Extracting Top-K Insights from Multi-dimensional Data

Bo Tang[†*] Shi Han[‡] Man Lung Yiu[†] Rui Ding[‡] Dongmei Zhang[‡]
[†] The Hong Kong Polytechnic University, Kowloon, Hong Kong
[‡] Microsoft Research, Beijing, China
[†] {csbtang,csmlyiu}@comp.polyu.edu.hk [‡] {shihan,juding,dongmeiz}@microsoft.com

ABSTRACT

OLAP tools have been extensively used by enterprises to make better and faster decisions. Nevertheless, they require users to specify group-by attributes and know precisely what they are looking for. This paper takes the first attempt towards automatically extracting top-k *insights* from multi-dimensional data. This is useful not only for non-expert users, but also reduces the manual effort of data analysts. In particular, we propose the concept of *insight* which captures interesting observation derived from aggregation results in multiple steps (e.g., rank by a dimension, compute the percentage of measure by a dimension). An example insight is: "Brand B's rank (across brands) falls along the year, in terms of the increase in sales". Our problem is to compute the top-k insights by a score function. It poses challenges on (i) the effectiveness of the result and (ii) the efficiency of computation. We propose a meaningful scoring function for insights to address (i). Then, we contribute a computation framework for top-k insights, together with a suite of optimization techniques (i.e., pruning, ordering, specialized cube, and computation sharing) to address (ii). Our experimental study on both real data and synthetic data verifies the effectiveness and efficiency of our proposed solution.

1. INTRODUCTION

OLAP tools facilitate enterprise knowledge workers (e.g., executives, managers, and analysts) on decision making in business intelligence applications. Their interfaces allow users to navigate the aggregation result by operations (e.g., slicing, dicing, drill up/down). Nevertheless, these tools still require users to specify the dimensions (i.e., group-by attributes) in OLAP queries. This analysis process requires tedious hit-and-trial from the user, on manually posing queries, analyzing results and deciding what is interesting [31]. To alleviate this issue, semi-automatic methods [25, 28] can be used to detect local anomalies or interesting views in an OLAP cube; however, these methods still require the user to specify a target (e.g., an OLAP cell or a dimension). Recent vision papers [10, 31] and the industry [9] have called for automatic techniques to obtain insights from data, helping users when they are not clear on what they are looking for [27, 15].

Taking a step further, in this paper, we take the first attempt to extract interesting insights from facts in a multi-dimensional dataset (e.g., sales data). We plan to: (i) formulate the concept of insight and propose a meaningful scoring function for it, and (ii) provide efficient solutions to compute top-k insights automatically.

Suppose that we have a car sales dataset with the schema (Year, Brand, Category, Sales) [1]. OLAP tools support aggregation on data (e.g., the SUM of Sales by Year and Brand, cf. Step 1 in Figure 1). However, aggregation alone does not reveal much information (e.g., clear trend), as illustrated in Figure 1(a). In this work, we consider *insight* as an interesting observation derived from aggregation in multiple steps. In following examples, we demonstrate how insights provide valuable information by performing analysis operations (e.g., rank, difference) over aggregation.

Example 1. (Yearly increased sales): We may compare the growth of different brands by the yearly increased sales (cf. Step 2 in Figure 1). In Figure 1(b), we observe that the yearly increased sales of brand H is rising with years. The aggregation result of the sales of brand H (cf. Figure 1(a)) first drops and then rises, whose trend is not intuitive to understand. In contrast, our insight (cf. Figure 1(b)) provides a clear rising trend. This insight can be effectively used to reveal the potential market and seek profits.

Example 2. (The rank of yearly increased sales among brands): Regarding the brand B, we have not found any "interesting" information about it from raw aggregation (cf. Step 1) and yearly increased sales (cf. Step 2). Nevertheless, we can obtain insight by applying analysis operations on the yearly increased sales. For example, if we rank the yearly increased sales across brands (cf. Step 3), we derive: "the rank of brand B's (across brands) yearly increased sales falls with years", as shown in Figure 1(c). Such insight implies that the competitiveness of brand B decreases with years.

The above examples illustrate typical insights extracted from multi-dimensional data. These insights have two usages in business intelligence applications. First, they provide informative summaries of the data to non-expert users, who do not know exactly what they are looking for [27, 15]. For example, a car seller is looking for interesting patterns in car sales, without knowing the exact patterns in advance. Insights can provide quick and informative summaries of interesting observations in the data and reveal interesting patterns (e.g., the sales of SUV is the best across Category in brand B). Second, data analysts may customize insights by their needs to guide directions for data exploration. For

[*]This work was conducted when the first author was a research intern at Microsof Research.

SIGMOD'17, May 14-19, 2017, Chicago, Illinois, USA

DOI: http://dx.doi.org/10.1145/3035918.3035922

[1]Due to confidential reasons, we anonymized some attributes of the car sales data set in our running examples and experiments.

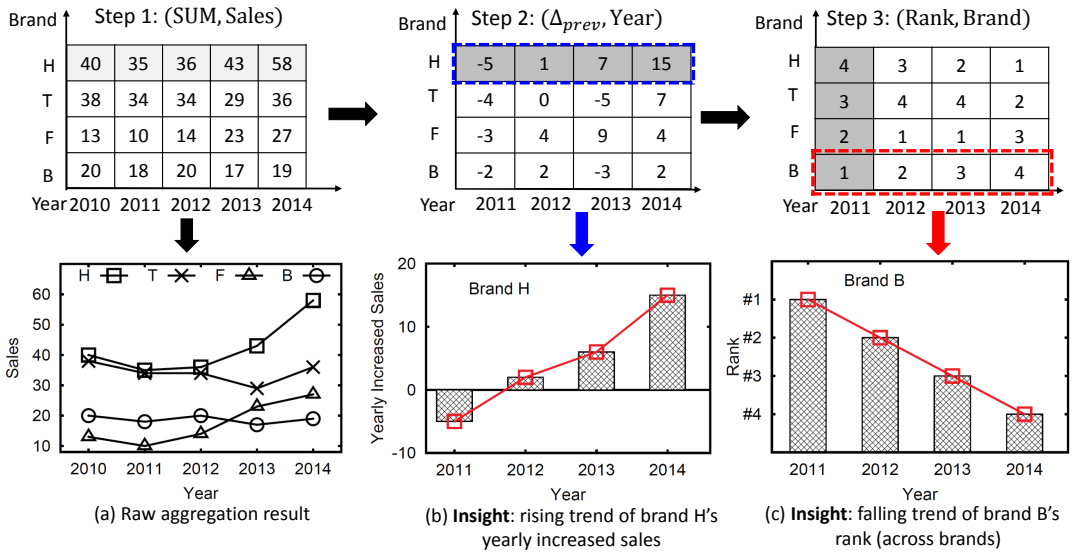


Figure 1: Example of insights

example, a data analyst focusing on brand B may wish to find insights related to B, e.g., the rank of brand B's yearly increased sales falls with year (as insight in Figure 1(c)). Then, he will continue to investigate the reason why such a trend happens.

Besides the above insights, the dataset may contain many other insights for other combinations of group-by attributes and analysis operations. Even with the aid of OLAP tools, it is tedious for data analysts to enumerate all possible insights and evaluate their "importance" manually. Motivated by this, we propose the **top-k insight problem**, which returns the best k insights with respect to an importance measure. Our problem does not require users to specify any input (e.g., group-by attributes, analysis operations in each step). Although there exist several works on extending the capability of OLAP [25, 34, 33, 1], they do not support automatic extraction of our proposed insights. We leave the discussion of these works in the related work section (cf. Table 6, Section 10).

Our problem poses challenges on (i) the effectiveness of the result and (ii) the efficiency of computation. Regarding the effectiveness aspect, existing OLAP tools lack the means to evaluate the importance of insights. We need a meaningful scoring function that supports generality and comparability among different insights. On the other hand, the efficiency aspect involves the following three technical challenges.

C1) Huge search space: The search space is exponential to the number of dimension attributes d and the depth of insight τ (i.e., the number of steps in an insight). Also, the search space is polynomial to the domain sizes of dimensions and the combinations of analysis operations. We leave the analysis in Section 2.3.

C2) Expensive insight computation: The evaluation of an insight requires applying multiple analysis operations after aggregation (e.g., Steps 2 and 3 in the above example). Each analysis operation may require accessing multiple values in aggregation results.

C3) Non-monotonicity of insight score: As we will explain shortly, the insight score function is not monotonic. For example, there is no insight in brand B with yearly increased sales (cf. Step 2), but there is an insight in the rank of brand B (across brands) with regard to the yearly increased sales, as shown in Figure 1(c). Moreover, there is no insight in the rank of brand T (as Figure 2 (a)) in above example; but there is an insight in its children spaces (e.g., $\langle$*, T, SUV$\rangle$ in Figure 2(b), where SUV is a value for the Category dimension). Such non-monotonicity prevents us from utilizing existing aggregation computation methods [13, 35], which require the function to be monotonic.

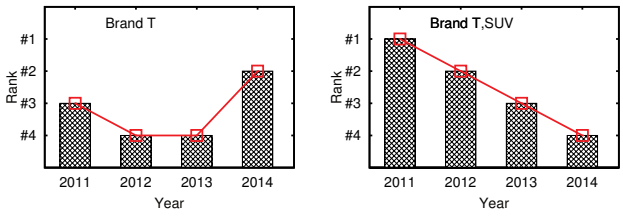


Figure 2: Examples of non-monotonicity

While the problem of automatic insight extraction is challenging, we develop efficient evaluation techniques that render insight extraction feasible for large-scale applications (i.e., the execution time of our solution is sub-linear with data size). Specifically, we devise a suite of optimization techniques to reduce the computation cost. The contributions of this paper are:

1. We formulate the top-k insight problem (Section 2) and propose a meaningful scoring function for insights (Section 3);
2. We propose the architecture of our top-k insight extraction system (Section 4) and the computation framework (Section 5);
3. We design a suite of optimization techniques — pruning, ordering, specialized cube (Section 6) and computation sharing (Section 7) to speedup insight extraction;
4. We verify the effectiveness of top-k insights on three real datasets by case study and user study (Section 8), and demonstrate the efficiency of our proposal (Section 9).

The remainder of this paper is organized as follows. Section 2 formulates our problem and Section 3 provides a meaningful score for insights. Section 4 describes the architecture of our proposed system and discusses its extensibility. Sections 5, 6 and 7 present the computation framework and a suite of performance optimization techniques. Sections 8 and 9 demonstrate the effectiveness

and efficiency of our proposal, respectively. Section 10 discusses related work, followed by the conclusion in Section 11.

2. PROBLEM STATEMENT

In this section, we provide formal definitions of the multi-dimensional data model, composite extractors, and the score function of insights. Finally, we formulate our insight extraction problem and analyze its search space.

2.1 Data Model and Subspace

We are given a multi-dimensional dataset $\mathbb{R}(\mathcal{D}, \mathcal{M})$ where $\mathcal{D} = \langle D_1, ..., D_d \rangle$ is the list of dimension attributes, and $\mathcal{M}$ is the measure attribute. Let $dom(D_i)$ denote the domain of attribute D_i. We assume that each D_i satisfies $|dom(D_i)| > 1$[2].

Consider the entire OLAP cube defined on the dataset $\mathcal{D}$. Given a cube cell, we can describe its attributes' values by a subspace S and its aggregate value by a measure $S.\mathcal{M}$, as defined below.

DEFINITION 1 (SUBSPACE). *A* **subspace** *is defined as an array $S = \langle S[1], ..., S[d] \rangle$, where $S[i]$ can take a value in $dom(D_i)$ or the value $*$ (i.e., 'all' values). Given the dataset $\mathcal{D}$, the aggregate measure $S.\mathcal{M}$ is defined as the aggregation of tuples in $\mathcal{D}$ that match with S.*

For simplicity, we also call $S.\mathcal{M}$ as the measure of subspace S. It is convenient to analyze the change of cube cells (i.e., subspaces) by varying a single dimension. Therefore, we define a *sibling group* to cover subspaces that differ on a single dimension only.

DEFINITION 2 (SIBLING GROUP). *Given a subspace S and a dimension D_i, a* **sibling group** *is defined as $\mathsf{SG}(S, D_i) = \{S' : S'[i] \neq * \ \wedge \ \forall j \neq i, S'[j] = S[j]\}$, i.e., a set of subspaces that instantiate the values in $dom(D_i)$. We also call D_i as a dividing dimension for $\mathsf{SG}(S, D_i)$.*

Example: Table 1 illustrates an example dataset (car sales). It contains two dimensions (Year, Brand) and a measure (Sales). By fixing the year (to 2010) and varying the brand, we can compare the sales of different brands in the same year: ⟨2010,F⟩, ⟨2010,B⟩, ⟨2010,H⟩, ⟨2010,T⟩. These four subspaces belong to the sibling group SG(⟨2010,*⟩,Brand).

Tuples	Tuples	Tuples	Tuples
2010, F, 13	2010, B, 20	2010, H, 40	2010, T, 38
2011, F, 10	2011, B, 18	2011, H, 35	2011, T, 34
2012, F, 14	2012, B, 20	2012, H, 36	2012, T, 34
2013, F, 23	2013, B, 17	2013, H, 43	2013, T, 29
2014, F, 27	2014, B, 19	2014, H, 58	2014, T, 36

Table 1: Car sales dataset (Year, Brand, Sales)

2.2 Composite Extractor

We shall conduct analysis operations on a sibling group in order to derive an observation. First, we introduce an extractor as a basic analysis operation on a sibling group.

DEFINITION 3 (EXTRACTOR). *An* extractor *ξ takes a sibling group $\mathsf{SG}(S, D_x)$ as input, and computes for each subspace $S_c \in \mathsf{SG}(S, D_x)$ a derived measure $S_c.\mathcal{M}'$ based on (i) $S_c.\mathcal{M}$ and (ii) $\{(S_{c'}, S_{c'}.\mathcal{M}) : S_{c'} \in \mathsf{SG}(S, D_x)\}$, i.e., the measures of all subspaces in $\mathsf{SG}(S, D_x)$.*

[2]We discard any attribute D_i with $|dom(D_i)| = 1$ because such an attribute is not meaningful for analysis.

