetilde{U}_a\}$;
9 $a^* \leftarrow \arg\max_{a \in \mathcal{I} \oplus \mathcal{I}'} r_a(x_a)$;
10 $x_{a^*} \leftarrow x_{a^*} + 1$;
11 (Q_{1,a^*}, Q_{2,a^*})+ = QueryDB$(P_1, P_2, A_{a^*}, x_{a^*})$;
12 Update $\widehat{U}_{a^*} \leftarrow U(Q_{1,a^*}, Q_{2,a^*})$;
13 Update $r_{a^*}(x_{a^*})$;
14 **until** $\mathcal{I} = \mathcal{I}'$;

Example 3. (*Running of TopKAttr in a toy example*)
For illustration we use a simplified example of finding top-2 attributes out of 3, i.e., C is 3 and K is 2. The true utility for A_1 to A_3 are 0.35, 0.5 and 0.8, and $\mathcal{I}^* = \{2, 3\}$. The number of equal-sized partitions $S = 10$. The utility estimator, radius and other variables during the algorithm are illustrated in Table 1. The first 3 DB requests are for initialization. We see that the initial estimate to U_1 and U_2 are in the wrong order, but their confidence intervals overlap. Attribute A_3 is among both current top-2 and alternative top-2, so only A_1 and A_2 are competing. After the next DB request, A_1's utility estimation is refined, yet A_1 and A_2 are still competing. A_1 and A_2 are selected to query in turn until their confidence intervals are separate after the 8th DB request. The algorithm returns the correct top-2 attributes. The sampling rate is roughly 5/30=17%. □

Via this toy example, we can see how the algorithm prioritizes competing attributes and saves unnecessary query cost for non-competing attributes (A_3 is queried only once). In real applications, the number of attributes will be much larger and the benefit will be amplified. Also note that the set of competing attributes generally varies as the adaptive querying goes on, though in this particular example it remains the same.

Symbol	Meaning
T_s	the $(s+1)$-th random partition of table T
N	number of rows in table T
C	number of candidate attributes
a	attribute index ranging from 1 to C
c_a	# distinct values or bins in A_a
U_a	utility of attribute A_a
x_a	number of partitions queried for A_a
r_a	confidence radius of U_a
$\widehat{U}_a$	estimated utility for A_a
$\widetilde{U}_a$	revised utility for A_a under a 'bad luck' case
β	selectivity of the more selective predicate of P_1 and P_2
m_i	# rows selected by P_i in queried partitions
n_i	# rows selected by P_i in T

Table 2. Notations

3.2. Theoretical Guarantee

The main theoretical results can be summarized as: (i) our algorithm is proved to return correct top-K attributes with high probability, and (ii) the sampling rate of our algorithm is near optimal.

3.2.1. Correctness. Let $\mathcal{I}^*$ be the set of true top-K attributes' indices, where the tie is broken by preferring smaller indicies.

Theorem 1. (*Probabilistic Guarantee of Correctness*)
With probability at least $1-\delta$, the top-K attributes returned by TopKAttr $\mathcal{I}=\mathcal{I}^*$.

Proof. We denote the event $|U_a-\widehat{U}_a(s)|\le r_a(s)$ as $o_{a,s}$. From Eq. (3), we have $Pr[\neg o_{a,s}]\le\tilde{\delta}$. We use O to denote the event $\cap_{a\in[C],s\in[S]}o_{a,s}$. According to the union bound,

$$Pr[O]=Pr\left[\cap_{a\in[C],s\in[S]}o_{a,s}\right]=1-Pr\left[\cup_{a\in[C],s\in[S]}\neg o_{a,s}\right]$$
$$\ge 1-\sum_{a\in[C],s\in[S]}Pr[\neg o_{a,s}]\ge 1-\sum_{a\in[C],s\in[S]}\tilde{\delta}\ge 1-\delta$$

Now, assume O happens and Algorithm 1 returns with some $\mathcal{I}=\mathcal{I}'$. We prove $\mathcal{I}=\mathcal{I}^*$. To simplify notations, We omit (x_a) when referring to $r_a(x_a)$.

From $\mathcal{I}=\mathcal{I}'$, we know that $\min_{a\in\mathcal{I}}\widetilde{U}_a\ge\max_{b\notin\mathcal{I}}\widetilde{U}_b$. Thus, $\forall a\in\mathcal{I}, b\notin\mathcal{I}$, we have $\widehat{U}_a(x_a)-r_a\ge\widehat{U}_b(x_b)+r_b$. Since O happens, we also have $\widehat{U}_a(x_a)-U_a\le r_a, U_b-\widehat{U}_b(x_b)\le r_b$. They together imply $U_a\ge U_b$. Due to the consistent tie breaker in $\mathcal{I}$ and $\mathcal{I}^*$, $\mathcal{I}=\mathcal{I}^*$. □

A weaker quality measure than correctness is accuracy, which can be defined as the fraction of true top-K attributes among the returned top-K attributes, i.e. $\frac{|\mathcal{I}\cap\mathcal{I}^*|}{K}$. Theorem 1 is also a strong result for accuracy. It guarantees that with high probability, the algorithm returns 100% accurate top-K attributes. TopKAttr is the first randomized algorithm with provable guarantee for top-K attribute recommendation.

3.2.2. Efficiency. The efficiency of the algorithm is mainly determined by the sampling rate, as query processing is the main cost. The analysis of efficiency is difficult due to the randomized partition. There always exists some adversarial case in which the sampling rate is 1. However, we prove probabilistic guarantee of the sampling rate due to the following property:

Definition 2. (*Square Root Convergence Rate*)
A utility function has a square root convergence rate if its confidence radius satisfies: $r_a(s)=\mathrm{O}(m(s)^{-1/2}\log\frac{1}{\delta})$, where $m(s)$ denotes the number of rows in the first s partitions that satisfy the more selective predicate of P_1 and P_2. We use β to denote the selectivity of such predicate. □

Both $\ell 1$ and $\ell 2$ have this property, which is shown in Section 4.

