QuickInsights: Quick and Automatic Discovery of Insights from Multi-Dimensional Data

Rui Ding, Shi Han, Yong Xu, Haidong Zhang, Dongmei Zhang
Microsoft Research
Beijing, China
{juding, shihan, yox, haizhang, dongmeiz}@microsoft.com

ABSTRACT

Discovering interesting data patterns is a common and important analytical need in data, with increasing user demand for automated discovery abilities. However, automatically discovering interesting patterns from multi-dimensional data remains challenging. Existing techniques focus on mining individual types of patterns. There is a lack of unified formulation for different pattern types, as well as general mining frameworks to derive them effectively and efficiently. We present a novel technique *QuickInsights*, which quickly and automatically discovers interesting patterns from multi-dimensional data. *QuickInsights* proposes a unified formulation of interesting patterns, called *insights*, and designs a systematic mining framework to discover high-quality insights efficiently. We demonstrate the effectiveness and efficiency of *QuickInsights* through our evaluation on 447 real datasets as well as user studies on both expert users and non-expert users. *QuickInsights* is released in Microsoft Power BI.

ACM Reference format:

Rui Ding, Shi Han, Yong Xu, Haidong Zhang, and Dongmei Zhang. 2019. QuickInsights: Quick and Automatic Discovery of Insights from Multi-Dimensional Data. In *Proceedings of ACM SIGMOD conference (SIGMOD'19), 15 pages.* https://doi.org/10.1145/3299869.3314037

1. INTRODUCTION

Discovering interesting data patterns is a common and important analytical need when users try to obtain meaningful, useful, and actionable information hidden in data through data analysis and exploration [1][2][3][4][8][22][24][29]. Such interesting patterns include correlation, anomaly, trend, etc. [8]. Two examples of interesting patterns are shown in Figure 1. The left chart shows CPU usage of a server is exceptionally lower than the other servers. The right chart shows sales of tablet devices in China is trending upwards in recent years.

Exploratory visual analysis is a commonly used approach for understanding and reasoning about data to uncover interesting data patterns [9][35][36][39][40], in which users have to manually select data variables and specify visual encodings, either via a programming library (e.g., ggplot [31]) or via a graphical interface (e.g., Tableau [30]). Although manual specification is flexible for data exploration, it is non-trivial to iteratively create and refine visualizations to search for the ones that are interesting and useful [9][22], especially for non-expert users who have limited time and limited skills in statistics and data visualization [29][40].

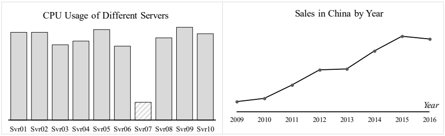


Figure 1. Two examples of interesting patterns

To speed up the data exploration process, we can complement interactive visual exploration tools with automated recommendation of interesting data patterns. With patterns automatically mined from the data and presented to users as visualizations, users can jump-start the exploration from them rather than from the scratch [28][29][36]. The patterns capture characteristics of a dataset from different perspectives, so they can help users understand data and prioritize their exploration actions. Some patterns may hit an "interesting zone" of users, thus inspiring them to generate new hypotheses and initiate further data exploration and analysis. Further, some patterns can directly lead to actions, e.g., system admin could login to server Svr07 for diagnosis when they find unexpected lower CPU usage from the data. Hence, Gartner's report [25] has identified smart, automated pattern detection as one critical capability of next-generation BI and analytics platforms.

However, automatically discovering interesting patterns from data remains an open research problem. First, there

SIGMOD'19, June 30-July 5, 2019, Amsterdam, Netherlands

ACM ISBN 978-1-4503-5643-5/19/06...$15.00
https://doi.org/10.1145/3299869.3314037

is a lack of unified and consistent formulation of "interesting patterns". A set of techniques [5][7][9][10] have been proposed to extract different types of interesting patterns from multi-dimensional data, such as anomalies or exceptions. However, these techniques focus on mining individual types of patterns; therefore, they are insufficient for facilitating comprehensive data analysis. While "facts taxonomy" [8] was proposed to categorize interesting patterns, it does not provide a unified formulation. Second, there is a lack of efficient mining frameworks that target general interesting patterns. The search space grows exponentially as the number of dimensions increases, and interesting patterns are hidden in unknown subsets of data.

We present *QuickInsights*, a novel technique for automatically discovering interesting patterns from multi-dimensional datasets. *QuickInsights* provides a unified formulation of interesting patterns, called *insights*, and a systematic mining framework to derive insights efficiently. Specifically, given a multi-dimensional dataset, an insight reflects something *interesting* on a specific *subject* in the data from certain *perspective*. We formulate an insight based on three key elements: *subject*, *perspective* and *interestingness*. Such formulation is able to unify different types of interesting patterns proposed in previous works [5][7][9][10]. Given the formulation, the mining framework of *QuickInsights* aims to automatically discover insights with quality and efficiency.