Inspired by Sarawagi et al. [23, 26] and market share analysis[3], we propose four instances of extractors (i.e., Rank, %, Δ_{avg}, Δ_{prev}) and describe their semantics in Table 2. The extractor Δ_{prev} imposes an requirement that D_x must be an ordinal attribute, because $prev_{D_x}(S_c)$ refers to the previous subspace of S_c along D_x. The other extractors are applicable to any type of attribute. In addition to these extractors, we also allow data analysts to define their own extractors for their applications.

Extractor ξ	Derived measure $S_c.\mathcal{M}'$ for S_c	Requirement
Rank	the rank of $S_c.\mathcal{M}$ in $\mathsf{SG}(S, D_x)$	nil
%	% of $S_c.\mathcal{M}$ over the SUM of measures in $\mathsf{SG}(S, D_x)$	nil
Δ_{avg}	$S_c.\mathcal{M}-$ the AVERAGE of measures in $\mathsf{SG}(S, D_x)$	nil
Δ_{prev}	$S_c.\mathcal{M} - prev_{D_x}(S_c).\mathcal{M}$	D_x is ordinal

Table 2: List of extractors, with the input $\mathsf{SG}(S, D_x)$

Example: We illustrate the output of the above extractors in Table 3. Consider the sibling group $\mathsf{SG}(S$,Year), where S=⟨*,F⟩. The extractor Rank computes the rank of each S_c among all years. the extractor % calculates the percentage of each S_c among all years. The extractor Δ_{avg} returns the difference of each S_c from the average measure. The extractor Δ_{prev} obtains the difference of each S_c from its previous subspace along Year.

Sib. group $\mathsf{SG}(S, D_x)$	Measure $S_c.\mathcal{M}$	Derived measure $S_c.\mathcal{M}'$ for			
		Rank	%	Δ_{avg}	Δ_{prev}
⟨2010,F⟩	13	4	15%	-4.4	
⟨2011,F⟩	10	5	11%	-7.4	-3
⟨2012,F⟩	14	3	16%	-3.4	4
⟨2013,F⟩	23	2	27%	5.6	9
⟨2014,F⟩	27	1	31%	9.6	4

Table 3: Examples for extractors

We then introduce a *composite extractor* $\mathcal{C}_e$ to capture a multi-step analysis operator on a sibling group.

DEFINITION 4 (COMPOSITE EXTRACTOR). *Given a depth parameter τ, a* **composite extractor** *$\mathcal{C}_e$ is defined as a length-τ array of pairs $(\mathcal{C}_e[i].\xi, \mathcal{C}_e[i].D_x)$ such that it satisfies: (i) $\mathcal{C}_e[1].\xi$ is an aggregate function and $\mathcal{C}_e[1].D_x$ is the measure attribute $\mathcal{M}$, (ii) each $\mathcal{C}_e[i]$ $(i > 1)$ is an extractor, and (iii) adjacent extractors are compatible [4] (see Appendix A).*

We assume that the aggregate function is SUM on the measure attribute $\mathcal{M}$ in the dataset. Nevertheless, we will consider other aggregate functions and multiple measure attributes in Section 4.2. The depth parameter τ captures the complexity of a composite extractor. When $\tau = 1$, a composite extractor is the same as the aggregate function. We recommend setting τ to 2 or 3, which are analogous to first-order and second-order derivatives in Mathematics, respectively. For the examples in Figure 1, we can express steps 1–2 by a depth-2 composite extractor $\mathcal{C}_e$ = ⟨(SUM,Sales),(Δ_{prev},Year)⟩, and express steps 1–3 by a depth-3 $\mathcal{C}_e$ = ⟨(SUM,Sales),(Δ_{prev},Year),(Rank, Brand)⟩.

Next, we define the result set of applying a composite extractor $\mathcal{C}_e$ on a sibling group $\mathsf{SG}(S, D_a)$, as in Definition 5.

DEFINITION 5 (RESULT SET OF COMPOSITE EXTRACTOR). *A* **composite extractor** *$\mathcal{C}_e$ takes sibling group $\mathsf{SG}(S, D_a)$ as input, and computes the result set $\Phi = \{(S', S'.\mathcal{M}_\tau) : S' \in \mathsf{SG}(S, D_a)\}$, where the $S_c.\mathcal{M}_i$ denotes the level-i derived measure of a subspace S_c with respect $\mathcal{C}_e[i]$. The value of $S_c.\mathcal{M}_i$ is defined recursively as follows.*

[3]https://en.wikipedia.org/wiki/Market_share_analysis

[4]Two extractors $\mathcal{C}_e[i]$ and $\mathcal{C}_e[i+1]$ are compatible if the output set of $\mathcal{C}_e[i]$ can be used as the input of $\mathcal{C}_e[i+1]$.

At any level $i > 1$, we obtain each $S'.\mathcal{M}_i$ by applying the extractor $\mathcal{C}_e[i].\xi$ on the set $\{(S_c, S_c.\mathcal{M}_{i-1}) : S_c \in \mathsf{SG}(S', \mathcal{C}_e[i].D_x)\}$.

At level i=1, $S'.\mathcal{M}_1$ is the aggregate result on the measure $\mathcal{M}$.

Example: Figure 3 shows the result set Φ after applying the composite extractor $\mathcal{C}_e$ = ⟨(SUM,Sales),(%,Year)⟩ on the sibling group SG(⟨*,F⟩,Year). For example, at level 2, the derived measure of S_c=⟨2014, F⟩ is: $S_c.\mathcal{M}_2 = \frac{S_c.\mathcal{M}_1}{\sum_{S' \in \mathsf{SG}(\langle *, \mathrm{F}\rangle, \mathrm{Year})} S'.\mathcal{M}_1} = \frac{27}{13+10+14+23+27} = 31\%$, as illustrated in Figure 3, where $S'.\mathcal{M}_1$ is SUM(S'). We will propose an algorithm to compute the result set in Section 5.2.

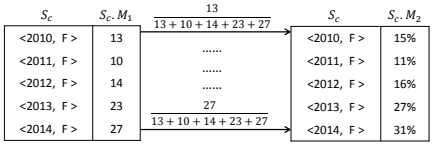


Figure 3: Example of composite extractor computation

In some cases, a composite extractor is not applicable to some sibling groups. For example, the composite extractor ⟨(SUM,Sales),(%,Year)⟩) cannot be applied to the sibling group SG(⟨*,*⟩,Brand) because the subspaces in the group do not have known values in the dimension 'Year'. We will discuss how to test the validity of a composite extractor $\mathcal{C}_e$ on a sibling group $\mathsf{SG}(S, D_i)$ in Appendix B.

2.3 Problem Definition

Intuitively, business analysts are interested in exceptional facts (e.g., significant differences within a sibling group) and unexpected trends (e.g., rapid rise during a time period). Let Φ be the result set after applying a composite extractor $\mathcal{C}_e$ on a sibling group $\mathsf{SG}(S, D_i)$. We propose to extract two representative types of "insights" from Φ.

1. **Point insight (outstanding):** Outstanding (No.1 / Last) means that a subspace is remarkably different from others in terms of $S_c.\mathcal{M}_\tau$.
2. **Shape insight (trend):** This insight is applicable when D_i is an ordinal dimension. A rising / falling trend means that $S_c.\mathcal{M}_\tau$ exhibits such a trend when D_i increases.

There are various other types of insights, e.g., those in the Microsoft Power BI product [3]. We will discuss the extensions of our solution for other types of insights in Section 4.2.

Formally, we denote a specific insight instance by $(\mathsf{SG}(S, D_i), \mathcal{C}_e, \mathsf{T})$ where T is an insight type. Our problem is to find the top-k insights according to a *score function* $\mathbb{S}(\mathsf{SG}(S, D_i), \mathcal{C}_e, \mathsf{T})$, which we will elaborate in Section 3.

PROBLEM 1 (INSIGHT PROBLEM). *Given a dataset $\mathcal{R}(\mathcal{D}, \mathcal{M})$ and composite extractor depth τ, find top-k insights $\{(\mathsf{SG}(S, D_i), \mathcal{C}_e, \mathsf{T})\}$ with the highest scores among all possible combinations of sibling groups, composite extractors, and insight types.*

Search Space Size: Before presenting our solutions, we first analyze the search space size of our problem, i.e., the number of possible insights $(\mathsf{SG}(S, D_i), \mathcal{C}_e, \mathsf{T})$, where $\mathsf{SG}(S, D_i)$ is a sibling group, $\mathcal{C}_e$ is a composite extractor, and T is an insight type. In our analysis, let $\mathfrak{D} = \max_{i=1}^{d} |dom(D_i)|$ be the maximum domain size, β be the number of extractor types, and $|\mathsf{T}|$ be the number of insight types. The search space of our solutions is as follows.

LEMMA 1 (SEARCH SPACE SIZE). *The number of possible insights is $O(|\mathsf{T}| \cdot d \cdot (\beta \cdot d)^{\tau-1} \cdot (\mathfrak{D}+1)^d)$.*

PROOF. First, the number of insight types is $|\mathsf{T}|$. For the number of sibling groups, the number of subspaces is $O((\mathfrak{D}+1)^d)$, and there are $O(d)$ choices for D_i.

An extractor (ξ, dim) has $O(\beta \cdot d)$ possible choices. Since a composite extractor contains $\tau - 1$ extractors, there are $O((\beta \cdot d)^{\tau-1})$ possible composite extractors.

By multiplying the above terms, we obtain the number of possible insights: $O(|\mathsf{T}| \cdot d \cdot (\beta \cdot d)^{\tau-1} \cdot (\mathfrak{D}+1)^d)$. □

The scope of this paper: The insight problem involves two evaluation metrics: (i) efficiency, and (ii) effectiveness. For efficiency, we propose a computation framework (in Section 5) with optimization techniques (in Sections 6 and 7) to extract top-k insights efficiently. For effectiveness, we present our methodology to measure the score of an insight (in Section 3), and then verify the effectiveness of top-k insights by case study and user study on real datasets (in Section 8).

3. MEANINGFUL INSIGHT SCORE

The insight score reflects the *interestingness* of an insight. In order to rank different insights, the insight score metric should exhibit: (i) generality (i.e., applicable to different types of insights), (ii) comparability (i.e, fair across different types of insights).

We first discuss existing works for evaluating the interestingness of information in the literature. In the problem context of [34], the score of a subspace S is defined as:

$$\mathbb{S}(q, S) = \mathsf{Rank}(q, S)^{-1} \cdot \mathsf{ObjCount}(S)$$

where q is a given query tuple, ObjCount(S) is the number of tuples in S, and Rank(q, S) is the percentile rank of q among tuples in S. Unfortunately, their score function cannot be readily applied to our insights because (i) our problem does not have any query tuple, (ii) the functions $\mathsf{Rank}(q, S)^{-1}$ and ObjCount(S) do not capture the characteristics of our insights (e.g., point and shape insights).

The concept of interestingness has also been studied in the context of OLAP cube analysis [25, 24, 32]. All of them define interestingness of a cell value (in the cube) by how surprising that value differs from the expectation. The expectation is often set by the system in [25, 32], whereas [24] allows a user to specify a list of "known cells" in order to set the expectation for other cells. Following the above works, [11] proposes measuring the interestingness of facets in textual documents by using p-value. Unfortunately, all these notions of interestingness are not applicable for our problem as they do not satisfy the properties (e.g., generality, comparability) of insight score.

3.1 Insight Score Function

We propose a more appropriate score function for an insight $(\mathsf{SG}(S, D_i), \mathcal{C}_e, \mathsf{T})$ as:

$$\mathbb{S}(\mathsf{SG}(S, D_i), \mathcal{C}_e, \mathsf{T}) = \mathsf{Imp}(\mathsf{SG}(S, D_i)) \cdot \mathsf{Sig}_{\mathsf{T}}(\Phi) \qquad (1)$$

where Imp measures the impact in the sibling group $\mathsf{SG}(S, D_i)$, $\mathsf{Sig}_{\mathsf{T}}$ measures the significance of type-T insight observed from Φ, and Φ is the result of $\mathcal{C}_e$ on $\mathsf{SG}(S, D_i)$.

The Impact Measure Imp: From a business viewpoint, the impact represents the market share of S. We employ

$$\mathsf{Imp}(\mathsf{SG}(S, D_i)) = \textstyle\sum_{S' \in \mathsf{SG}(S, D_i)} \mathsf{SUM}(S') / \mathsf{SUM}(\langle *, \cdots, * \rangle),$$

so that its value domain is normalized in the interval $[0, 1]$.

The Significance Measure $\mathsf{Sig}_{\mathsf{T}}$: It reveals the uncommonness of an observed insight in the result set Φ. The higher the score, the

more uncommon/unexpected that insight is. We intend to formulate Sig_T based on the p-value, which essentially measures how extreme an event is. It also allows fair comparisons among different types of insights when user properly set common cases (i.e., null hypotheses) for each insight type. In addition, the usage of p-value has been justified by the user-study in [11]. Therefore, we use p-value to measure Sig_T.

3.2 The Sig of insight

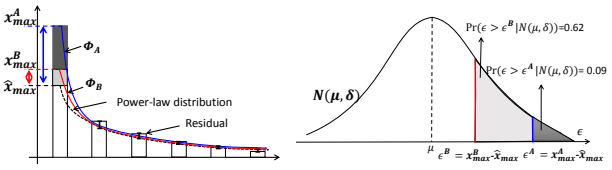
In statistics, the p-value is defined as "the probability of obtaining a result equal to or more extreme than what is actually observed, with the given null hypothesis being true" [17]. To achieve generality, we measure the p-value of different types of insights by using different kinds of null hypotheses. We suppose that the null hypotheses for different type of insights are common in real world. We then propose significance functions for point insight and shape insight. The detailed methodologies are available in Appendix C. Please note that, alternatively, users may customize null hypotheses for personalized analysis and employ their own significance functions. We will discuss them in the extensions in Section 4.2.