Upper bound of sampling rate for our algorithm. Our upper bound depends on the utility distribution of all attributes. Let $U_{[i]}$ be the i-th largest utility among the C attributes. Intuitively, the closer the utility of an attribute is to the boundary of top-K ($U_{[K]}$ and $U_{[K+1]}$), the harder it is to decide whether it is in $\mathcal{I}^*$. We define the *utility gap* of attribute A_a as:

$$\Delta_a\triangleq\begin{cases}U_a-U_{[K+1]} & a\in\mathcal{I}^*\\ U_{[K]}-U_a & a\notin\mathcal{I}^*\end{cases}\tag{5}$$

To characterize the overall difficulty of detecting top-K, we define j-th order *density* of a utility distribution as: $\rho_j\triangleq\sum_{a=1}^{C}\frac{1}{\Delta_a^j}$. The higher the density is, the more "dense" the utilities are distributed near the boundary of top-K, and the more difficult it is to find top-K.

Lemma 1. (*Upper Bound of #Iterations Per Attribute*)
With probability at least $1-\delta$, $\forall a\in[C]$, the number of iterations x_a for attribute A_a to be selected for querying in Algorithm 1 satisfies:

$$x_a\le\left\lceil S\left(\frac{C_1}{\Delta_a^2\beta N}+\frac{C_2}{\Delta_a\beta\sqrt{\beta}N}+\frac{\log(1/\tilde{\delta})}{\beta^2 N}\right)\right\rceil\tag{6}$$

where C_1, C_2 are small logarithmic factors $\mathrm{O}(\log^2\tilde{\delta})$.

Proof. First, we prove by contradiction that when the event O (as defined in the previous proof) happens, $r_{a^*}\ge\frac{\Delta_{a^*}}{4}$ in every iteration. Assume $r_{a^*}<\frac{\Delta_{a^*}}{4}$. It is not hard to prove $r_{a^*}>0$. So $0<r_{a^*}<\frac{\Delta_{a^*}}{4}$. We discuss the following 4 cases.

(i) $a^*\in\mathcal{I}^*, a^*\in\mathcal{I}\setminus\mathcal{I}'$. $\Delta_a=U_a-U_{[K+1]}$. Since $\mathcal{I}'\neq\mathcal{I}^*$, there exists $a\in\mathcal{I}'\setminus\mathcal{I}^*$. If $a\in\mathcal{I}$, $U_a\ge\widehat{U}_a-r_a\ge\widehat{U}_{a^*}-r_{a^*}\ge U_{a^*}-2r_{a^*}>U_{[K+1]}$. If $a\notin\mathcal{I}$,

$$U_a\ge\widehat{U}_a-r_a\ge\widehat{U}_a+r_a-2r_a$$
$$\ge\widehat{U}_{a^*}-r_{a^*}-2r_{a^*}\ge U_{a^*}-4r_{a^*}>U_{[K+1]}$$

Both imply $a\in\mathcal{I}^*$ as a contradiction.

(ii) $a^*\in\mathcal{I}^*, a^*\in\mathcal{I}'\setminus\mathcal{I}$. $\Delta_a=U_a-U_{[K+1]}$. Since $\mathcal{I}\neq\mathcal{I}^*$, there exists $a\in\mathcal{I}\setminus\mathcal{I}^*$. If $a\notin\mathcal{I}'$, $U_a\ge\widehat{U}_a-r_a\ge\widehat{U}_{a^*}-r_{a^*}\ge U_{a^*}-2r_{a^*}>U_{[K+1]}$ which is a contradiction. If $a\in\mathcal{I}'$, we know that $\mathcal{I}'\neq\mathcal{I}^*$, so there exists $b\in\mathcal{I}^*\setminus\mathcal{I}'$. If $b\in\mathcal{I}$, $U_a\ge\widehat{U}_a-r_a\ge\widehat{U}_b-r_b\ge\widehat{U}_{a^*}-r_{a^*}>U_{[K+1]}$ which is a contradiction. If $b\notin\mathcal{I}$, $U_a\ge\widehat{U}_a-r_a\ge\widehat{U}_b+r_b\ge U_b$. The equality holds only if $U_a=U_b$ and $\widetilde{U}_a=\widetilde{U}_b$. Since $a\in\mathcal{I}'$ and $b\notin\mathcal{I}'$, we have $a<b$. Then it implies $a\in\mathcal{I}^*$, which is a contradiction.

(iii) $a^*\notin\mathcal{I}^*, a^*\in\mathcal{I}\setminus\mathcal{I}'$. $\Delta_a=U_{[K]}-U_a$. Since $\mathcal{I}^*\neq\mathcal{I}$, there exists $a\in\mathcal{I}^*\setminus\mathcal{I}$. If $a\in\mathcal{I}'$, $U_a\le\widehat{U}_a+r_a\le\widehat{U}_{a^*}+r_{a^*}\le U_{a^*}+2r_{a^*}<U_{[K]}$ which is a contradiction. If $a\notin\mathcal{I}'$, since $\mathcal{I}'\neq\mathcal{I}^*$, there exists $b\in\mathcal{I}'\setminus\mathcal{I}^*$. If $b\notin\mathcal{I}$, $U_a\le\widehat{U}_a+r_a\le\widehat{U}_b+r_b\le\widehat{U}_{a^*}+r_{a^*}<U_{[K]}$ which is a contradiction. If $b\in\mathcal{I}$, $U_a\le\widehat{U}_a+r_a\le\widehat{U}_b-r_b\le U_b$. The equality holds only if $U_a=U_b$ and $\widetilde{U}_a=\widetilde{U}_b$. Since $a\notin\mathcal{I}'$ and $b\in\mathcal{I}'$, we

have $a > b$. Then it implies $a \notin \mathcal{I}^*$, which is a contradiction.
(iv) $a^* \notin \mathcal{I}^*, a^* \in \mathcal{I}' \setminus \mathcal{I}$. $\Delta_a = U_{[K]} - U_a$. Since $\mathcal{I}^* \neq \mathcal{I}'$, there exists $a \in \mathcal{I}^* \setminus \mathcal{I}'$. If $a \in \mathcal{I}$,

$$U_a \leq \widehat{U}_a + r_a = \widehat{U}_a - r_a + 2r_a$$
$$\leq \widehat{U}_{a^*} + r_{a^*} + 2r_{a^*} \leq U_{a^*} + 4r_{a^*} < U_{[K]}$$

If $a \notin \mathcal{I}$, $U_a \leq \widehat{U}_a + r_a = \widehat{U}_{a^*} + r_{a^*} \leq U_{a^*} + 2r_{a^*} < U_{[K]}$. Both imply $a \notin \mathcal{I}^*$, which is a contradiction.