Quality challenge: Some insights may be easily inferred by users based on data schema information. They provide little information gain, thus are less interesting to users. E.g., an almost perfect linear correlation of two measures over years, where measure1 is sales in USD, and measure2 is sales in EUR (i.e., only differ by exchange rate) will become easily inferable to users. We try to avoid such *Easily Inferable Insights* (EII for short) to guarantee high-quality insight mining results. How to effectively detect and eliminate EIIs imposes challenges on insight mining.

Efficiency challenge: The search space of mining multi-dimensional dataset grows exponentially as the number of dimensions increases. Moreover, since *QuickInsights* is mostly used in interactive data exploration, it must output insights within a short time budget. To effectively utilize the time budget, we should try to first explore the "best" possible subsets of data where high-quality insights exist. In addition, insight evaluation always involves a lot of data aggregation queries against the database, which may further impact mining performance.

To address the quality challenge, we notice that EII is mainly caused by inter-dimensional dependency. Therefore, we conduct functional dependency checking of insight subjects, and implement an efficient algorithm to detect and eliminate EIIs caused by functional dependency. To address the efficiency challenge, we first employ a "best-first" search mechanism to prioritize insight evaluation tasks. Given a time budget, this mechanism tries to prioritize insight evaluation tasks, by estimating which task would result in a higher score before evaluation. We then employ a smart-batching mechanism to effectively reduce the number of queries by taking advantage of spatial locality across multiple related queries in data, thus improve query performance.

We conducted quantitative experiments on 447 real datasets to evaluate the effectiveness and efficiency of *QuickInsights*. We also performed qualitative user studies, which showed that the insights generated by *QuickInsights* are useful and valuable to both expert users and non-expert users. *QuickInsights* has been released in Microsoft Power BI [14] as a feature available to end users, which is recognized by Gartner as a basic form of smart data discovery [25]. We make the following contributions:

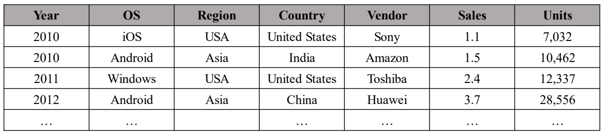
- We propose a unified formulation of interesting patterns, called *insights* on multi-dimensional dataset.
- We build an insight mining framework to achieve efficient insight mining performance using two key techniques: *best-first* search mechanism to prioritize insight evaluation tasks, and *smart query-grouping* to reduce the number of queries.
- We design an insight evaluation algorithm to eliminate *EIIs* to achieve high-quality insight results.
- We evaluate *QuickInsights* and verified its effectiveness and efficiency on discovering insights. *QuickInsights* is released in Microsoft Power BI.

2. INSIGHT MODELING

2.1 Data Model

Multi-dimensional data conceptually is organized in a tabular format that consists of a set of records, and each record is represented by a set of attributes (columns in the table). Table 1 shows some sample data from a multi-dimensional dataset about tablet sales. There are two types of columns in the table: dimensions and measures. Dimensions are used to group or filter records. The values of dimensions are either *categorical* (e.g., "Country") or *ordinal* (e.g., "Year"). Measures are numerical columns (e.g., "Sales") on which certain aggregations (e.g., SUM, AVG) can be performed. Formally, given a multi-dimensional dataset $\mathbb{R}(\mathcal{D}, \mathcal{M})$, where $\mathcal{D} = \{D_1, \dots D_d\}$ is the collection of dimensions and $\mathcal{M}$ is the collection of measures. Let $dom(D_i)$ be the domain of D_i.

Table 1. A sample of multi-dimensional data.



Year	OS	Region	Country	Vendor	Sales	Units
2010	iOS	USA	United States	Sony	1.1	7,032
2010	Android	Asia	India	Amazon	1.5	10,462
2011	Windows	USA	United States	Toshiba	2.4	12,337
2012	Android	Asia	China	Huawei	3.7	28,556
…	…		…	…	…	…

Subspace. A subspace is defined as a size-d collection of filters $s = \{s[1], \ldots, s[d]\}$, where $s[i] \in dom(D_i) \cup \{*\}$, and '*' refers to the "any" value. We hide the filters with star value ('*') for brevity. We call a subspace s with dimensionality $l \coloneqq |\{s[i] | s[i] \in s, s[i] \neq *\}|$. Each subspace associates with an aggregate value per each measure, e.g., {Country: China} is one subspace with $l = 1$, and its corresponding aggregation on measure *Sales* is aggregated by SUM. For conciseness, we denote {Country: China} as {China} for short.