Measuring Sig of Point Insight. Let $\mathcal{X} = \{x_1, x_2, ..., x_n\}$ be the set of numeric values in the result Φ. In the business domain [5], the sale of products often follows a power-law distribution[5]. Thus, we set the null hypothesis of point insight as:

H_0 : $\mathcal{X}$ follows a power-law distribution with Gaussian noises.

The significance should reveal how surprisingly the maximum value differs from the rest of values in Φ with H_0 is true[6].

First, we sort $\mathcal{X}$ in the descending order and obtain the maximum value x_{max}. Then, we fit the values in $\mathcal{X} \setminus \{x_{max}\}$ to a power-law distribution if it is good fit, like in Figure 4(a), where the prediction errors of $x_i \in \mathcal{X} \setminus \{x_{max}\}$ (i.e., subtracting observed value x_i from estimated value $\hat{x}_i$, also called residuals) approximately follow Gaussian distribution $N(\mu, \delta)$. Next, we determine how surprising it is that x_{max} observed against the hypothesis H_0 is true by (i) deriving x_{max}'s prediction error by $\epsilon_{max} = \hat{x}_{max} - x_{max}$, (ii) calculating the p-value $p = Pr(\epsilon > \epsilon_{max} | N(\mu, \delta))$, as we depicted in Figure 4(b). Finally, we obtain the significance as $\mathsf{Sig}_\mathsf{T}(\Phi) = 1 - p$.



(a) Power-law distribution (b) p-value

Figure 4: The significance of point insight

We illustrate an example in Figure 4. For the result set Φ_A, the prediction error x^A_{max} is large, so we obtain $p = Pr(\epsilon > \epsilon^A_{max} | N(\mu, \delta)) = 0.09$ and derive the significance as $\mathsf{Sig}_\mathsf{T}(\Phi_A) = 1 - p = 0.91$. For the result set Φ_B, the prediction error x^B_{max} is small, so we derive the significance as $\mathsf{Sig}_\mathsf{T}(\Phi_A) = 1 - p = 1 - 0.62 = 0.38$. Thus, Φ_A is more significant than Φ_B.

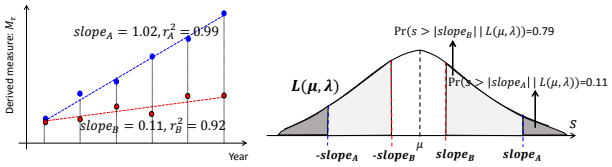
Measuring Sig of Shape Insight. Let $\mathcal{X} = \langle x_1, x_2, ..., x_n \rangle$ be the time series of values in the result Φ. It is common that the trend is neither rising nor falling. Therefore, we set the null hypothesis as:

H_0 : $\mathcal{X}$ forms a shape with slope ≈ 0.

[5] https://en.wikipedia.org/wiki/Long_tail

[6] We omit the minimum value discussion, as it is similar with the maximum case.

In business intelligence applications, data analysts are attracted to a clear rising/falling trend, whose slope is very different from 0. Thus, the p-value should measure how surprisingly the slope differs from 0.



(a) Slope of trends (b) p-value

Figure 5: The significance of shape insight

First, we fit $\mathcal{X}$ to a line by linear regression analysis (see Figure 5(a)), and then compute its slope $slope$ and the goodness-of-fit[7] value r^2. In this paper, we use logistic distribution $L(\mu, \lambda)$ [6], where μ, λ are constant parameters, to model the distributions of slopes. In Figure 5(b), the p-value is the probability of the slope values equal to or larger than the observed slope of the rising trend[8]. Specifically, we compute the p-value as $p = Pr(s > |slope| \mid L(\mu, \lambda))$. Finally, we define the significance as $\mathsf{Sig}_\mathsf{T}(\Phi) = r^2 \cdot (1 - p)$, where the goodness-of-fit value r^2 is used as a weight.

We illustrate an example in Figure 5. Consider the shapes of blue dots and red dots in Figure 5(a). After fitting blue dots to a line, we obtain the slope $slope_A = 1.02$ and the goodness-of-fit value $r^2_A = 0.99$. Similarly, after fitting red dots, we obtain $slope_B = 0.11$ and $r^2_B = 0.92$. As illustrated in Figure 5(b), we then compute: $p_A = 0.11$ and $p_B = 0.79$. In this example, since $slope_A > slope_B$, the significance of A (i.e., $0.88 = 0.99 * (1 - 0.11)$) is larger than that of B (i.e., 0.19).

4. SYSTEM ARCHITECTURE

We first describe the architecture of our top-k insight extraction system, then discuss the extensibility of our system.

4.1 Architecture Overview

Figure 6 depicts the architecture of our system, which consists of three layers.

1. The *system configuration* layer (at the bottom) allows a user to configure system settings, e.g., specify a new insight type, or customize the null hypothesis based on the user's belief. We will elaborate on the details of this layer in Section 4.2.

2. The *insight extraction* layer (in the middle) is the core component of our system. First, it enumerates every possible pair $(\mathsf{SG}(S, D_i), \mathcal{C}_e)$ of sibling group and composite extractor. Then, it feeds each pair $(\mathsf{SG}(S, D_i), \mathcal{C}_e)$ into the computation engine and invokes the insight engine to compute the score. During this process, the layer maintains the top-k insights list.

3. The *user interface* layer (at the top) is front-end of our system. It presents and visualizes the top-k insights to the users.

[7] https://en.wikipedia.org/wiki/Coefficient_of_determination

[8] We omit the falling trend discussion, as it is similar with rising trend.

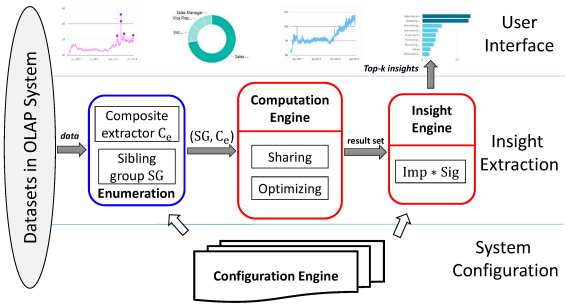


Figure 6: Top-k insight extraction system architecture

4.2 Extensibility

We have suggested some type(s) of aggregate functions, extractors, dataset, insights, and their score functions so far. Nevertheless, our system is extensible as follows:

Aggregate function and extractors: A composite extractor must take an aggregate function as the level-1 extractor, and then take any other extractor at higher levels. First, we support many typical aggregate functions in OLAP, e.g., SUM, COUNT, AVERAGE, MAX, MIN. For example, we will consider the aggregate function COUNT, in the user study in Section 8. Second, we also allow the data analyst to define his own extractor (e.g., difference from Rank-1). Regarding the validity of insight score, we only need to slightly revise the composite extractor adjunct taxonomy (cf. Table 7) and Imp function to ensure the validation of score computation.

Dataset: Since our system is built on top of an OLAP system, it can handle any kind of dataset in the OLAP system. For a dataset with multiple measure attributes, a user can either choose one measure attribute, or specify a derived measure as a weighted sum of other measure attributes [16] in system configuration layer.

Customization of insights: Our system supports other insight types, e.g., the correlation between two trends, and the seasonality of a trend [3]. We will elaborate the details in Appendix D.

Our score functions, e.g., significance functions, are also customizable. Recall in Section 3.2 that the significance of an insight type is defined based on p-value, which essentially measures how extreme an event is against a "common observation" in real world. Data analysts may customize their "common observations" by their domain knowledge.

Customization of the search space: Expert users may have some ideas of what they are looking for. As such, we enable expert users to declare constraints and limit the search space. For example, an expert user may only consider sibling groups related to brand B in the car sales dataset.

5. INSIGHT EXTRACTION

We present a computation framework for the *insight extraction layer*.

5.1 Computation Framework

Algorithm 1 is the pseudo-code of our computation framework for the *insight extraction layer* in the system architecture. It employs a heap $\mathcal{H}$ to keep the top-k insights found so far. The algorithm needs to generate all possible instances of composite extractor $\mathcal{C}_e$ and sibling group $\mathsf{SG}(S, D_i)$. It then computes the insight from every $(\mathsf{SG}(S, D_i), \mathcal{C}_e, \mathsf{T})$, and updates $\mathcal{H}$ upon finding a better insight.

Generally, the number of sibling groups is much larger than the

Algorithm 1 Insights (dataset $\mathcal{R}(\mathcal{D}, \mathcal{M})$, depth τ, result size k)

```
1: initialize a min-heap H ← ∅                                  ▷ store top-k insights
2: let ub_k be the k-th largest score in H
3: O ← enumerate all possible C_e with depth τ                  ▷ enumerate C_e
4: for each C_e in O do
5:     for i := 1 to d do                                       ▷ enumerate SG
6:         initialize subspace S ← ⟨*, *, ··· , *⟩
7:         EnumerateInsight( S, D_i, C_e )
8: return H

   Function: EnumerateInsight ( S, D_i, C_e ):
9: if isValid ( SG(S, D_i), C_e ) then                          ▷ Phase I
10:    Φ ← ExtractΦ( SG(S, D_i), C_e )        ▷ Alg. 2: computation engine
11:    for each insight type T do
12:        𝕊 ← Imp(SG(S, D_i)) · Sig_T(Φ)      ▷ Sec. 3: insight engine
13:        if 𝕊 > ub_k then
14:            update H, ub_k by (SG(S, D_i), C_e, T)
15: for each value v ∈ dom(D_i) do                              ▷ Phase II
16:    S' ← S, S'[D_i] ← v
17:    for each j with S'[D_j] = * do                           ▷ enumerate SG
18:        EnumerateInsight( S', D_j, C_e )
```

number of composite extractors. To keep the memory consumption manageable, we adopt the divide-and-conquer approach to generate sibling groups. Specifically, we implement this with a recursive function (Lines 9–18), which consists of two phases.

In Phase I (Lines 9–14), we first check whether the pair $(\mathsf{SG}(S, D_i), \mathcal{C}_e)$ is valid. If yes, then we compute the result Φ of the pair by Algorithm 2 which will be elaborated in Section 5.2. Next, we compute the score for each insight type and update $\mathcal{H}$ upon finding a better insight.

In Phase II (Lines 15–18), we create a child subspace S' from S by instantiating its value on dimension D_i. For each S', we pick a dimension D_j where $S'[D_j] = *$, and then invoke a recursive call on the sibling group $\mathsf{SG}(S', D_j)$

Example: Given the dataset in Table 1, we illustrate the obtained insights in Table 4. For ease of illustration, we only consider point insights and a fixed composite extractor $\mathcal{C}_e = \langle$(SUM,Sales),(Δ_{prev},Year)$\rangle$. In Table 4, each row shows a sibling group $\mathsf{SG}(S, D_i)$ and its insight score $\mathbb{S}$, i.e., the product of impact Imp and significance Sig. Due to page limits, we do not show the steps for computing Imp and Sig. When $k = 1$, the top-1 insight corresponds to the first row in Table 4.

$\mathsf{SG}(S, D_i)$	$\mathbb{S}$ = Imp · Sig	Point insight: outstanding No.1
SG(⟨*,*⟩, Year)	0.61 = 1.00 · 0.61	⟨2014⟩
SG(⟨*,H⟩, Year)	0.16 = 0.38 · 0.42	⟨2014, H⟩
SG(⟨*,T⟩, Year)	0.14 = 0.30 · 0.45	⟨2014, T⟩
SG(⟨*,F⟩, Year)	0.02 = 0.15 · 0.10	⟨2013, F⟩
SG(⟨*,B⟩, Year)	0.01 = 0.17 · 0.03	⟨2012, B⟩
SG(⟨2014,*⟩, Brand)	0.07 = 0.25 · 0.27	⟨2014, H⟩
SG(⟨2012,*⟩, Brand)	0.03 = 0.18 · 0.14	⟨2012, H⟩
SG(⟨2013,*⟩, Brand)	0.02 = 0.20 · 0.12	⟨2013, H⟩
SG(⟨2011,*⟩, Brand)	0.01 = 0.17 · 0.04	⟨2011, F⟩

Table 4: Insight candidates for $\mathcal{C}_e = \langle$(SUM,Sales),(Δ_{prev},Year)$\rangle$

5.2 Computation Engine

We introduce Algorithm 2 to apply a composite extractor $\mathcal{C}_e$ on a sibling group $\mathsf{SG}(S, D_i)$ and then compute a corresponding result set Φ. It enumerates each subspace $S' \in \mathsf{SG}(S, D_i)$ (Line 3) and computes the derived measure of S' with respect to $\mathcal{C}_e$ (Line 4). Finally, it inserts each S' with its derived measure into Φ and returns it to the caller.

Conceptually, the computation of the derived measure (at Line

Algorithm 2 ExtractΦ($\mathsf{SG}(S, D_i), \mathcal{C}_e$)

```
1: initialize a result set Φ ← ∅
2: for each value v in dom(D_i) do
3:     S' ← S, S'[D_i] ← v
4:     M' ← RecurExtract( S', τ, C_e )
5:     insert (S', M') into Φ
6: return Φ

   Function: RecurExtract( Subspace S', level, C_e ):
7: if level > 1 then
8:     initialize a result set Φ_level
9:     D_ξ ← C_e[level].D_x
10:    for each value v in D_ξ do
11:        S_v ← S', S_v[D_ξ] ← v
12:        M'_v ← RecurExtract( S_v, level − 1, C_e )
13:        insert (S_v, M'_v) into Φ_level
14:    M' ← derived S'.M' by applying C_e[level] on Φ_level   ▷ Def. 3
15: else
16:    M' ← SUM( S' )                                  ▷ SUM(S'): data cube
17: return M'
```

4) involves building trees in the top-down manner and running extractors on tree nodes in the bottom-up manner. This can be implemented by a recursive function 'RecurExtract'.