All the 4 cases lead to contradiction. So $r_{a^*} \geq \frac{\Delta_{a^*}}{4}$. That implies

$$r_a(x_a - 1) \geq \frac{\Delta_a}{4} \tag{7}$$

because for x_a to increase, a must be chosen as a^* in some iteration.

By Definition 2, there exists a constant C_0 s.t.

$$r_a(x_a - 1) \leq \frac{C_0}{\sqrt{m(x_a - 1)}} \log \frac{1}{\tilde{\delta}}$$

Combining with Eq. (7), we have:

$$m(x_a - 1) \leq \frac{C_0^2 \log^2 \tilde{\delta}}{r_a(x_a - 1)^2} \leq \frac{16 C_0^2 \log^2 \tilde{\delta}}{\Delta_a^2} \tag{8}$$

By Hoeffding-Serfling inequality (Theorem 2.4 in [4]), with probability at least $1 - \tilde{\delta}$,

$$\frac{m(x_a - 1)S}{N(x_a - 1)} \geq \beta - \sqrt{\frac{-(S - x_a + 1)\log \tilde{\delta}}{2N(x_a - 1)}} \tag{9}$$

We denote the event that the above equation is true for all $a \in [C]$ as O'. We have: $Pr[O \cap O'] \geq 1 - \tilde{\delta} C(S+1) = 1 - \delta$. Now assume that both O and O' happen. Eq. (8) and (9) imply:

$$x_a \leq \begin{cases} \left\lceil S\left(\frac{C_1}{\Delta_a^2 \beta N} + \frac{C_2}{\Delta_a \beta \sqrt{\beta N}} + \frac{\log(1/\tilde{\delta})}{\beta^2 N}\right) \right\rceil & \Delta_a > \Delta_0 \\ S & \Delta_a \leq \Delta_0 \end{cases}$$

where $C_1 = 16 C_0^2 \log^2 \tilde{\delta}, C_2 = 4 C_0 \log^2 \tilde{\delta}$, and $\Delta_0 = \frac{8 C_0 \log(1/\tilde{\delta})}{\sqrt{2\beta N + \sqrt{\beta^2 N^2 - \frac{N \log \tilde{\delta}}{2}}}}$.

When $\Delta_a \leq \Delta_0$, $\frac{C_1}{\Delta_a^2 \beta N} > 1$. So with probability at least $1 - \delta$, Eq. (6) is true for all $a \in [C]$. □

Theorem 2. (*Upper Bound of Sampling Rate*)
The sampling rate of Algorithm 1 has the following bound with probability at least $1 - \delta$:

$$\phi = \frac{x}{CS} \leq \frac{C_1 \rho_2}{C\beta N} + \frac{C_2 \rho_1}{C\beta\sqrt{\beta N}} + \frac{\log(1/\tilde{\delta})}{\beta^2 N} = \begin{cases} \widetilde{O}\left(\frac{\rho_2}{C\beta N}\right) & \rho_2 = \Omega\left(\frac{1}{\beta}\right) \\ \widetilde{O}\left(\frac{1}{C\beta^2 N}\right) & \rho_2 = O\left(\frac{1}{\beta}\right) \end{cases}$$

where C_1 and C_2 are small logarithmic factors and omitted in the $\widetilde{O}$ notation.

Proof. With Lemma 1, we sum up Eq. (6) for all $a \in [C]$, and apply the inequality $\lceil x \rceil \leq x + 1$ to prove the theorem. □

An important implication of this theorem is that the upper bound of query cost ϕCN (i.e., the sample complexity) does not grow with data size CN when density ρ_2 and selectivity β are invariant. The larger the table, the more speedup over the exact approach.

Lower bound of sampling rate for any algorithm. We prove the following theorem with nearly matching lower bound.

Theorem 3. (*Lower Bound of Sampling Rate*)
Every algorithm requires $\Omega(\frac{\rho_2}{C\beta N})$ sampling rate to find top-K attributes with probability at least 2/3.

Proof. To prove the case for $\ell 2$, we need the following 'closeness-test' lemma (Theorem 1.2 in [6]): With $b = \max\{\|\mathbf{p}\|_2, \|\mathbf{q}\|_2\}$, it requires $\Omega(\frac{\sqrt{b}}{\epsilon^2})$ samples to distinguish the case that $\mathbf{p} = \mathbf{q}$ from the case that $\|\mathbf{p} - \mathbf{q}\|_2 > 2\epsilon$ with success probability at least $\frac{2}{3}$.

It follows that, given K pairs of unknown distributions $(\mathbf{p_a}, \mathbf{q_a}), a \in [K]$ such that $1 = \max\{\|\mathbf{p_a}\|_2, \|\mathbf{q_a}\|_2\}$, for each pair $(\mathbf{p_a}, \mathbf{q_a})$, using fewer than $\Omega(\frac{1}{\epsilon^2})$ samples from each of $\mathbf{p_a}$ and $\mathbf{q_a}$, no algorithm can distinguish the two cases:

i. $\|\mathbf{p_a} - \mathbf{q_a}\|_2 > 2\epsilon$ and
ii. $\mathbf{p_a} = \mathbf{q_a}$.