Sibling group & breakdown. Given a subspace s and a dimension D_i, a sibling group is defined as $SG(s, D_i) = \{s' | s'[i] \neq *, s'[j] = s[j] \forall j \neq i\}$, i.e., a set of subspaces only differ in the values of $dom(D_i)$. In this setting, we call D_i the *breakdown* dimension (i.e., the group-by operation against a subspace), and we denote $s \oplus D_i \rightarrow SG(s, D_i)$ to indicate that sibling group $SG(s, D_i)$ is generated from subspace s by breaking down of D_i. For example, subspaces {2011, China}, …, {2016, China} form a sibling group because they only differ in the value of dimension *Year*, and *Year* is the breakdown dimension.

2.2 Insight Formulation

In the domain of multi-dimensional data analysis, an interesting pattern can generally be summarized as follows: it reflects something *interesting* on a specific *subject* of data from a certain *perspective.* We refer to such kinds of interesting pattern as *insight.* Subject scopes the content of an insight. Taking the trend insight in Figure 1 as an example, its subject includes the sibling group $SG(\{China\}, Year)$ and the measure Sales. Its aggregate values form a time series over years for trending analysis, which is the perspective of this insight. Its interestingness is reflected by "trending upwards rapidly and consistently". Below we describe *subject*, *perspective*, and *interestingness* of an insight accordingly.

2.2.1 Insight Subject

We define insight subject as:

Definition 1. $subject \coloneqq \{subspace(s), breakdown, measure(s)\}$

For example, the subjects of the two insights in Figure 1 are: {{*}, ServerName, CPU Usage}, and {{China}, Year, Sales}, respectively. Insight subject specifies the scope of content of an insight, and it corresponds to one or more sets of aggregate values, which can be used to quantify the interestingness. To facilitate intuitive understanding, let's map to visual charts. Each combination of {subspace, breakdown, measure} corresponds to a sibling group and their aggregate values on the measure. The values of the breakdown dimension can map to x-axis values; and the aggregate values can map to y-axis values; while the subspace can map to filter. For the cases with multiple subspaces or multiple measures, they can map to multiple series of y-axis values with the same x-axis.

Such a natural mapping to visual charts is an important advantage of Definition 1, given that insights are typically consumed via visual interfaces [8], thus enabling seamless integration with visual objects as the underlying object model. In addition, there are more advantages as follows. First, it is an abstraction that covers a wide range of subjects of specific "insights" in the literature. E.g., [9] automatically discovers insights with large deviation over a distribution, where the distribution can be properly modeled by Definition 1. Second, based on the feedback from several data science teams that we have closely engaged with in Microsoft, the insights derived from Definition 1 is satisfactory to facilitate their basic analytical needs.

2.2.2 Insight Types

We materialize different perspectives as different insight types. For instance, insight type "Outstanding#1" corresponds to the perspective of finding "the leading value that is outstandingly higher than the remaining values". Specifying insight type is essential for further quantifying insight interestingness. E.g., given the sales in China over years, the evaluation criteria are different for perspectives such as trend or seasonality.

We have developed 12 types of insights, corresponding to 12 different perspectives commonly adopted in practice, such as *Attribution, Change Point, Correlation, Outlier, Seasonality,* etc. Details are available on website [15]. The mining framework of *QuickInsights* is designed to be extensible, and configurable (see Section 3.1.3 for details) to support new insight types easily.

2.2.3 Insight Scoring

We quantify the "interestingness" of an insight by assigning an appropriate score to it. Intuitively, *interestingness* of an insight is judged by two factors. First, the subject of the insight should be non-trivial, so that the insight expresses something important, e.g., we would like insight subject to be a best-selling brand, or a category that has large market share rather than being neglectable. Second, aggregation results of the subject should exhibit significant differences against a baseline. We express the baseline as a statistical hypothesis, which reflects common situations formed up by majority of non-insights (i.e., aggregation results with uninteresting patterns). E.g., for correlation analysis, it is desirable to look for two time-series instances exhibiting correlation against null hypothesis $H_0\colon \rho = 0$. Such a null hypothesis reflects one common situation where two time-series instances are independent. In this paper, we term these two factors as *impact* and *significance*, respectively, and score an insight by combining them.