In this function, the parameter $level$ indicates the current level of the extractor in $\mathcal{C}_e$. The initial $level$ is τ, which corresponds to the highest level. Let D_ξ be the dimension used by the current extractor $\mathcal{C}_e[level]$ (Line 9). We examine each child subspace S_v (of S'), apply the $level-1$ extractor to it recursively, and insert S_v with its derived measure into a temporary result set Φ_{level} (Lines 10–13). Finally, we apply the current extractor on Φ_{level} to compute the derived measure at the current level.

When we reach the bottom level (i.e., $level = 1$), it suffices to compute the SUM of measures in the subspace S'. This can be obtained efficiently from a data cube.

Example: Consider the composite extractor $\mathcal{C}_e = \langle$(SUM,Sales),(Δ_{prev},Year)$\rangle$ with the sibling group SG($\langle$2013,*$\rangle$,Brand) in Table 4. We illustrate the recursive computation of derived measures in Figure 7. Each tree node represents a recursive call of 'RecurExtract', and it is associated with a subspace S_v and a derived measure M'_v. In the first phase, we build a tree in a top-down manner. The second phase begins when we reach the bottom level (i.e., $level = 1$). Next, we examine these tree nodes in a bottom-up manner and apply the corresponding extractor on each tree node to compute its derived measure. Then, we obtain the result set Φ={($\langle$2013,F$\rangle$,9), ($\langle$2013,B$\rangle$,-3), ($\langle$2013,H$\rangle$,7), ($\langle$2013,T$\rangle$,-5)}. Finally, we compute the Sig value of Φ (cf. Section 3.2) and the Imp value in order to obtain the insight score (i.e., 0.02).

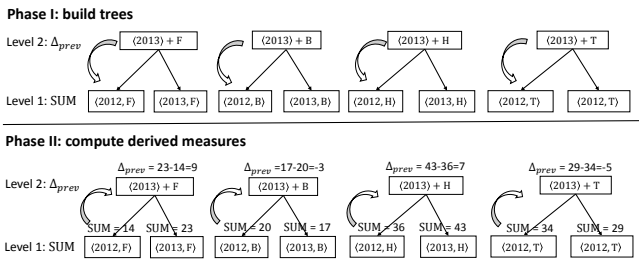
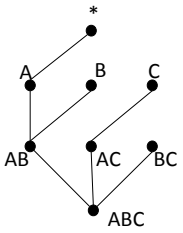


Figure 7: Running a composite extractor on a sibling group, $\mathcal{C}_e = \langle$(SUM,Sales),(Δ_{prev},Year)$\rangle$, SG($\langle$2013,*$\rangle$,Brand)

Data cube optimization: Our framework performs aggregation frequently, e.g., SUM(S) (Line 16 in Alg. 2), and Imp(SG(S, D_i)) = SUM$_{S' \in \mathsf{SG}(S,D_i)}(S')$/SUM($\langle *, *, \cdots, * \rangle$) (Line 12 in Alg. 1). We can construct a data cube and utilize it to reduce the aggregation cost.

A data cube [14, 8] is a collection of *cuboids*, where each cuboid stores the group-by aggregate result for a particular set of dimensions. Figure 8(a) illustrates a data cube built for a dataset with schema $(A, B, C, \mathcal{M})$. It contains eight cuboids. The content of cuboid $\langle A, B \rangle$ is shown in Figure 8(b).

To compute SUM(S) efficiently, we propose to store each cuboid as a hash table. Given a subspace S, we can lookup the corresponding entry in the cuboid and then retrieve SUM(S) in $O(1)$ time.



(a) a data cube

key: A, B	value: $\mathcal{M}$
a_1, b_1	2
a_1, b_2	3
...	...
a_2, b_1	7
a_2, b_2	9
...	...
a_3, b_2	8
a_3, b_3	1

(b) cuboid $\langle A, B \rangle$

Figure 8: Example of a data cube

5.3 Time Complexity Analysis

Since both Algorithms 1 and 2 spend the most of time on recursive calls, we focus on analyzing the number of recursive calls in these algorithms. We follow the notations in Section 2.3. In our analysis, $\mathfrak{D} = \max_{i=1}^{d} |dom(D_i)|$ denotes the maximum domain size, and β denotes the number of types of extractors.

For Algorithm 1: It invokes the recursive function 'EnumerateInsight' for all possible insights of $(\mathsf{SG}(S, D_i), \mathcal{C}_e, \mathsf{T})$. Each recursive call examines $|\mathsf{T}|$ types of insights. Combining this with the results in Section 2.3, the number of recursive calls to 'EnumerateInsight' is $O(|\mathsf{T}| \cdot d \cdot (\beta \cdot d)^{\tau-1} \cdot (\mathfrak{D}+1)^d)$.

For Algorithm 2: It examines each value of attribute D_i and thus calls the recursive function 'RecurExtract' for $\mathfrak{D}$ times at most.

Let j be the current level in the function 'RecurExtract'. When $j > 1$, the function 'RecurExtract' examines each value of attribute $\mathcal{C}_e[i].D_x$ and thus calls the function 'RecurExtract' for $\mathfrak{D}$ times at most.

In summary, the number of recursive calls to 'RecurExtract' is $O(\mathfrak{D} \cdot \prod_{j=2}^{\tau} dom_{max}) = O(\mathfrak{D}^\tau)$.

6. OPTIMIZATION TECHNIQUES

In this section, we propose optimization techniques to reduce the running time of our solution.

6.1 Pruning by Upper Bound Score

Consider the computation of insight score at Lines 10–12 in Algorithm 1. The term Imp(SG(S, D_i)) can be computed efficiently (cf. Section 5.2). However, it is expensive to compute Φ as it invokes Algorithm 2.

To reduce the cost, we propose an upper bound score

$$\mathbb{S}^{UB}(\mathsf{SG}(S, D_i), \mathcal{C}_e, \mathsf{T}) = \mathsf{Imp}(S) \tag{2}$$

and show that it serves as an upper bound of the insight score (cf. Lemma 2).

LEMMA 2 (UPPER BOUND PROPERTY).

$$\mathbb{S}^{UB}(\mathsf{SG}(S, D_i), \mathcal{C}_e, \mathsf{T}) \geq \mathbb{S}(\mathsf{SG}(S, D_i), \mathcal{C}_e, \mathsf{T})$$

PROOF. By Def. 2, we have $\mathsf{Imp}(\mathsf{SG}(S, D_i)) = \mathsf{Imp}(S)$ with $S[i] = *$ (Line 17, Alg 1). Hence, $\mathbb{S}(\mathsf{SG}(S, D_i), \mathcal{C}_e, \mathsf{T}) = \mathsf{Imp}(S) \cdot \mathsf{Sig}_{\mathsf{T}}(\Phi)$. Since $\mathsf{Sig}_{\mathsf{T}}(\Phi) \leq 1$, we have: $\mathbb{S}(\mathsf{SG}(S, D_i), \mathcal{C}_e, \mathsf{T}) \leq \mathsf{Imp}(S) = \mathbb{S}^{UB}(\mathsf{SG}(S, D_i), \mathcal{C}_e, \mathsf{T})$. □

With this lemma, we can implement the following pruning rule before Line 10 in Algorithm 1. We compute $\mathbb{S}^{UB}(\mathsf{SG}(S, D_i), \mathcal{C}_e, \mathsf{T})$ and then compare it with ub_k (i.e., k-th insight score found so far). If $ub_k > \mathbb{S}^{UB}(\mathsf{SG}(S, D_i), \mathcal{C}_e, \mathsf{T})$, then we skip the execution of Lines 10–14.

6.2 Subspace Ordering

The effectiveness of the above pruning rule (cf. Section 6.1) depends on ub_k (i.e., k-th insight score found so far). To enable effective pruning, it is desirable to obtain a high ub_k as early as possible. Therefore, we propose techniques to reorder both outer and inner loops (Lines 15–18) in Algorithm 1.

Ordering of outer-loop (Lines 15–16): Observe that the upper bound score $\mathbb{S}^{UB}(\mathsf{SG}(S', D_i), \mathcal{C}_e, \mathsf{T}) = \mathsf{Imp}(S')$ depends on S' only. Thus, we propose to compute $\mathbb{S}^{UB}$ for each subspace S' at Line 16, and then examine those subspaces in descending order of $\mathbb{S}^{UB}$.

Ordering of inner-loop (Lines 17–18): An intuitive strategy is to order dimensions in ascending order of the domain size $|dom(D_j)|$. When $|dom(D_j)|$ is small, few subspaces will be generated and the average impact of each subspace is expected to be high. This would increase the possibility to obtain a high ub_k early.

6.3 Sibling Cube

Our framework incurs significant overhead on (i) hash table lookup operation per computing $\mathsf{SUM}(S)$ (cf. Section 5.2), and (ii) sorting operation in implementing subspace ordering (cf. Section 6.2).

In this section, we propose an *sibling cube* in order to reduce the number of lookup operations in hash tables. Furthermore, our sibling cube can avoid redundant sorting operations in our framework.

6.3.1 Sibling cube structure

Our *sibling cube* is designed in a fashion that suits better with the operations used in our framework. Specifically, our sibling cube is a collection of the following cuboids:

DEFINITION 6 (CUBOID IN SIBLING CUBE). *A cuboid is labeled by $\langle \mathcal{D}' \rangle \circ \underline{D_i}$, where $\mathcal{D}' \subset \mathcal{D}$ is a subset of dimensions and D_i is a dimension not in $\mathcal{D}'$.*

The cuboid contains a cell for a subspace S if $\forall j \in \mathcal{D}', S[j] \neq *$ and $\forall j \notin \mathcal{D}', S[j] = *$.

The cell for subspace S is an array of pairs $\langle (v_x, \mathcal{M}_x) : v_x \in dom(D_i) \rangle$ sorted in the descending order of $\mathcal{M}_x$. We require that $\mathcal{M}_x = \mathsf{SUM}(S')$, where S' is a child subspace of S with its dimension D_i set to v_x.

Following the example in Section 5.2, we consider a dataset with schema $(A, B, C, \mathcal{M})$. We compare a data cube with our sibling cube in Figure 9. A cuboid in a data cube contains many cells (see Figure 9(a)). On the other hand, a cuboid in an sibling cube contains fewer cells but each cell stores more information (see Figure 9(b)).

Compared to the data cube, the sibling cube occupies at most d times the space in the worst case. Nevertheless, the iceberg cube technique [8] can be adapted to shrink our cube size significantly. Specifically, we only store entries whose measures are above $minsup\%$ (e.g., 0.1%) of measure in the dataset. Our experimental study shows that our sibling cube is small enough to fit in main memory.

In the following discussion, we demonstrate the advantages of using the sibling cube over the data cube.

6.3.2 Reducing hash table lookup operations

Our algorithms in Section 5 execute this operation: "Given a sibling group $\mathsf{SG}(S, D_i)$, retrieve $\mathsf{SUM}(S')$ for each subspace $S' \in \mathsf{SG}(S, D_i)$."

For example, we take $\mathsf{SG}(\langle a_1, * \rangle, B)$ as the sibling group and assume $dom(B) = \{b_1, b_2, b_3\}$. When using a traditional data cube, we issue three lookup operations $\langle a_1, b_1 \rangle, \langle a_1, b_2 \rangle, \langle a_1, b_3 \rangle$ to the cuboid in Figure 9(a).

With our sibling cube, it suffices to issue one lookup operation (a_1) to the cuboid in Figure 9(b). Then, we can retrieve the list of entries for $\langle a_1, b_3 \rangle, \langle a_1, b_2 \rangle, \langle a_1, b_1 \rangle$ and process the list sequentially.

key: A, B	value: $\mathcal{M}$
a_1, b_1	2
a_1, b_2	3
...	...
a_2, b_1	7
a_2, b_2	9
...	...
a_3, b_2	8
a_3, b_3	1

(a) data cube: cuboid $\langle A, B \rangle$

key: A	value: $B, \mathcal{M}$		
a_1	$b_3, 6$	$b_2, 3$	$b_1, 2$
a_2	$b_2, 9$	$b_1, 7$	$b_3, 5$
a_3	$b_2, 8$	$b_1, 4$	$b_3, 1$

(b) sibling cube: cuboid $\langle A \rangle \circ \underline{B}$

Figure 9: Data cube vs. sibling cube

In addition to reducing lookup operations, the sibling cube improves the data access locality (e.g., converting random accesses to sequential accesses) and benefits the performance of CPU cache [7].

6.3.3 Avoiding sorting operations in loop ordering

When we implement the outer loop ordering (see Section 6.2) at Lines 15–16 in Algorithm 1, we need to sort subspaces $S' \in \mathsf{SG}(S, D_i)$ in descending order of their upper bound scores (which can be derived from SUM values).

With our sibling cube, we can retrieve a sorted list directly and avoid sorting operations on-the-fly.

We extend the computation framework (Algorithm 1) with the above optimization techniques, and then present the optimized computation framework (Algorithm 3) in Appendix E.1.

7. COMPUTATION SHARING

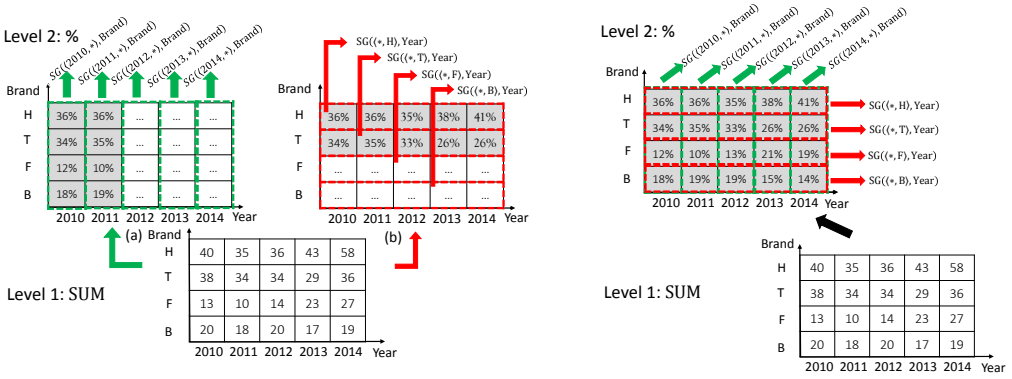
This section presents computation sharing techniques to further accelerate our solution.