Assume algorithm Alpha can correctly identify top-K attributes with probability at least $\frac{2}{3}$. We design a tester Beta, which has a special database to feed algorithm Alpha. First, we let $C = 2K$. Attribute A_1 to A_K have cardinality d equal to the number of dimensions in $\mathbf{p_a}$. Attribute A_{K+1} to A_{2K} are binary. A special binary attribute A_{2K+1} is used for predicates and not included in candidate attributes. Let $N_{a,i,j}, i \in [d], j \in \{0, 1\}$ be the number of rows in T that have value i and j for attribute A_a and A_{2K+1} respectively. It is not hard to construct a table T such that

$$\frac{N_{a,i,j}}{N} = \begin{cases} \frac{p_{a,i}}{2} & a \in [K], i \in [d], j = 0 \\ \frac{1}{2} & a \in [K+1, 2K], i = 0, j = 0 \\ 0 & a \in [K+1, 2K], i = 1, j = 0 \\ \frac{q_{a,i}}{2} & a \in [K], i \in [d], j = 1 \\ \frac{(1 - \epsilon/\sqrt{2})}{2} & a \in [K+1, 2K], i = 0, j = 1 \\ \frac{\epsilon}{2\sqrt{2}} & a \in [K+1, 2K], i = 1, j = 1 \end{cases}$$

Let P_1 be the predicate $A_{2K+1} = 0$ and P_2 be $A_{2K+1} = 1$. $\beta = \frac{1}{2}$. The true $\ell 2$ utility for attribute $K+1$ to $2K$ is $\frac{\epsilon}{\sqrt{2}}$, and the true $\ell 2$ utility for attribute 1 to K is either 0 or larger than $\sqrt{2}\epsilon$. For all $a \in [C]$, $\Delta_a \geq \frac{\epsilon}{\sqrt{2}}$. $\forall a \in [K]$, attribute A_a is among the true top-K attributes if case i happens, and not among the true top-K if case ii happens.

After Alpha terminates, Beta outputs testing result for $(\mathbf{p_a}, \mathbf{q_a}), a \in [K]$ in the following way: if $a \in \mathcal{I}$, output case i; otherwise output case ii. We know that Alpha can find true top-K attributes with probability at least 2/3. Therefore, tester Beta can test between case i and case ii with success probability at least 2/3, for all the pairs $(\mathbf{p_a}, \mathbf{q_a}), a \in [K]$. By 'closeness-test' Lemma, the total number of samples Beta needs for attribute A_a is $\Omega(\frac{1}{\epsilon^2})$. It follows that the total number of samples Alpha draws is $\Omega(\frac{K}{\epsilon^2}) = \Omega(\sum_{a=1}^{2K} \frac{1}{\Delta_a^2}) = \Omega(\rho_2)$. And the sampling rate is $\Omega(\frac{\rho_2}{C\beta N})$.

To prove the case for $\ell 1$, we need a different lemma about the hardness of closeness test in $\ell 1$ distance (Theorem 1.1 in [6]). The proof process is similar. □

Together with the upper bound, the theorem implies that our algorithm is optimal when $\rho_2 = \Omega(\frac{1}{\beta})$, i.e., when the density dominates the selectivity as the main source of difficulty.

4. CONFIDENCE INTERVAL

Standard techniques of confidence interval estimation such as [15, 25] are not applicable to our problem, as we consider more complex utility function than the mean of a set of numbers. Heuristic application of the standard techniques as in SeeDB [22] results in biased and uninformative confidence intervals. To resolve this challenge, we develop a new technique for calculating confidence interval with respect to the number of aggregated rows.

4.1. Derivation

We first present a new view of the utility estimator, which offers a fine model of the relation between the utility and the data tuples. For ease of discussion, we assume a and x_a are given and use $r, \widehat{U}$ and c as the abbreviations for $r_a(x_a)$, $\widehat{U}_a(x_a)$ and c_a.

Let n_i and m_i be the number of tuples that satisfy P_i in T and in the first x_a partitions respectively. The utility depends on the two sets of n_1 and n_2 tuples only. They are present in T_0 to T_S in a random order, so each tuple can be modeled as a sample without replacement. We define $\mathbf{Z}_i \triangleq (Z_{ij})_{j=1}^{n_i}$ as the random permutation vector where the variable $Z_{ij} \in [n_i]$ indicates the j-th sample's index among the n_i tuples. Then, the estimator $\widehat{U}$ can be viewed as a function of two permutation vectors of length n_1 and n_2 respectively. $\widehat{U}$ is only sensitive to the first m_i elements in $\mathbf{Z}_i, i = 1, 2$.

Theorem 4. (*Confidence Radius for $\ell 1$*)
For $\ell 1$, the radius $r^{\ell 1}$ can be calculated as:

$$
\begin{aligned}
r^{\ell 1} &= \sum_{k=1}^{2}\left[h_k + g_k\sqrt{2\log\frac{\sum_{j=1}^{2}\sqrt{g_j}}{\tilde{\delta}\sqrt{g_k}}}\right] \\
h_k &\triangleq \frac{1}{2}\sqrt{\frac{(c-1)(n_k-m_k)}{m_k(n_k-1)}} \qquad (10)\\
g_k &\triangleq \frac{1}{m_k}\sqrt{(n_k-m_k)^2\sum_{j=n_k-m_k+1}^{n_k}\frac{1}{j^2}} \le \sqrt{\frac{n_k-m_k}{m_k n_k}}
\end{aligned}
$$

Proof. In this proof, U refers to $\ell 1(\mathbf{p}_1, \mathbf{p}_2)$, and $\widehat{U} = \ell 1(\mathbf{q}_1, \mathbf{q}_2)$. First, by triangle inequality,

$$|\|\mathbf{q}_1 - \mathbf{q}_2\|_1 - \|\mathbf{p}_1 - \mathbf{p}_2\|_1| \le \|\mathbf{q}_1 - \mathbf{p}_1\|_1 + \|\mathbf{q}_2 - \mathbf{p}_2\|_1$$

We define $f_k = \ell 1(\mathbf{q}_k, \mathbf{p}_k)$. Using Cauchy-Schwarz inequality,

$$
\begin{aligned}
\|\mathbf{q} - \mathbf{p}\|_1^2 &= (\sum_{i=1}^{c}|q_i - p_i|)^2 = (\sum_{i=1}^{c}\frac{|q_i - p_i|}{\sqrt{p_i}}\sqrt{p_i})^2 \\
&\le \sum_{i=1}^{c}\frac{(q_i - p_i)^2}{p_i}
\end{aligned}
$$

So by Jensen's inequality, $E[f_k] \le h_k = \frac{1}{2}\sqrt{\frac{(c-1)(n_k-m_k)}{m_k(n_k-1)}}$.