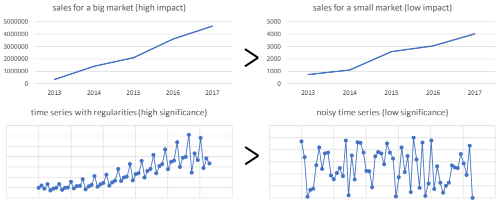


Figure 2. Illustration of impact and significance.

Impact. Impact reflects the importance of the subject of an insight against the entire dataset. It can be determined by the best possible perspective for promoting the insight regarding any "meaningful measures". Here we term these "meaningful measures" as *impact-measures*, and denote the value of impact on a specific impact-measure *i* as $impact_i(insight.subject)$ or just $impact_i$ for brevity. Figure 2 shows sales trends of two different markets when impact-measure is market share. The higher the market share the more important. $impact_i$ should hold anti-monotonic condition [16] , and should be normalized for fairness comparison across different impact-measures. Anti-monotonic is necessary because it is compliant with common sense: if the subject of insight A is a superset of the subject of insight B, then impact of A should be no less than impact of B. [11] provides calculations to accommodate anti-monotonic condition being held by various aggregations. The corresponding calculations are denoted as $MonoAggr_i(insight.subspace)$. Normalization is necessary for fairness comparison across impact-measures. Having these, we propose: $impact_i = \frac{MonoAggr_i(insight.subspace)}{MonoAggr_i(\{*\})}$. To avoid divide-by-zero, we restrict the impact-measures to be measures only containing strictly positive values. E.g., COUNT is a valid impact-measure; *Sales* or *Units* in Table 1 are also suitable impact-measures. Users can specify meaningful impact-measures aligned with their needs. Under this restriction, $impact_i$ is well-defined and bounded within [0, 1], and we define impact of an insight, which seeks the impact-measure that best promotes insight:

Definition 2. $impact = \max_i(impact_i)$

Lemma 1. Definition 2 satisfies anti-monotonic condition and is bounded between 0 and 1. (Proof is in Appendix).

Significance. Significance is evaluated on the aggregation values of the insight subject, and it is designed to reflect how significant the fact (i.e., the obtained aggregate values) against a baseline in a stochastic fashion. We express the baseline as an insight-type-dependent null hypothesis, which reflects common situations formed up by majority of non-insights and quantify insight significance by conducting significance-based hypothesis testing. The bellowing two charts in Figure 2 shows two different time series signals: the left one is more significant than the right one, because it contains certain regularities instead of pure noise.

More specifically, in the scenario of *QuickInsights*, without knowing further knowledge of user preferences, we propose baseline for each type of insight based on common sense. Such common sense should approximate the distribution of possible outcome which is uninteresting (i.e., trivial or less valuable for data analysis). E.g., to calculate significance of whether there exists a change point on a time series instance, a reasonable baseline is to assume the time series to be relatively stable, which is compliant with common sense (such time series provides no value on change point related analysis), and can be easily formalized as: $H_0\colon for\ 1 \le k \le N\colon p_\theta(y_k|y_{k-1}{\sim}y_1) = p_{\theta_0}(y_k|y_{k-1}{\sim}y_1)$, where p_{θ_0} is a fixed probability distribution [21]. The insight significance takes a value within [0, 1]. The closer the value to 1, the more significant the insight is. Detailed baseline setup and significance calculations are available at website [15].

Score. By combining the two factors together, we come up with the final score which quantifies the overall "interestingness" of an insight:

Definition 3. $score_t = f(impact) \cdot g_t(significance)$

Here the subscript *t* refers to a specific insight type, considering the significance calculation is insight type dependent. *f* and *g* are any non-negative, monotonic functions. Currently, we take the simplest form: $score_t = impact \cdot significance_t$

Definition 4 (Insight representation). With the above considerations, we represent an insight as a 5-tuple

$$insight \coloneqq \{subspace(s), breakdown, measure(s), type, score\}$$

3. INSIGHT MINING

3.1 Mining Framework

Overall, *QuickInsights* aims to achieve three design goals: (1) be a time-bounded mining procedure; (2) be portable to commodity query engines; (3) be extensible to adapt new types of insights.

Time-bounded mining procedure. The typical scenario of *QuickInsights* is one that targets interactive data exploration, thus it must output insights within a given limited time budget, e.g., 10 seconds. To effectively utilize the time budget, the mining procedure should try to explore the best possible subjects (i.e., combination of subspace and breakdown), where high-quality insights might exist. To discover insights, data queries and significance evaluations are performed by a set of *tasks*, where each task takes certain subspace(s) (and the corresponding impact of each subspace) and breakdown as input, and is responsible for evaluating certain types of insights that are applicable to the input parameters (e.g., time series related insights are evaluated when input breakdown dimension is ordinal). Therefore, a best-first prioritization of tasks is necessary (Section 3.1.1).