7.1 Sharing within a Sibling Group

First, we identify sharing opportunities within a sibling group in an example. Then, we devise the condition for sharing.

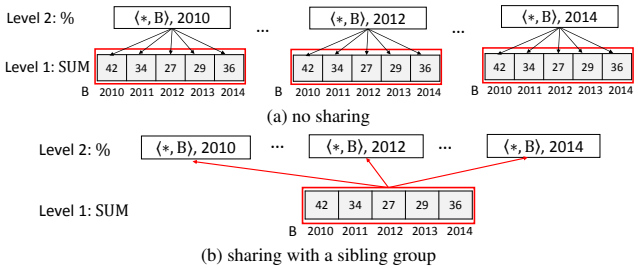
As an example, suppose that we apply the composite extractor $\mathcal{C}_e$ = ⟨(SUM,Sales),(%,Year)⟩ on the sibling group SG(⟨∗,B⟩,Year). Figure 11(a) illustrates the computation process of Algorithm 2 on this example. Observe that these trees have the same content at level 1, as highlighted by red rectangles. In order to reduce computation cost, we propose to identify the shared content and compute it only once, as shown in Figure 11(b).

We discover that significant computation can be saved when certain condition is satisfied. Specifically, we prove in Lemma 3 that,



(a) no sharing (b) sharing across sibling groups

Figure 10: Running a composite extractor on multiple sibling groups, $\mathcal{C}_e$=⟨(SUM,Sales),(%,Brand)⟩



(a) no sharing

(b) sharing with a sibling group

Figure 11: Running a composite extractor on a sibling group, $\mathcal{C}_e$=⟨(SUM,Sales),(%,Year)⟩,SG(⟨∗, B ⟩,Year)

if a sibling group $\mathsf{SG}(S, D_i)$ and the last extractor of $\mathcal{C}_e$ have the same dimension (i.e., $\mathcal{C}_e[\tau].D_x = D_i$), then we can share the intermediate result at level $\tau - 1$.

LEMMA 3 (SHARING WITHIN A SIBLING GROUP). *Given a composite extractor $\mathcal{C}_e$ and a sibling group $\mathsf{SG}(S, D_i)$, if $\mathcal{C}_e[\tau].D_x = D_i$, then all subspaces of $\mathsf{SG}(S, D_i)$ share the same intermediate result at level $\tau - 1$.*

PROOF. Let $S' \in \mathsf{SG}(S, D_i)$ be a subspace. Since S' and S differ on dimension D_i only, we have: $\mathsf{SG}(S', D_i) = \mathsf{SG}(S, D_i)$ ——(★).

According to Definition 5, we derive $S'.\mathcal{M}_\tau$ from the set $\Phi' = \{(S_c, S_c.\mathcal{M}_{\tau-1}) : S_c \in \mathsf{SG}(S', \mathcal{C}_e[\tau].D_x)\}$.

By combining (★) with the given condition $\mathcal{C}_e[\tau].D_x = D_i$, we derive: $\mathsf{SG}(S', \mathcal{C}_e[\tau].D_x) = \mathsf{SG}(S, D_i)$. Therefore, Φ' is independent of S' and it can be used to derive $S''.\mathcal{M}_\tau$ for any $S'' \in \mathsf{SG}(S, D_i)$. □

We enhance the computation engine (Algorithm 2) with the above sharing idea, and then obtain the optimized version (Algorithm 4) in Appendix E.2.

7.2 Sharing across Sibling Groups

We proceed to investigate sharing opportunities across multiple sibling groups.

Consider our computation framework in Algorithm 1. After fixing the composite extractor $\mathcal{C}_e$ (at Line 4), we enumerate sibling groups and apply $\mathcal{C}_e$ on each of them. In this example, we assume $\mathcal{C}_e$=⟨(SUM,Sales),(%,Brand)⟩. Figure 10(a) illustrates the computation process when we apply $\mathcal{C}_e$ on multiple sibling groups: $\mathsf{SG}(\langle 2010, *\rangle$,Brand) $\cdots$ $\mathsf{SG}(\langle 2014, *\rangle$,Brand), then $\mathsf{SG}(\langle *$,H$\rangle$,Year) $\cdots$ $\mathsf{SG}(\langle *$,F$\rangle$,Year). Observe that, at level 2, the derived measures in green rectangles are the same as those in red rectangles. This happens because some subspace ($\langle 2010$,H$\rangle$:42%) appears in more than one sibling groups ($\mathsf{SG}(\langle 2010, *\rangle$,Brand) and $\mathsf{SG}(\langle *$,H$\rangle$,Year)).

We illustrate how this method works with the example in Figure 10(b). We employ a temporary hash table Ψ to store the derived measure $S'.M'$ for subspace S' that we have processed before (in other sibling groups). Initially, Ψ is empty. First, we examine $\mathsf{SG}(\langle 2010, *\rangle$, Brand), and process four subspaces $\langle 2010$, F$\rangle$, $\langle 2010$, B$\rangle$, $\langle 2010$, H$\rangle$, $\langle 2010$, T$\rangle$. Since Ψ is empty, we need to compute the derived measures for the above subspaces and then insert them into Ψ. Similarly, we populate Ψ when we examine $\mathsf{SG}(\langle 2011, *\rangle$, Brand), $\cdots$, $\mathsf{SG}(\langle 2014, *\rangle$, Brand). Finally, when we examine $\mathsf{SG}(\langle *$,H$\rangle$,Year), we can find its subspaces in Ψ and thus retrieve their derived measures from Ψ directly.

We then discuss how to incorporate the above techniques into our algorithms. First, we apply the above technique and obtain an efficient computation engine (Algorithm 5) in Appendix E.3. Second, by using all techniques in Sections 6 and 7, we have an efficient computation framework (Algorithm 6) for insight extraction layer in Appendix E.4.

8. EFFECTIVENESS STUDY

In this section, we evaluate the effectiveness of our top-k insight extraction system by (1) case study, (2) insight utility study, and (3) human effort study on real datasets.

8.1 Case Studies

We collect the following two real datasets (i.e, car sales and tablet sales), and then demonstrate the insights obtained from these datasets.

Car sales dataset[9]**:** The dataset contains 276 tuples. Each tuple (i.e., a car) has 4 dimensions and a measure *Sales*. The domain sizes of dimensions are: *Year* (5), *Brand* (8), *Category* (8) and *Model* (55).

Tablet sales dataset[10]**:** The dataset contains 20,685 tuples. Each tuple (i.e., a tablet) has 11 dimensions and a measure *Sales*. The domain sizes of dimensions are: *Year* (11), *CPU* (2), *OS* (7), *Connectivity* (5), *Price* (23), *Region* (9), *Country* (54), *Product* (2), *Resolution* (18), *Size* (9) and *Vendor* (157).

[9] http://www.goodcarbadcar.net/p/sales-stats.html

[10] This is a private real dataset collected from the industry.

Insight	score	SG(S,D_i)	composite extractor $\mathcal{C}_e$
Top-1	0.31	SG({SUV},Brand)	⟨(SUM,Sales),(Δ_{avg},Category)⟩
Point	When measuring the importance of SUV sales for a certain Brand, brand F is Outstanding No.1.		
Top-2	0.30	SG({SUV},Year)	⟨(SUM,Sales),(%,Category)⟩
Shape	There is a rising trend of SUV's market share.		

(a) car sales, top-2 insights with $\tau = 2$

Top-1	0.32	SG({SUV},Year)	⟨(SUM,Sales),(%,Year),(Δ_{avg},Category)⟩
Point	In 2014, SUV exhibits most advantage over other categories than ever.		
Top-2	0.25	SG({F},Year)	⟨(SUM,Sales),(%,Brand),(Δ_{avg}, Year)⟩
Shape	There is a falling trend of brand F's market share.		

(b) car sales, top-2 insights with $\tau = 3$

Top-1	0.96	SG({*},Year)	⟨(SUM,Sales),(Δ_{prev},Year)⟩
Shape	The yearly increase of tablet sales is slowing down.		
Top-2	0.64	SG({WiFi},Year)	⟨(SUM,Sales),(Δ_{prev},Year)⟩
Shape	The yearly increase of sales of WIFI tablets is slowing down.		

(c) tablet sales, top-2 insights with $\tau = 2$

Top-1	0.99	SG({ * },Year)	⟨(SUM,Sales),(Δ_{prev},Year),(Δ_{avg},Year)⟩
Point	2012/04-07's yearly increase of tablet sales is remarkably lower than ever.		
Top-2	0.96	SG({Tablet},Year)	⟨(SUM,Sales),(%,Year),(Rank,Year)⟩
Shape	There is a rising trend of Tablet (vs. eReader) sales.		

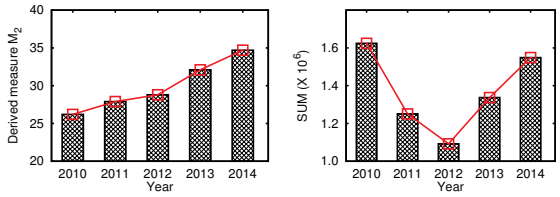
(d) tablet sales, top-2 insights with $\tau = 3$

Table 5: Case studies of insights on real datasets

Table 5 shows the top-2 insights on car sales and tablet sales, respectively, at τ=2 and τ=3. For convenience, we have omitted $*$ in sibling groups in Table 5. For example, SG({SUV},Year) is equivalent to SG(⟨*,*,SUV,*⟩, Year). We then elaborate some of these insights from Figure 12 to 15.

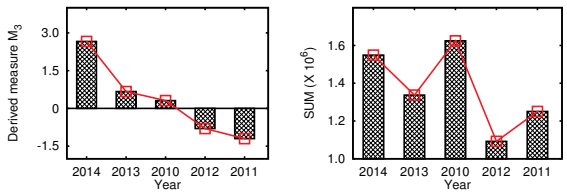
Insights from car sales: We first compare our insight with a raw aggregation result on car sales. Figure 12(a) refers to the top-2 shape insight in Table 5(a). Its SG(⟨*,*,SUV,*⟩, Year) means that we compare SUV cars by year. Its $\mathcal{C}_e$ = ⟨(SUM,Sales),(%, Category)⟩ means that we analyze the percentage of SUV's sales among all categories. Figure 12(a) shows that such a percentage rises with year. On the other hand, the raw aggregation result for the same SG(⟨*,*,SUV,*⟩, Year) does not reveal much information.

Figure 13(a) refers to the top-1 point insight (outstanding No.1) in Table 5(b). Its SG(⟨ *,*,SUV,* ⟩, Year) means we compare SUV cars by year. Its $\mathcal{C}_e$ = ⟨(SUM,Sales),(%,Year),(Δ_{avg}, Category)⟩ means we analyze SUV's yearly share over the average yearly share of all categories. Figure 13(a) shows that in 2014, SUV exhibits the most advantages over the other years. However, the raw aggregation result in Figure 13(b) does not reveal this information.



(a) insight: rising trend (b) raw aggregation result

Figure 12: Car sales shape insight: SG(⟨∗, ∗,SUV,∗⟩, Year)

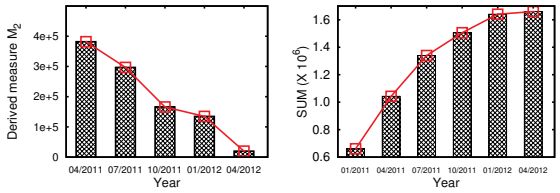


(a) Point insight: Outstanding No.1 (b) raw aggregation result

Figure 13: Car sales point insight: SG(⟨ *,*,SUV,* ⟩, Year)

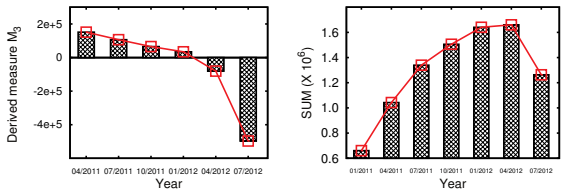
Insights from tablet sales: We then compare our insights with a raw aggregation result on tablet sales. Figure 14(a) refers to the top-1 shape insight in Table 5(c). Its SG(⟨∗, · · · , ∗⟩, Year) means that we compare the tablet sales by year. Its $\mathcal{C}_e$ = ⟨(SUM,Sales),(Δ_{prev},Year)⟩ means that we analyze the incremental sales between successive years. As shown in Figure 14(a), the incremental sales falls with year. In contrast, the raw aggregation result in Figure 14(b) only shows a rising trend, but it is not as informative as the above insight. We elaborate the details in Appendix F.

Figure 15(a) refers to the top-1 point insight (outstanding Last) in Table 5(d). Its SG(⟨ *,· · · ,* ⟩,Year) means we compare tablet sales by year. Its $\mathcal{C}_e$ = ⟨(SUM,Sales), (Δ_{prev}, Year), (Δ_{avg}, Year)⟩ means we analyze the incremental sales of each year over the average of incremental sales among all years. Figure 15(b) shows that 07/2012 is the "outstanding last" when compared with the other years. On the other hand, the raw aggregation results in Figure 15(b) do not reveal the above insight.



(a) insight: falling trend (b) raw aggregation result

Figure 14: Tablet sales shape insight: SG(⟨∗, · · · , ∗⟩, Year)



(a) Point Insight: Outstanding Last (b) raw aggregation result

Figure 15: Tablet sales point insight: SG(⟨ *,· · · ,* ⟩, Year)

8.2 Insight Utility Study

In this section, we assess the utility of our top-k insights by 6 domain experts from a leading IT company.