Second, we bound the maximal change of $f_k(\mathbf{Z}_k) = \|\mathbf{q}_k - \mathbf{p}_k\|_1$ with a perturbed permutation vector $\mathbf{Z}_k^{ij}, i \in [m_k], j \in [m_k+1, n_k]$ (obtained by swapping Z_{ki} with Z_{kj}). Without loss of generality, we assume the i-th tuple has value 1 and j-th tuple has value 2 for attribute A_a.

$$
\begin{aligned}
|f_k(\mathbf{Z}_k) - f(\mathbf{Z}_k^{ij})| = \frac{1}{2}\Big|&|\frac{m_{k1}}{m_k} - \frac{n_{k1}}{n_k}| + |\frac{m_{k2}}{m_k} - \frac{n_{k2}}{n_k}| \\
&- |\frac{m_{k1}-1}{m_k} - \frac{n_{k1}}{n_k}| - |\frac{m_{k2}+1}{m_k} - \frac{n_{k2}}{n_k}|\Big| \le \frac{1}{m_k}
\end{aligned}
$$

Applying Lemma 2 in [11] to $f_k(\mathbf{Z}_k)$, we have:

$$Pr[f_k - E[f_k] \ge \epsilon_i] \le e^{-\frac{\epsilon^2}{2g_k^2}}$$

Set $\epsilon_k = g_k\sqrt{2\log\frac{\sum_{j=1}^{2}\sqrt{g_j}}{\tilde{\delta}\sqrt{g_k}}}$. We have:

$$Pr[f_k - E[f_k] \ge \epsilon_k] \le \frac{\sqrt{g_k}}{\sum_{j=1}^{2}\sqrt{g_j}}\tilde{\delta} \Rightarrow Pr[f_k \ge h_k + \epsilon_k] \le \frac{\sqrt{g_k}}{\sum_{j=1}^{2}\sqrt{g_j}}\tilde{\delta}$$

With union bound,

$$
\begin{aligned}
Pr\left[\sum_{k=1}^{2} f_k \ge \sum_{k=1}^{2} h_k + \epsilon_k\right] &\le Pr[\cup_{k=1}^{2}[f_k \ge h_k + \epsilon_k]] \\
&\le \sum_{k=1}^{2} Pr[f_k \ge h_k + \epsilon_k] \le \tilde{\delta}
\end{aligned}
$$

Finally, due to $|\widehat{U} - U| \le \sum_{k=1}^{2} f_k$, we have $Pr(|\widehat{U} - U| \ge r^{L_1}) \le \tilde{\delta}$. □

Theorem 5. (*Confidence Radius for $\ell 2$*)
The radius $r^{\ell 2}$ for $\ell 2$ follows Eq. (10), with h_k's definition replaced by $h_k \triangleq \sqrt{\frac{(c-1)(n_k-m_k)}{2cm_k(n_k-1)}}$.

The proof is similar to the case of $\ell 1$ and omitted. It is easy to verify the square root convergence rate.

4.2. Practical Discussion

The confidence intervals are dependent on n_1 and n_2, the number of tuples satisfying the predicates P_1 and P_2. In general, these numbers are unknown before the predicates are evaluated in the full data. To apply the theorems, we can use the inequality $n_k \le N$ and $g_k < \sqrt{\frac{N-m_k}{m_k N}}$ to get an upper bound of the radius. However, N is a rather rough bound of n. The following theorem offers a tighter probabilistic upper bound of n.

Theorem 6. (*Upper Bound of n*)
With probability higher than $1 - \frac{\tilde{\delta}}{2}$,

$$
\begin{aligned}
n_k &\le max(\acute{n}_k, \widehat{n}_k), \acute{n}_k \triangleq N\left(\frac{m_k}{M} + \sqrt{\frac{\rho_M \log(2/\tilde{\delta})}{2M}}\right) \\
\widehat{n}_k &\triangleq N\left(\frac{m_k}{M} + \hat{\sigma_k}\sqrt{\frac{2\rho_M \log(10/\tilde{\delta})}{M}} + \frac{z\log(10/\tilde{\delta})}{M}\right)
\end{aligned}
$$

where M is the total number of tuples in the first x_a partitions, $\hat{\sigma} \triangleq \frac{\sqrt{m_k(M-m_k)}}{M}$, $z = (7/3 + 3/\sqrt{2})$, and

$$\rho_M \triangleq \begin{cases} 1 - \frac{M-1}{N} & M \le \frac{N}{2} \\ (1 - \frac{M}{N})(1 + \frac{1}{M}) & M > \frac{N}{2} \end{cases}$$

Proof. We can prove it using Theorem 2.4 and 4.3 in [4], and union bound. □

The dominant factor in our confidence radius is the inverse square root of the aggregated rows, while the previous confidence radius in [22] is dominated by the inverse square root of partitions. For

Name	Rows	Attributes	Size	Partitions
TPCH3	300M	17	108G	10
TPCH6	600M	17	211G	10
TPCH12	1200M	17	410G	10
Flight	168M	77	58G	4
Prod	341M	225	195G	7

Table 3. Datasets

comparison, when a partition contains millions of rows, our result can be tighter by 3 orders of magnitude. That difference has a big impact on the effectiveness of sample-based pruning.

Note that our calculation method is agnostic of data distributions and has generic applicability. In practice, if we only keep the dominating factor in the confidence interval calculation, it still works well when m is large. In experiments, we use the following simplified formulas.

$$r^{\ell 1} = \sum_{k=1}^{2} \sqrt{\frac{(c-1)\log\frac{2}{\delta}}{2m_k(1-\frac{M}{N})}}, r^{\ell 2} = \frac{r^{\ell 1}}{c} \tag{11}$$

5. EXPERIMENTAL EVALUATION

This section presents an experimental evaluation of the techniques proposed in this paper. The goals of the study are: (a) compare the TopKAttr algorithm with the exact approach and the state-of-the-art approximate approaches and (b) isolate the individual impacts of the adaptive querying algorithm and the confidence interval calculation.

5.1. Experimental Setting

Datasets. We use both synthetic and real-world datasets.

• Synthetic data. We use the TPC-H benchmark[2] and consider the join of all the relations as the dataset to explore. We remove long text fields that contain random strings, such as c_phone. We create three TPC-H datasets of different size. They are named TPCH3, TPCH6 and TPCH12, which contain 300, 600 and 1200 million rows.

• Flight. We use the flight on-time performance dataset from Bureau of Transportation[3]. Each row is a flight record from 1987-2016. The attributes can be categorized as time periods, airline, origin, destination, departure and arrival performance, cancellations and diversions, cause of delay, gate return information and so on.

• Prod. We use a production dataset from a large company. The dataset contains user demographics such as gender and age groups, and product usage activities aggregated in a period.

The statistics of the datasets are summarized in Table 3. Note that our technique has an even larger advantage for datasets with more than hundreds of columns. But such datasets are beyond the reach of most relational DBMSs due to column size limitation, so we do not include them in order to have a consistent evaluation environment.

Test case generation. For each dataset, we generate random test cases for the two utility functions. For each test case, we randomly generate equality and range predicates covering different attributes, with selectivity varying from 10^{-9} to 1. All categorical attributes except predicate attributes are used as candidate attributes. We enumerate the number K of top attributes to return in $\{1, 3, 5, 10\}$, as these are natural choices. We generate 1224 test cases in total. An example test case is shown in Table 4.

Metrics. We evaluate efficiency and correctness. To evaluate efficiency, we calculate the speedup of each sampling-based approach over the exact approach. To evaluate correctness, we report the error rate of each algorithm.

Methods compared.

• Exact approach. We use SQL Server 2016 which supports highly-efficient columnstore, and enable multi-thread query processing. We tried the sharing-based optimization proposed by [22], but did not find any performance gain, probably due to the highly optimized query processing by the DB engine. So this is a strong baseline. Rowstore is too slow to query for these datasets.

• SeeDB$_{\text{CI}}$. The sampling and pruning method proposed in [22] based on confidence interval. Note that the definition of confidence interval in [22] is different from this work, as discussed in Section 4.

• SeeDB$_{\text{MAB}}$. The multi-armed bandit alike pruning in [22]. The two SeeDB methods are the best known approximate methods.

• TopKAttr as outlined in Algorithm 1. δ's impact to the speedup and error rate is negligible when it varies from 0.3 to 0.9. We fix $\delta = 0.5$ in experiments.

♢ RoundRobin. This is an algorithm for ablation study. The only difference from TopKAttr is that it chooses attributes to query in a round-robin fashion. It is implemented by replacing line 9 in Algorithm 1 with $a^* \leftarrow a^* \% C + 1$.

♢ TopKAttr$_{\text{SeeDBCI}}$. This is an algorithm for ablation study. The only difference from TopKAttr is that it uses SeeDB's definition of confidence interval.

All the experiments are performed in a Azure VM with 16 cores. For preprocessing, we need to choose the size of each partition such that the overhead of sequentially querying partitioned data is low. A good size can be found by an offline tuning process: use no predicates, and increase the number of partitions until the overhead is larger than a threshold (e.g., 10%). In our testing environment, we found that 50-60 million rows in one partition is generally a good size, and ended up with 10, 4 and 7 partitions for TPCH6, Flight and Prod accordingly. TPCH3 and TPCH12 have the same partition size as TPCH6, for the purpose of controlled comparison. All the compared methods share identical preprocessing.

5.2. Comparison of Various Approaches

Figure 4 plots the tail distribution of the speedup in three datasets, and the error rate in all testcases. For more than 85% test cases of TPCH12, TopKAttr is 5-25× faster than the exact approach, and the average speedup is 13×. In contrast, the highest speedup of SeeDB$_{\text{CI}}$ and SeeDB$_{\text{MAB}}$ over the exact approach are 1.2× and 2.3×, and the average speedup is only 0.7× and 1.5× respectively. It is surprising that the average speedup of SeeDB$_{\text{CI}}$ is below 1×. A further investigation reveals that its average sampling rate is above 96%, which is not low enough to counter the overhead due to querying partitioned data. This confirms that the exact approach using a mature commercial DBMS is not easy to beat.

[2] http://www.tpc.org/tpch/default.asp
[3] https://www.transtats.bts.gov/Fields.asp?Table_ID=236

Question	Input	Top attributes	Discovered insight
What distinguishes Alaska airline from other airlines	P_1: UniqueCarrier=AS, P_2 : ∅, Utility: Euclidean	**DivReachedDest**, DestState, OriginState	Higher fraction of diverted flights do not reach scheduled destination

Table 4. Example test case and result in Flight data; insight explained for attributes in bold

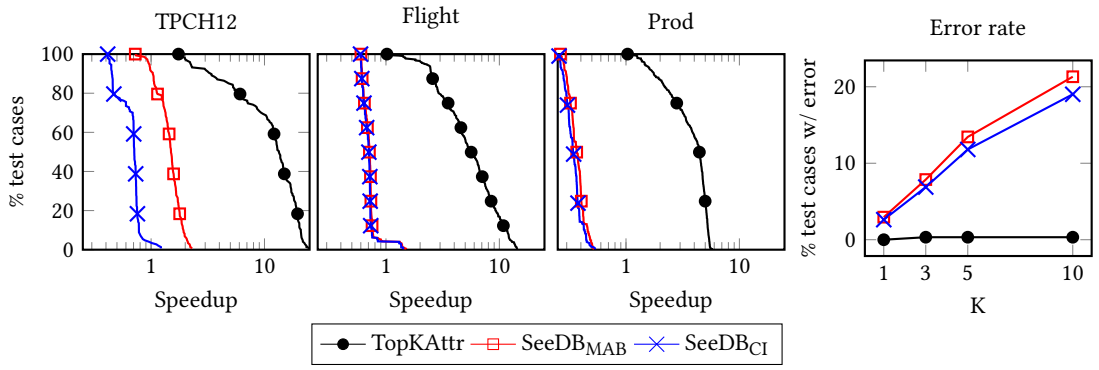

Fig. 4. Tail distribution of log-scale speedup (the first three plots) and the error rate (the fourth plot) in all test cases

Regarding correctness, TopKAttr makes only 3 mistakes among all 1224 test cases, resulting in a near-zero error rate 3/1224=0.00245. The accuracy of these 3 top-K answers is 70%, 80% and 90%. As a comparison, SeeDB_{CI} and $\text{SeeDB}_{\text{MAB}}$ produce wrong top-K in 19% and 21% test cases respectively when K=10. The accuracy of the wrong top-K answers from SeeDB is as low as 0%, 0%, 40%, 40% for K=1, 3, 5, 10 respectively. So the advantage of TopKAttr in correctness is also significant.