Intern dataset: This dataset is obtained from the University Relationship (UR) team of the above IT company from 2012 to 2016. It contains 1,201 tuples. Each tuple (i.e., an intern) has 15 dimensions. The domain sizes of dimensions are: *Year (4)*, *Group (50)*, *Name (1109)*, *FullTime (2)*, *Start Quarter (13)*, *End Quarter (13)*, *Duration (4)*, *Mentor (300)*, *Nationality (16)*, *Degree (3)*, *Origin (20)*, *University (200)*, *Department (813)*, *Research Area (511)*, *Advisor (831)*. The aggregate function is COUNT in this study.

Study methodology: We first extract top-5 insights with depth-2 and depth-3 composite extractors, and illustrate these insights in Table 9 in Appendix G. Due to confidential reasons from the data provider, we anonymize some attributes by pseudo-values (e.g., A, B, C, D).

In the following user study, we invite 3 UR managers and 3 data analysts (from the above IT company) and call them *the domain experts* because they have conducted analysis on this dataset before. We conduct one-on-one interviews with them, collect their comments on our insights, and also ask them to rate the insights by the following two metrics:

1. **Usefulness**: (from 1 to 5), a higher score indicates more useful insight.

2. **Difficulty**: (from 1 to 5), a higher score indicates that the insight is more difficult to obtain by using an existing data analysis tool (i.e., Microsoft Excel PivotTable).

Results and Feedback: In these interviews, the domain experts appreciate our top-k insights and find them to be quite useful. They agreed that most of our insights are actionable. For example, the UR team may take actions to improve the intern diversity from a certain region, or further analyze the root cause of some unusual quarter regarding the number of check-in and check-out interns.

We report the ratings of our top-5 insights by the domain expert in Table 9. The average usefulness score of depth-2 insights and depth-3 insights are 3.24 and 3.76, respectively. On the other hand, the average difficulty score of depth-2 insights and depth-3 insights are 2.88 and 4.12, respectively. In summary, the domain experts agreed that depth-3 insights are more useful. However, these insights are harder to summarize with their data analysis tool.

8.3 Human Effort Study

We measure the time taken by users to obtain our insights by using other tools. Due to page limits, we place the details in Appendix H. In summary, users spend significant time compute our insights (i.e., 29.2 minutes by SQL, 14.2 minutes by Microsoft Excel PivotTable), whereas our system takes just 0.17 seconds.

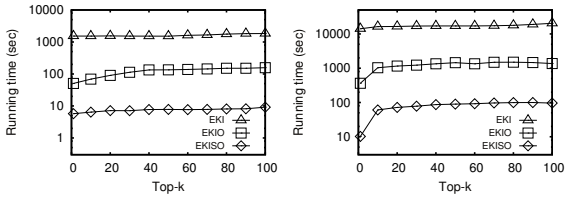
9. PERFORMANCE EVALUATION

We proceed to evaluate the performance of our solutions. We conducted all experiments (with single thread) on a machine with an Intel i7-3770 3.4GHz processor, 16GB of memory. We implemented our three solutions in C#. We denote the baseline solution (cf. Algorithm 1) for Extracting top-K Insights problem as EKI. EKIO (cf. Algorithm 3 in Appendix E.1) applies all optimization techniques in Section 6. EKISO (cf. Algorithm 6 in Appendix E.4) applies techniques in Section 6 (optimizations) and Section 7 (computation sharing).

We report the running time (i.e., wall clock time) of solutions in our experiments. Before running experiments, we load the datasets from disk to main memory. SUM is used as the aggregate function. As discussed in Section 2.2, unless otherwise stated, the depth of composite extractor τ is set to 2 or 3 by default.

First, we show the efficiency of our solutions on a real dataset. Then, we investigate the efficiency and scalability of our solutions on TPC-H data with respect to various parameters.

9.1 Real dataset: Tablet sales



(a) at τ=2 (b) at τ=3

Figure 16: Runtime on tablet sales vs. result size k

Among the real datasets described in Section 8, we use only the tablet sales dataset as it is much larger than the car sales and intern dataset. In Figure 16, we vary the result size k and report the running time of solutions on the tablet sales dataset. EKIO performs better than EKI by 10 times, implying the power of our proposed sibling cube and optimization techniques. Since EKISO employs computation sharing techniques, it further outperforms EKIO by an order of magnitude.

9.2 TPC-H dataset

TPC-H[11] **data:** By default, we generate TPC-H data with scale factor set to 1. We extract the *lineitem* table, which contains 6,001,215 tuples and 16 dimensions. We use $l\_extendedprice$ (ranging from 901.00 to 104949.50) as measure. We use the following 6 dimensions; their domain sizes are as follows: $l\_shipdate(2526)$, $l\_discount(11)$, $l\_returnflag(3)$, $l\_shipinstruct(7)$, $l\_shipmode(4)$ and $l\_linestates(2)$.

We then study the efficiency and scalability of our methods for various parameters. The default parameter setting is: the number of tuples N=1,000,000, the depth of composite extractor τ=2, the result size k=10, and the number of dimensions d=6.

Effect of result k: Figure 17(a) compares the performance of our solutions by varying k from 1 to 100. EKISO is two orders of magnitude faster than EKI. It allows us to obtain the top-1, top-10, top-100 results at 94s, 137s, 622s, respectively. Its running time scales sub-linearly with k.

Effect of number of tuples N: Then we test the performance of our solutions with respect to N. According to Figure 17(b), EKISO outperforms EKI by at least two orders of magnitude. Their performance gap widens as N increases. The running time of EKISO also rises sub-linearly with N.

Effect of dimensions d: In Figure 17(c), we vary the number of dimensions d from 2 to 10. In addition to the 6 dimensions mentioned earlier, we include 4 more dimensions when $d > 6$: *l_suppkey(1000) l_tax(9)*, *l_linenumber(7)* and *l_quantity(50)*. When d is small (≤ 4), we can obtain top-10 results in 1–2 minutes. At large values of d, the running time becomes high due to the huge combinations of composite extractors and sibling groups. The performance gap widens with d, and EKISO achieves three orders of magnitude improvement over EKI at d=10.

Effect of depth τ: Next, we examine the performance of our solutions with respect to the depth of composite extractor (from 2 to 4). As illustrated in Figure 17(d), the running time rises with the depth because the number of possible composite extractors increases rapidly with τ. EKISO again outperforms the other solutions. We omitted EKI as it is too slow.

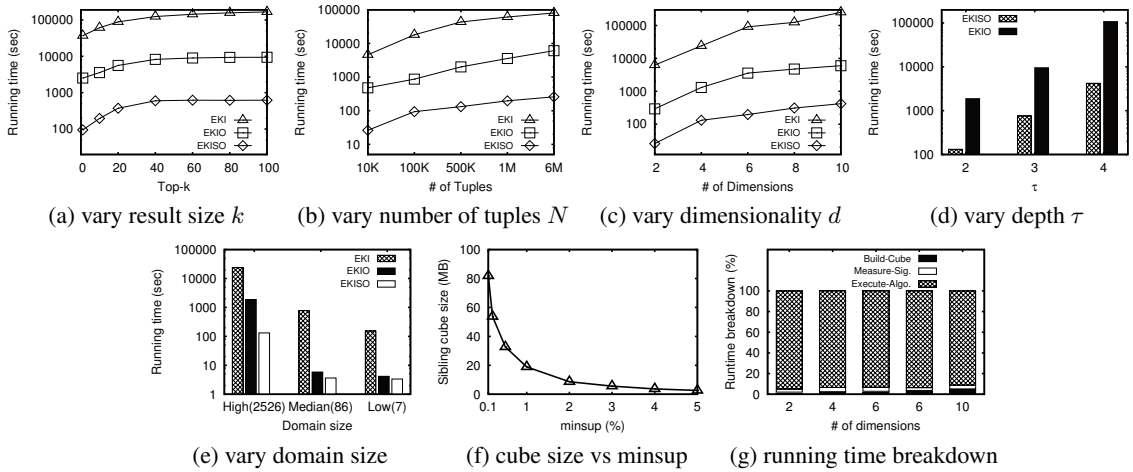
Effect of domain size: In this experiment, we vary the domain size of the *l_shipdate* dimension and fix the domain size of other dimensions. The default domain size of *l_shipdate* is 2562 values (one per day). We obtain smaller domain sizes by changing the granularity: 84 values (one per month), 7 values (one per year). As displayed in Figure 17(e). EKISO is significantly faster than EKI. EKISO and EKIO perform similarly at low and median domain sizes because the total number of sibling groups is quite small.

Finally, we evaluate the space and running time overhead of our sibling cube.

Space: Figure 17(f) displays the sibling cube size with respect to the iceberg constraint $minsup$. The size of sibling cube ranges from 87.6MB to 1.68MB. All of them can fit in main memory.

Running time breakdown: Figure 17(g) shows the breakdown of the running time of EKISO. We vary the dimensionality d from 2 to 10. Observe that the sibling cube construction time and insight

[11] http://www.tpc.org/tpch/



(a) vary result size k (b) vary number of tuples N (c) vary dimensionality d (d) vary depth τ

(e) vary domain size (f) cube size vs minsup (g) running time breakdown

Figure 17: Performance results on the TPC-H data

significance measurement time occupy only less than 5% of the total running time.

10. RELATED WORK

Multiple research areas are relevant to our problem. In the following, we review the related work in each relevant area.

Top-k queries: Top-k queries have been extensively studied in databases [16]. They require user to specify a ranking function (or the weighting of attributes), and then return k result objects. In contrast, our problem does not require any user parameter and our results are insights rather than objects.

OLAP data cube: The OLAP data cube model [14] supports efficient aggregation on a multi-dimensional dataset and allows users to navigate the aggregation result by operations (e.g., slicing, dicing, drill up/down). Efficient construction algorithms for data cubes have also been studied [13, 8]. In our problem context, data cube supports only aggregation but not the computation of extractors and insights.

Advanced cubes have been proposed for other forms of analysis beyond aggregation such as dominant relationship analysis [18], statistical analysis [19], and ranking analysis [33, 34]. Nevertheless, these cubes cannot be readily applied to evaluate composite extractors and insights in our problem. Thus, these existing OLAP systems cannot be readily applied to evaluate composite extractors and insights in our problem.

Subspace mining: The data mining community has developed subspace mining techniques to discover interesting subspaces from a dataset [22, 34]. Müller et al. [22] have studied various problems on subspace mining analysis (e.g., subspace clustering, outlier mining). However, the work in [22] has not considered the composite extractor and insights in our paper. Wu et al. [34] studied how to find the top-R subspaces with the highest 'promotiveness' values, for a given query object specified by user. Our problem differs from [34] in two ways. First, [34] requires a query object, while our problem does not require any user input. Second, our proposed composite extractors and insights have not been studied in [34].

Mining and learning-based techniques: Recent works [20, 2] employ data mining techniques (e.g., outlier detection, cluster analysis) and machine learning techniques (e.g., inductive learning [21]) on datasets to perform pattern discovery and predictive analytics, respectively. Our problem differs from them in two aspects. First, we focus on the data warehousing setting, in which users are more interested in computing aggregations on a fact table. Our proposed insights can be considered as interesting observations derived from aggregation results in multiple steps. Second, since existing algorithms in DM and ILP (e.g., outlier detection [4], subgroups discovery [20]) are dedicated to the statistical modeling of data or the prediction of future data, they cannot be adapted to compute our top-k insights.

Exploratory analysis: The most relevant works in exploratory analysis area are: Sarawagi et al. [25], Wu. et. al [34], ARcube [33], IBM Cogons[1], and SEEDB [30]. We summarize the main differences between these works and our proposal in Table 6, with respect to four features, i.e., user input, top-k, result, and concepts. Our proposal only requires the user to provide the number of results (k), but the other works demand more input from the user, Thus, our proposal is more user-friendly for non-expert users. Observe that the result and concepts of our proposal are also significantly different from those of the other works.

Data exploration: Data exploration is about efficiently extracting knowledge from data [15]. In the database community, several works have investigated efficient data exploration [23, 26, 12, 27]. We omit the discussion of these works and refer readers to the recent overview paper [15]. Unlike these works, our insight extraction solution can return interesting insights to users automatically, instead of letting the users determine whether it is interesting [31].

11. CONCLUSION AND FUTURE WORK

Conclusion: This paper investigates how to extract top-k insights from multi-dimensional data. We proposed a systematic computation framework for this problem, and a suite of performance optimization techniques (e.g., pruning, ordering, sibling cube and computation sharing). Our effectiveness studies (e.g., case study, utility study) have demonstrated that top-k insights reveal meaningful observations on different real datasets. Our best solution EKISO outperforms the baseline solution by several orders of magnitude.

Future work: This paper takes the first attempt to extract insights hidden in the data. We want to pursue several promising directions in the future to support both expert data analysts and non-expert executives or managers. First, we plan to incorporate user feedback in insight extraction. Second, for massive datasets, we will investigate how to extract insights efficiently in a distributed environment.

Approaches	User input	Top-k	Result	Concepts
Sarawagi. et. al [25]	OLAP operations	No	anomalies	precomputed expectations
Wu. et. al [34]	promotion objectives, integer k	Yes	subspaces	subspace ranking function
ARcube [33]	aggregate queries, integer k	Yes	aggregate values	materialized data cubes
IBM Cogons [1]	OLAP operations	No	aggregate values	traditional OLAP techniques
SEEDB [30]	queries	No	visualization	visualization and DBMS
Our paper	**integer k**	**Yes**	**insights**	**composite extractor / ranking on insights**

Table 6: Comparison with related works

Acknowledgements

Man Lung Yiu and Bo Tang were supported by grant GRF 152043/15E from the Hong Kong RGC. The authors would like to thank the participants of our insight utility study, and the anonymous reviewers for their valuable comments.

12. REFERENCES

[1] Ibm cogons. https://goo.gl/6dYxLc.

[2] Ibm watson analytics. http://goo.gl/EK1nNU.

[3] Quick insights in microsoft power bi. https://goo.gl/xHwCLg.

[4] C. C. Aggarwal and P. S. Yu. Outlier detection for high dimensional data. In *ACM Sigmod Record*, 2001.

[5] C. Anderson. *The long tail: Why the future of business is selling more for less*. Hyperion, 2008.