In summary, TopKAttr is a clear winner in both efficiency and correctness. It has a dominant speedup across all datasets, with almost zero error.

5.3. Ablation Study

This section evaluates the impact of the two sources of efficiency improvement in TopKAttr: the adaptive querying algorithm and the confidence interval calculation. We compare with RoundRobin (adaptive querying replaced) and $\text{TopkAttr}_{\text{SeeDBCI}}$ (the confidence interval calculation replaced) to see the impact of removing each source. Figure 5 shows the sampling rate of each algorithm. For synthetic datasets, we vary the number of rows from 300M to 1200M in TPCH3 to TPCH12 (Figure 5a). For example, the sampling rate of TopKAttr, RoundRobin, $\text{SeeDB}_{\text{MAB}}$ and $\text{TopKAttr}_{\text{SeeDBCI}}$ for $\ell 1$ is 7%, 38%, 70% and 90% respectively in TPCH6. And we vary K for Flight dataset (Figure 5b). For example, the sampling rate of TopKAttr and RoundRobin for $\ell 2$ is 9% and 72% when K=3, whereas the other two methods have close to 100% sampling rate.

From the result, it is clear that the sample size required by TopKAttr is significantly lower than competing methods, which explains the superior performance in efficiency. Moreover, both the confidence interval calculation and the adaptive querying algorithm are critical. When proper confidence interval is used, the adaptive selection of the next attribute to query is the key to saving unnecessary cost. Comparing TopKAttr and RoundRobin, we see more than 60% of the cost reduction due to the adaptation in Flight dataset. On the other hand, the algorithm does not work as well when improper confidence interval is used – the performance of $\text{TopkAttr}_{\text{SeeDBCI}}$ is worse than baseline as seen in Figure 5a.

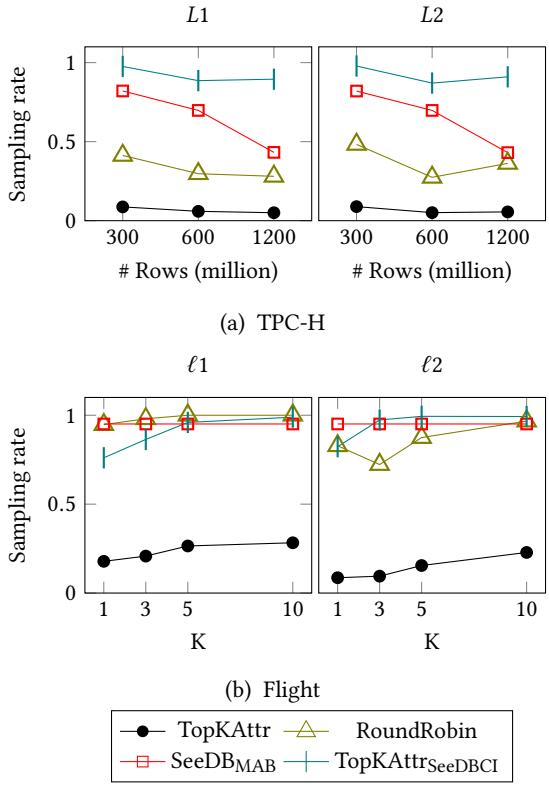

Fig. 5. The impact of adaptive querying and confidence interval

6. RELATED WORK

We review related work in different areas.

Attribute recommendation. Research in modern exploratory analysis of multi-dimensional data has recognized a growing demand of automated query recommendation [21], which dates back to two decades ago in the context of data cubes [18]. We mainly review the attribute recommendation work in the ad-hoc exploration setting. Hierarchical Clustering Explorer (HCE) [20] ranks attributes to visualize distributions for 1 or 2 dimensions. The ranking criteria include modality, entropy etc.. VizDeck [14] generates all possible queries for 2-D visualizations of a dataset in a dashboard and ranks them with a combination of heuristics and supervised training. The features used for training include number of distinct values, entropy, variation etc.. Voyager [24] shows all univariate summaries for each attribute prior to user interaction. Once users select some elements of the query, it suggests additional attributes beyond those explicitly selected, without ranking. Anand and Talbot [3] suggests one additional attribute to partition the data such that conditional correlations between two attributes are revealed. SeeDB [22] is the first work to handle large datasets, and it is extended by [10, 19] to explore numeric attributes.

Feature selection. Filter-based feature ranking algorithms are commonly used for large datasets due to its computational efficiency. Each feature is scored according to a ranking function, such as chi-square and information gain [12]. The goal of feature selection is to maximize the accuracy of a classifier or regressor while minimizing the model complexity. Our goal is to recommend attributes for data scientists to explore ad-hoc subsets and find insights. Hence, both the ranking criterion and the data to compute for the ranking (static training data v.s. ad-hoc subsets specified via predicates) are different. These differences hinder application of previous sampling techniques designed for feature selection, such as [16], to attribute recommendation.

Multi-armed bandit. Researchers have studied the best arm identification problem, which aims to find the arms with highest expectation of reward using adaptive pulls of the arms [5, 7]. They target applications where the reward is repeatedly sampled from a distribution, such as scientific experiments and clinical trials. In our problem, the ranking criterion is a function of samples and cannot be directly sampled from a distribution, so these methods are not applicable. But the adaptive sampling idea has inspired both [22] and our work.