[6] N. Balakrishnan. *Handbook of the logistic distribution*. CRC Press, 2013.

[7] C. Balkesen, G. Alonso, J. Teubner, and M. T. Özsu. Multi-core, main-memory joins: Sort vs. hash revisited. *PVLDB*, 2013.

[8] K. Beyer and R. Ramakrishnan. Bottom-up computation of sparse and iceberg cube. In *SIGMOD Record*, 1999.

[9] S. Chaudhuri. What next?: a half-dozen data management research goals for big data and the cloud. In *PODS*, 2012.

[10] S. Chaudhuri, U. Dayal, and V. Narasayya. An overview of business intelligence technology. *Communications of the ACM*, 2011.

[11] D. Dash, J. Rao, N. Megiddo, A. Ailamaki, and G. Lohman. Dynamic faceted search for discovery-driven analysis. In *CIKM*, 2008.

[12] K. Dimitriadou, O. Papaemmanouil, and Y. Diao. Explore-by-example: An automatic query steering framework for interactive data exploration. In *SIGMOD*, 2014.

[13] M. Fang, N. Shivakumar, H. Garcia-Molina, R. Motwani, and J. D. Ullman. Computing iceberg queries efficiently. In *VLDB*, 1999.

[14] J. Gray, S. Chaudhuri, A. Bosworth, A. Layman, D. Reichart, M. Venkatrao, F. Pellow, and H. Pirahesh. Data cube: A relational aggregation operator generalizing group-by, cross-tab, and sub-totals. *Data Mining and Knowledge Discovery*, 1997.

[15] S. Idreos, O. Papaemmanouil, and S. Chaudhuri. Overview of data exploration techniques. In *SIGMOD*, 2015.

[16] I. F. Ilyas, G. Beskales, and M. A. Soliman. A survey of top-*k* query processing techniques in relational database systems. *ACM Comput. Surv.*, 2008.

[17] M. Krzywinski and N. Altman. Points of significance: Significance, p values and t-tests. *Nature methods*, 2013.

[18] C. Li, B. C. Ooi, A. K. Tung, and S. Wang. Dada: a data cube for dominant relationship analysis. In *SIGMOD*, 2006.

[19] X. Li, J. Han, Z. Yin, J.-G. Lee, and Y. Sun. Sampling cube: a framework for statistical olap over sampling data. In *SIGMOD*, 2008.

[20] P. Y. Lum, G. Singh, A. Lehman, T. Ishkanov, M. Vejdemo-Johansson, M. Alagappan, J. Carlsson, and G. Carlsson. Extracting insights from the shape of complex data using topology. *Scientific Reports*, 2013.

[21] R. S. Michalski. A theory and methodology of inductive learning. In *Machine learning*. Springer, 1983.

[22] E. A. Müller. *Efficient knowledge discovery in subspaces of high dimensional databases*. PhD thesis, RWTH Aachen University, 2010.

[23] S. Sarawagi. Explaining differences in multidimensional aggregates. In *VLDB*, 1999.

[24] S. Sarawagi. User-adaptive exploration of multidimensional data. In *VLDB*, 2000.

[25] S. Sarawagi, R. Agrawal, and N. Megiddo. Discovery-driven exploration of olap data cubes. In *EDBT*, 1998.

[26] S. Sarawagi and G. Sathe. i3: intelligent, interactive investigation of olap data cubes. In *ACM SIGMOD Record*, 2000.

[27] T. Sellam and M. L. Kersten. Meet charles, big data query advisor. In *CIDR*, 2013.

[28] T. Sellam, E. Müller, and M. L. Kersten. Semi-automated exploration of data warehouses. In *CIKM*, 2015.

[29] R. H. Shumway and D. S. Stoffer. *Time series analysis and its applications: with R examples*. Springer Science & Business Media, 2010.

[30] M. Vartak, S. Rahman, S. Madden, A. Parameswaran, and N. Polyzotis. SEEDB: efficient data-driven visualization recommendations to support visual analytics. *PVLDB*, 2015.

[31] A. Wasay, M. Athanassoulis, and S. Idreos. Queriosity: Automated data exploration. In *IEEE Congress on Big Data*, 2015.

[32] P. Wu, Y. Sismanis, and B. Reinwald. Towards keyword-driven analytical processing. In *SIGMOD*, 2007.

[33] T. Wu, D. Xin, and J. Han. Arcube: supporting ranking aggregate queries in partially materialized data cubes. In *SIGMOD*, 2008.

[34] T. Wu, D. Xin, Q. Mei, and J. Han. Promotion analysis in multi-dimensional space. *PVLDB*, 2009.

[35] D. Xin, J. Han, X. Li, Z. Shao, and B. W. Wah. Computing iceberg cubes by top-down and bottom-up integration: The starcubing approach. *TKDE*, 2007.

APPENDIX

A. COMPOSITION TAXONOMY

We give a well-defined composition closure to ensure the validity of generated composite extractors, as shown in Table 7. The result of extractors in the first column serves as the input of extractors in the first row. '✓' means a meaningful composition, '✗' means a meaningless composition.

ξ	$Rank$	%	Δ_{avg}	Δ_{prev}
$Rank$	✓	✗	✓	✓
%	✓	✗	✓	✓
Δ_{avg}	✓	✗	✓	✓
Δ_{prev}	✓	✗	✓	✓

Table 7: Composition taxonomy for adjacent extractors

Example: The composite extractor $\mathcal{C}_e$ = ⟨SUM, (Δ_{prev},Year), ($Rank$, Brand)⟩ is valid according to Table 7. It corresponds to the semantic that ranks the value of "difference from previous year" across all brands in the dataset in Table 1. However, the composite extractor $\mathcal{C}_e$ = ⟨SUM, ($Rank$, Year), (%, Brand)⟩ is invalid, because it does not make sense to calculate the percentage of ranking positions.

B. VALIDITY

DEFINITION 7 (VALIDITY OF $\mathsf{SG}(S, D_i)$ AND $\mathcal{C}_e$). *A sibling group $\mathsf{SG}(S, D_a)$ is a* **valid input** *for composite extractor $\mathcal{C}_e$ iff for each pair (ξ, D_x) in $\mathcal{C}_e$, D_x=D_a or $S[D_x] \neq *$.*

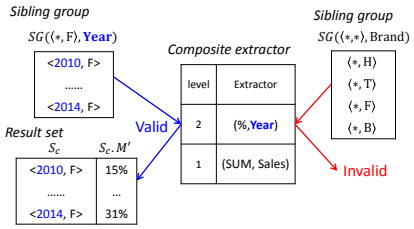


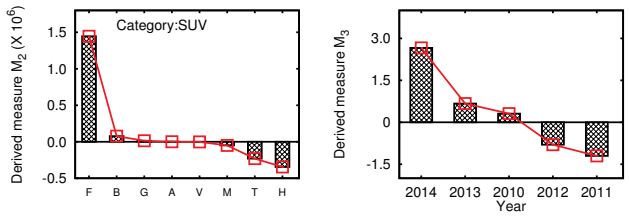
Figure 18: Example of $\mathsf{SG}(S, D_i)$ and $\mathcal{C}_e$

Example: In Figure 18, a sibling group SG(⟨*,F⟩,Year) is valid for the composite extractor $\mathcal{C}_e$ = ⟨(SUM,Sales),(%,Year)⟩), as the dimension 'Year' in $\mathcal{C}_e[2]$ has known values in every subspace S_c of SG(⟨*,F⟩,Year). However, the sibling group SG(⟨*,*⟩,Brand) is not valid for the same $\mathcal{C}_e$ because SG(⟨*,*⟩,Brand) does not have known values in the dimension 'Year'.

C. INSIGHT EVALUATION

We describe the significance evaluation procedure for each type of insight in Table 8.

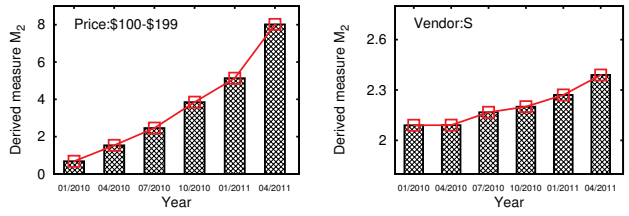
Figure 19 shows the significance value of the insight "outstanding No.1" for two different pairs of $(\mathsf{SG}(S, D_i), \mathcal{C}_e)$. Figure 19(a) shows that the highest bar is remarkably higher than other bars, thus the significance value is high (0.96). In Figure 19(b), since the highest bar is not much higher than other bars, the significance is relatively low (0.36).



(a) Sig = 0.97 (b) Sig = 0.36

Figure 19: The significance of outstanding No.1

As another example, we then illustrate the significance of shape insights. In Figure 20(a), the derived measure increases quickly from 01/2010 to 04/2011 (i.e., high slope), thus its significance is high (0.99). However, in Figure 20(b), the derived measure rises slowly, so its significance is low (0.31).



(a) Sig = 0.99 (b) Sig = 0.31

Figure 20: The significance of rising trend

D. CORRELATION INSIGHT

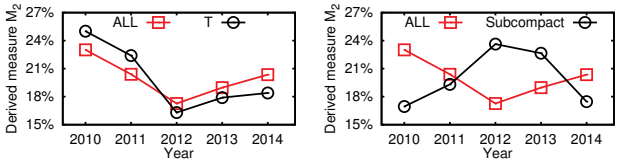
We first introduce the correlation insight, and then demonstrate the top-2 correlation insights on a real dataset.

Correlation Insight: Given two sibling groups $\mathsf{SG}(S_a, D_i)$ and $\mathsf{SG}(S_b, D_i)$ where D_i is an ordinal dimension, a positive / negative correlated insight means that the series $S_a.\mathcal{M}_\tau$ and $S_b.\mathcal{M}_\tau$ exhibit a positive / negaitve correlation with D_i.

We denote the computation result of two sibling groups (i.e., $\mathsf{SG}(S_a, D_i)$ and $\mathsf{SG}(S_b, D_i)$) with composite extractor $\mathcal{C}_e$ as two time series $\mathcal{X}$ and $\mathcal{Y}$. Generally, data analysts are interested in positive / negative correlated time series. Thus, we set the null hypothesis of correlated insight as:

$$H_0 : \text{the Pearson correlation } \rho\,(\mathcal{X}, \mathcal{Y}) \approx 0$$

The p-value reveals how surprisingly the correlation $\rho(\mathcal{X}, \mathcal{Y})$ is against the null hypothesis H_0, with the correlation coefficient following the Normal distribution $N(\mu, \delta)$ [29], where $\mu = 0$, $\delta = 0.05$. Thus, the p-value of the correlation of two time series can be calculated by $Pr(\rho(\mathcal{X}, \mathcal{Y})|N(\mu, \delta))$.



(a) Positive correlated SG({ * }, Year) with SG({T}, Year)

(b) Negative correlated SG({ * }, Year) with SG({Subcompact}, Year)

Figure 21: Correlation insights in Tablet sales

We compute some correlation insights on the car sales dataset, and then discuss about them briefly. The top-2 correlation insights of car sales dataset are visualized in Figures 21(a) and (b). Figure 21(a) reveals the percentage of brand T's sales is positively correlated to the percentage of the entire car market from 2010 to 2014. However, the percentage of Subcompact's (i.e., Category=Subcompact) sales is negatively correlated to the percentage of the entire car market from 2010 to 2014, as shown in Figure 21(b).

E. ALGORITHMS

E.1 Algorithm EKIO

Algorithm 3 is an optimized version of Algorithm 1 (cf. Section 5.1) that incorporates all optimization techniques in the above subsections. We employ a pruning technique (cf. Section 6.1) at Line 6. We apply loop ordering techniques (cf. Section 6.2) at Lines 3 and 10. Also, we construct an sibling cube SIBCUBE (cf. Section 6.3) at Line 2, and then use it at Lines 7 and 9–10. At Line 12, we store the value of $\mathcal{M}$, obtained at Line 10, in S'.SUM. When the recursive call requires $\mathsf{SUM}(S')$, it can access S'.SUM immediately.

E.2 Sharing within a Sibling Group

Algorithm 4 applies Lemma 3 (cf. Section 7.1) to accelerate the computation of $\mathcal{C}_e$ on $\mathsf{SG}(S, D_i)$. First, we check the condition in Lemma 3. If it is not satisfied, we revert back to calling Algorithm 2. Otherwise, we invoke the function 'RecurExtractII' to compute an intermediate result set Φ_τ (Line 6) and then reuse it to obtain the derived measure for each subspace in $\mathsf{SG}(S, D_i)$.

Note that 'RecurExtractII' returns a pair (M', Φ_{level}), where Φ_{level} is the intermediate result set for computing the derived measure M' of subspace S'. Actually we only need Φ_{level} at level τ and only M' at other levels.

Insight types	Description & Significance definition
Point Insight: *outstanding No.1*	Given a group of numerical values $\mathcal{X} = \{x_1, x_2, ..., x_n\}$, the significance of the biggest value x_{max} being *Outstanding No. 1* of $\mathcal{X}$ is defined by the p-value against the null hypothesis H_0: $\mathcal{X}$ follows a power-law distribution.
Significance calculation: p-value of *outstanding No.1*	i) sort $\mathcal{X}$ in descending order; ii) conduct regression analysis for the values $\mathcal{X} \setminus x_{max}$ using power-law function $\alpha \cdot i^{-\beta}$, where i is index iii) use residuals in regression analysis to train a Gaussian model $N(\mu, \delta)$; iv) obtain the residual ϵ_{max} by $\hat{x}_{max} - x_{max}$ v) calculate the p-value by $P(\epsilon > \epsilon_{max}\|N(\mu, \delta))$. vi) compute the significance of x_{max} by $1 - P(\epsilon > \epsilon_{max}\|N(\mu, \delta))$
outstanding Last	Replace $x_{max} \in \mathcal{X}$ by $x_{min} \in \mathcal{X}$, and the significance calculation is the same as *Outstanding No.1*
Example	cf. Figure 19
Shape Insight: *trend*	A time series has an remarkable trend (increase/decrease) with a certain turbulence level (steadily/with turbulence). For a time series $\mathcal{X} = \langle x_1, x_2, ..., x_n \rangle$, the trend insight reflects a relatively sustained trend of $\mathcal{X}_i$. The significance of shape insight is defined by the p-value against the null hypothesis $H_0 : \mathcal{X}$ forms a shape with slope ≈ 0
Significance calculation: trend significance	i) fit $\mathcal{X}$ to a line by linear regression analysis and obtain goodness-of-fit value r^2. ii) compute the $slope$ of the $\mathcal{X}$'s fitted line iii) employ a logistic distribution to capture the distributions of slope $L(\mu, \lambda)$, $\mu = 0.2$ and $\lambda = 2$ iv) calculate the p-value by $P(s > \|slope\|\|L(\mu, \delta))$ v) compute the significance of $\mathcal{X}$ by $r^2 \cdot (1 - P(s > \|slope\|\|L(\mu, \delta)))$
Example	cf. Figure 20

Table 8: Insights categories and evaluation procedures

Algorithm 3 Insights+Optimized (dataset $\mathcal{R}(\mathcal{D}, \mathcal{M})$, depth τ, result size k)

1: *run Lines 1–3 in Alg. 1*
2: construct an sibling cube SIBCUBE ▷ Sec. 6.3
3: sort $D_i \in \mathcal{D}$ by ascending domain size ▷ Sec. 6.2
4: *run Lines 4–8 in Alg. 1* ▷ call the function below

Function: EnumerateInsightI ($S, D_i, \mathcal{C}_e$):
5: **if** isValid(SG$(S, D_i), \mathcal{C}_e$) **then**
6: **if** $ub_k \leq \mathbb{S}^{UB}$(SG$(S, D_i), \mathcal{C}_e$, T) **then** ▷ Sec. 6.1
7: $\Phi \leftarrow$ use SIBCUBE in ExtractΦ(SG$(S, D_i), \mathcal{C}_e$) ▷ Alg. 2
8: *run Lines 11–14 in Alg. 1*
9: sorted list $L \leftarrow$ SIBCUBE$[S \circ D_i]$ ▷ Sec. 6.3
10: **for** each value-measure pair $(v, \mathcal{M}) \in L$ **do** ▷ Sec. 6.2
11: $S' \leftarrow S, S'[D_i] \leftarrow v$
12: S'.SUM $\leftarrow \mathcal{M}$ ▷ stored value of SUM(S')
13: *run Lines 17–18 in Alg. 1* ▷ Line 18: EnumerateInsightI

Algorithm 4 ExtractΦII(SG$(S, D_i), \mathcal{C}_e$)

1: **if** $D_i \neq \mathcal{C}_e[\tau].D_x$ **then** ▷ test for Lem. 3
2: $\Phi \leftarrow$ ExtractΦ(SG$(S, D_i), \mathcal{C}_e$) ▷ Alg. 2
3: **else**
4: initialize a result set $\Phi \leftarrow \emptyset$
5: $S' \leftarrow S, S'[D_i] \leftarrow dom(D_i).first$
6: $(M', \Phi_\tau) \leftarrow$ RecurExtractII($S', \tau, \mathcal{C}_e$)
7: **for each** value $v \in dom(D_i)$ **do**
8: $S' \leftarrow S, S'[D_i] \leftarrow v$
9: $M' \leftarrow$ derived $S'.M'$ after applying $\mathcal{C}_e[\tau]$ on Φ_τ
10: insert (S', M') into Φ
11: **return** Φ

Function: RecurExtractII(Subspace $S', level, \mathcal{C}_e$):
12: initialize a result set Φ_{level}
13: $D_\xi \leftarrow \mathcal{C}_e[level].D_x$
14: **if** $level > 2$ **then**
15: **for each** value v in D_ξ **do**
16: $S_v \leftarrow S', S_v[D_\xi] \leftarrow v$
17: $(M'_v, \Phi_{temp}) \leftarrow$ RecurExtractII($S_v, level - 1, \mathcal{C}_e$)
18: insert (S_v, M'_v) into Φ_{level}
19: **else**
20: $\Phi_{level} \leftarrow$ SIBCUBE$[S' \circ D_\xi]$ ▷ Sec. 6.3
21: $M' \leftarrow$ derived $S'.M'$ by applying $\mathcal{C}_e[level]$ on Φ_{level} ▷ Def. 3
22: **return** (M', Φ_{level})

E.3 Sharing across Sibling Groups

To save computation cost, we need to detect the shared content and reuse it, as depicted in Figure 10(b). We present Algorithm 5 to integrate Algorithm 2 with a sharing technique (cf. Section 7.2). It employs a hash table Ψ to store the derived measure $S'.M'$ for subspace S' that we have processed before (in other sibling groups). If Ψ contains S', we can retrieve its derived measure from Ψ immediately. Otherwise, we need to compute the derived measure and store it into Ψ.

Algorithm 5 ExtractΦIII(SG$(S, D_i), \mathcal{C}_e$, hash table Ψ)

1: initialize a result set $\Phi \leftarrow \emptyset$
2: **for each** value v in $dom(D_i)$ **do**
3: $S' \leftarrow S, S'[D_i] \leftarrow v$
4: **if** Ψ contains S' **then**
5: $M' \leftarrow \Psi[S']$ ▷ get from Ψ
6: **else**
7: $M' \leftarrow$ RecurExtract($S', \tau, \mathcal{C}_e$) ▷ function in Alg. 2
8: $\Psi[S'] \leftarrow M'$ ▷ store into Ψ
9: insert (S', M') into Φ
10: **return** Φ

E.4 Algorithm EKISO

Algorithm 6 is an integrated version of Algorithm 3 (cf. Appendix E.1) that incorporates sharing computation techniques in Section 7. We employ sharing within a sibling group (cf. Section 7.1) at Line 14. We apply sharing across sibling groups (cf. Section 7.2) at Line 12. Hash table Ψ will be flushed for each composite extractor at Line 4.

F. TOP-2 INSIGHTS IN TABLET SALES

In this section, we discuss the two insights in Table 5(c). We illustrate the significance of these two insights, i.e., $\mathsf{Sig}_{\mathsf{Shape}}(\Phi_1)$ and $\mathsf{Sig}_{\mathsf{Shape}}(\Phi_2)$, in Figures 22(a) and (b), respectively.

The insight scores of top-1 and top-2 shape insights were computed as $\mathsf{Imp}(\mathsf{SG}(*, \mathsf{Year}) \cdot \mathsf{Sig}_{\mathsf{Shape}}(\Phi_1) = 1 \cdot 0.96 = 0.96$, and $\mathsf{Imp}(\mathsf{SG}(\{\mathsf{WiFi}\}, \mathsf{Year}) \cdot \mathsf{Sig}_{\mathsf{Shape}}(\Phi_2) = 0.643 \cdot 0.99 = 0.64$, respectively. The top-1 and top-2 shape insights have similar significance scores (i.e., 0.96 and 0.99). However, the top-1 shape insight is applicable to the entire tablet market (i.e., $\mathsf{Imp}(\mathsf{SG}(*, \mathsf{Year}) =$

Usefulness	Difficulty	SG(S,D_i)	Composite Extractor $\mathcal{C}_e$
2.8	2.6	SG({*}, End Quarter)	⟨(COUNT),(Δ_{Prev}, EndQuarter)⟩
Top-1 Insight		We consider the increase of check-out interns in each quarter with the previous quarter. The largest increase happens in 2016Q2.	
2.6	2.2	SG({3 months}, Nationality)	⟨(COUNT),(Δ_{avg}, Duration)⟩
Top-2 Insight		The internship duration of interns from Country B is always 3 months.	
3.6	3.6	SG({6 months}, Start Quarter)	⟨(COUNT),(Δ_{Prev}, StartQuarter)⟩
Top-3 Insight		Consider the increase of check-in interns whose internship duration is 6 months among successive start quarters. The largest increase happens in 2015Q1.	
3.6	3.0	SG({PhD}, Nationality)	⟨(COUNT),(Δ_{avg}, Degree)⟩
Top-4 Insight		Consider all interns from Country A. The number of PhD interns is obviously higher than the number of interns with other degrees.	
3.6	3.0	SG({Undergraduate}, Year)	⟨(COUNT),(Rank, Degree)⟩
Top-5 Insight		Regarding the rank (i.e., rank by the number of interns) of each degree among all years, the rank of undergraduates is the lowest in 2013.	

(a) Top-5 insights from Intern Dataset with $\tau = 2$

3.8	3.6	SG({PhD}, Year)	⟨(COUNT),(%, Year), (Δ_{avg}, Degree)⟩
Top-1 Insight		Consider the percentage of interns in each year by their degrees. The percentage of PhD interns is the highest in 2014.	
3.4	4.4	SG({2015Q2}, Group)	⟨(COUNT),(%, Group), (Δ_{avg}, EndQuarter)⟩
Top-2 Insight		Group A at 2015Q2 is the best in terms of the percentage of check-out interns among all quarters.	
3.4	4.4	SG({2016Q1}, Group)	⟨(COUNT),(%, Group), (Δ_{prev}, StartQuarter)⟩
Top-3 Insight		Group B at 2016Q1 is the best in terms of the increased percentage of check-in interns among all successive quarters.	
4.4	4.0	SG({University H}, Group)	⟨(COUNT),(%, Group), (Δ_{avg}, University)⟩
Top-4 Insight		Group C is the most favorite group among all research groups for the interns from University H.	
3.8	4.2	SG({2014Q3}, Group)	⟨(COUNT),(%, Group), (Δ_{avg}, EndQuarter)⟩
Top-5 Insight		Group D at 2014Q3 is the worst in terms of the percentage of check-out interns among all quarters.	

(b) Top-5 insights from Intern Dataset with $\tau = 3$

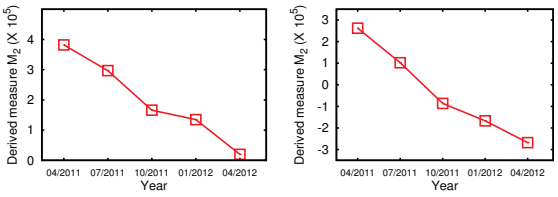
Table 9: User study result on the intern dataset with COUNT

Algorithm 6 Insights+Sharing+Optimized (dataset $\mathcal{R}(\mathcal{D}, \mathcal{M})$, depth τ , result size k)

```
1: create a hash table Ψ
2: run Lines 1–3 in Alg. 3
3: for each C_e ∈ O do
4:     clear Ψ
5:     for i := 1 to d do
6:         initialize subspace S ← ⟨*, *, ··· , *⟩
7:         EnumerateInsightIII( S, D_i, C_e, Ψ )
8: return H

   Function: EnumerateInsightIII ( S, D_i, C_e, Ψ ):
9: if isValid ( SG(S, D_i), C_e ) then
10:    if ub_k ≤ S^UB(SG(S, D_i), C_e, T) then
11:        if D_i ≠ C_e[τ].D_x then
12:            ExtractΦIII( SG(S, D_i), C_e, Ψ )        ▷ Alg. 5
13:        else
14:            ExtractΦII( SG(S, D_i), C_e )            ▷ Alg. 4
15:        run Line 8 in Alg. 3
16: run Lines 9–13 in Alg. 3          ▷ Line 13: EnumerateInsightIII
```

1), and the top-2 shape insight is only applicable to the sub-market for WIFI tablets (i.e., Imp(SG({WiFi}, Year) = 0.643). Thus, the interestingness score of the top-1 shape insight is higher than that of the top-2 shape insight (i.e., $0.96 > 0.64$).



(a) $\text{Sig}_{\text{Shape}}(\Phi_1)$=0.96 (b) $\text{Sig}_{\text{Shape}}(\Phi_2)$ =0.99

Figure 22: Significances of top-2 insights in Tablet sales

G. INSIGHT UTILITY STUDY

Tables 9(a) and (b) illustrate the top-5 insights extracted from the intern dataset with depth-2 and depth-3 composite extractors, respectively. Each insight has five attributes: usefulness score, difficulty score, sibling group, composite extractor, and its meaning in English. We conducted one-on-one interviews with 6 domain experts (i.e., managers, data analysts) from the leading IT company. In these interviews, we provided the last three attributes to them in a questionnaire, and then asked them to rate the insights by usefulness and difficulty. We reported the average ratings in Table 9.

H. HUMAN EFFORT STUDY

In this section, we measure the time taken by users to obtain our insights by using two existing tools: (i) SQL queries, (ii) Microsoft Excel PivotTable. Observe that these tools cannot be readily applied to extract our top-k insights. Nevertheless, we can provide users with a sibling group SG and a composite extractor $\mathcal{C}_e$, then ask users to compute the result of applying $\mathcal{C}_e$ on SG.

In this study, we invited 4 senior database researchers, who are proficient in SQL queries and familiar with Microsoft Excel PivotTable (i.e., a data analysis tool). To make this study manageable, we chose the top-3 insights obtained from car sales with depth-2 composite extractors. This car sales dataset and the depth of composite extractor (i.e., τ = 2) were chosen because it is smallest and simple for human effort study. We provided the participants with the sibling groups and composite extractors of those insights, and then asked them to compute the result in each case by two methods: (i) SQL queries and (ii) Microsoft Excel PivotTable, respectively. For each participant, we measure the total time of computing all three insights by using SQL queries and Microsoft Excel PivotTable, respectively. We exclude the time of loading data into the database and Excel files.

Given information	Analysis tool	User 1	User 2	User 3	User 4
(SG,$\mathcal{C}_e$) pairs	SQL Query	20.2	23.4	34.7	38.5
	Excel PivotTable	12.6	10.8	17.3	16.1

Table 10: Study on human effort (in minutes)

We reported the time taken in Table 10. On overage, the participants spent 29.2 minutes with SQL queries and 14.2 minutes with Microsoft Excel PivotTable to complete the above task. In contrast, our system just takes 0.17 seconds to compute the top-3 insights.