Sampling for data mining. As Cormode and Duffield [9] pointed out, the interplay between sampling and data mining is an important subject in the big data era, but not well understood in general. A few recent studies represent the advances of sampling techniques in frequent pattern mining [17], collaborative filtering [8], and graph mining [2]. The problem of attribute recommendation bears similarity to the problem of mining discriminative patterns [1] (of length 1). Our confidence interval result and the adaptive sampling technique can be potentially integrated with a pattern mining algorithm for mining longer patterns.

7. CONCLUSION

In this work, we address the efficiency of attribute recommendation, and develop a new solution based on the principle of adaptively querying randomly partitioned data. Our solution has provable correctness and near-optimal sample complexity with high probability. It vastly outperforms previous sampling approaches in both correctness and efficiency.

Our techniques have a few other desirable traits, which make them promising to solve broader problems. First, although this paper focuses on recommending top-K attributes, the algorithm and the theoretical guarantee apply to the generic problem of finding top-K utilities with square root convergence rate from algebraic queries. Second, our confidence interval results enable robustly bounding the error of sample-estimated statistical distances $\ell 1$ and $\ell 2$, which are fundamental metrics in data mining.

Acknowledgments. We would like to thank Surajit Chaudhuri, Vivek Narasayya, Christian Konig, Wentao Wu for their insightful discussion, and the reviewers for their constructive feedback.

REFERENCES

[1] C. C. Aggarwal, and J. Han. *Frequent Pattern Mining*. Springer. 2014.
[2] N. K. Ahmed, N. Duffield, T. L. Willke, and R. A. Rossi. "On Sampling from Massive Graph Streams." *Proc. VLDB Endow.* 10 (11): 1430–1441. 2017.
[3] Anushka Anand, and Justin Talbot. "Automatic Selection of Partitioning Variables for Small Multiple Displays." *IEEE Transactions on Visualization & Computer Graphics* 22 (1): 669–677. 2016.
[4] R. Bardenet, and O.-A. Maillard. "Concentration Inequalities for Sampling without Replacement." *Bernoulli* 21 (3): 1361–1385. Aug. 2015.
[5] S. Bubeck, T. Wang, and N. Viswanathan. "Multiple Identifications in Multi-Armed Bandits." In *ICML'13*. 2013.
[6] S.-O. Chan, I. Diakonikolas, G. Valiant, and P. Valiant. "Optimal Algorithms for Testing Closeness of Discrete Distributions." In *SODA'14*. 2014.
[7] S. Chen, T. Lin, I. King, M. R. Lyu, and W. Chen. "Combinatorial Pure Exploration of Multi-Armed Bandits." In *NIPS'14*. 2014.
[8] T. Chen, Y. Sun, Y. Shi, and L. Hong. "On Sampling Strategies for Neural Network-Based Collaborative Filtering." In *KDD'17*. 2017.
[9] G. Cormode, and N. Duffield. "Sampling for Big Data: A Tutorial." In *KDD'14*. 2014.
[10] H. Ehsan, M. A. Sharaf, and P. K. Chrysanthis. "Efficient Recommendation of Aggregate Data Visualizations." *IEEE Transactions on Knowledge and Data Engineering* 30 (2): 263–277. Feb. 2018.
[11] R. El-Yaniv, and D. Pechyony. "Stable Transductive Learning." In *COLT'06*. 2006.
[12] J. Han, and M. Kamber. *Data Mining: Concepts and Techniques*. Morgan Kaufmann Publishers. 2001.
[13] S. Kandel, R. Parikh, A. Paepcke, J. M. Hellerstein, and J. Heer. "Profiler: Integrated Statistical Analysis and Visualization for Data Quality Assessment." In *Proc. Intl. Working Conf. on Advanced Visual Interfaces*. AVI'12. 2012.
[14] A. Key, B. Howe, D. Perry, and C. Aragon. "VizDeck: Self-Organizing Dashboards for Visual Analytics." In *SIGMOD'12*. 2012.
[15] A. Kim, E. Blais, A. G. Parameswaran, P. Indyk, S. Madden, and R. Rubinfeld. "Rapid Sampling for Visualizations with Ordering Guarantees." *PVLDB* 8 (5). 2015.
[16] H. Liu, H. Motoda, and L. Yu. "A Selective Sampling Approach to Active Feature Selection." *Artificial Intelligence* 159 (1): 49–74. 2004.
[17] M. Riondato, and E. Upfal. "Mining Frequent Itemsets Through Progressive Sampling with Rademacher Averages." In *KDD'15*. 2015.
[18] S. Sarawagi. "Explaining Differences in Multidimensional Aggregates." In *VLDB'99*. 1999.
[19] T. Sellam, and M. Kersten. "Fast, Explainable View Detection to Characterize Exploration Queries." In *SSDBM'16*. 2016.
[20] J. Seo, and B. Shneiderman. "A Rank-by-Feature Framework for Interactive Exploration of Multidimensional Data." *Information Visualization* 4 (2): 96–113. 2005.
[21] B. Tang, S. Han, M. L. Yiu, R. Ding, and D. Zhang. "Extracting Top-K Insights from Multi-Dimensional Data." In *SIGMOD'17*. 2017.
[22] M. Vartak, S. Rahman, S. Madden, A. Parameswaran, and N. Polyzotis. "SeeDB: Efficient Data-Driven Visualization Recommendations to Support Visual Analytics." *Proc. VLDB Endow.* 8 (13): 2182–2193. Sep. 2015.
[23] A. Wasay, M. Athanassoulis, and S. Idreos. "Queriosity: Automated Data Exploration." In *Proc. IEEE Intl. Congress on Big Data*. 2015.
[24] K. Wongsuphasawat, D. Moritz, A. Anand, J. Mackinlay, B. Howe, and J. Heer. "Voyager: Exploratory Analysis via Faceted Browsing of Visualization Recommendations." *IEEE Trans. Visualization & Comp. Graphics (Proc. InfoVis)*. 2016.
[25] Y. Yan, L. J. Chen, and Z. Zhang. "Error-Bounded Sampling for Analytics on Big Sparse Data." In *VLDB'14*. 2014.