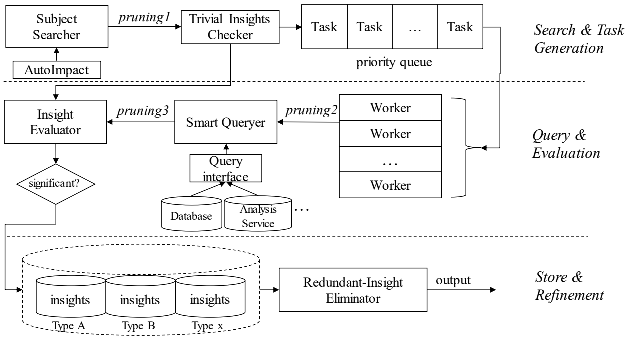


Figure 3. Overall workflow of *QuickInsights*

Portable to arbitrary query engines. As a general mining framework, *QuickInsights* should be portable to build upon arbitrary query engines such as SQL Databases, SQL Server Analysis Services, etc. where multi-dimensional datasets are typically stored. Thus, an abstracted and general query interface layer is necessary.

Extensible to adapt new types of insights. *QuickInsights* is designed to support new insight types easily. Therefore, we decouple the mining procedure into two parts: subject enumeration and insight's significance evaluation, only insight evaluation module is responsible for registering new insight types (Section 3.1.3).

Figure 3 depicts the overall workflow of *QuickInsights*. The workflow can be divided into three stages, "Search & Task Generation" (Stage 1), "Query & Evaluation" (Stage 2), and "Store and Refinement" (Stage 3). The first two stages are executed simultaneously in a parallel fashion within a time budget. Once the time exceeds the time budget, refinement is conducted in Stage 3 and then the qualified insights are output.

In Stage 1, the SubjectSearcher module tries to enumerate all possible subspaces. Each subspace is assigned with *impact* by using the AutoImpact module. Insight evaluation tasks are then generated by combining subspaces with any valid breakdowns that pass trivial-insight checks (by Functional-Dependency checker). The generated tasks are stored in a priority queue, to be executed in Stage 2. The tasks associated with higher impacts will be assigned higher priorities. In Stage 2, the tasks are computed in parallel by a set of dedicated worker threads. The computing of tasks consists of three steps. First, the task with highest priority from the queue is fetched by a worker thread; then data query is performed as the next step, by conducting aggregation over all measures, conditioned on the task parameters. Insight evaluation is conducted as the last step, where the discovered insights (i.e., significance exceeds certain threshold) are stored. Both Stage 1 and Stage 2 are executed within a time budget. Below are the details.

3.1.1 *Best-First Prioritization*

The generated tasks are stored in a priority queue, as depicted in Figure 3 to facilitate best-first prioritization. Recall that each task has three input parameters: subspace(s), breakdown and impact, and we use the impact as *priority* to prioritize different tasks. According to Definition 3, the score of insight is monotonic to both impact and significance, so without knowing the significance (since insight evaluation has not yet been done), impact is useful for prioritizing and pruning tasks.

3.1.2 *Query Abstraction*

To make *QuickInsights* portable for general systems, an abstracted query interface layer is necessary. Table 2 shows the query interface *AggregationQuery*, which builds a connection between the mining layer of *QuickInsights* and the data store. Thus, QuickInsights is portable as long as the underlying data store provides the implementation of *AggregationQuery*. A query via our query interface is semantically equivalent to a SQL query:

"*SELECT Aggr1(measure1), Aggr2(measure2), … GROUP BY breakdownDimension where filter = subspace*".

Note that the efficiency of QuickInsights mainly depends on the efficiency of underlying query engine. Microsoft Power BI team has supported our query API based upon Analysis Service. To further improve query performance by leveraging data locality, we introduce a pre-fetch mechanism and modify the above GROUP BY clause to:

"*GROUP BY expandingDimension, breakdownDimension*".

The aggregation results are packaged into a dictionary. Each item of the dictionary collects the result of each value in *expandingDimension*. Setting it to null disables pre-fetching. Table 3 shows two typical query examples and the corresponding results.

Table 2. Query Interface

```
/* aggregate one or more measures for a subspace, group-by a breakdown dimension.
If an expanding dimension is provided, also aggregates for the siblings of this
subspace based on the expanding dimension. */
Dictionary<BasicValue, Dictionary<Measure, AggrResult>> AggregationQuery(
    Subspace subspace,
    Dimension expandingDimension,
    Dimension breakdownDimension,
    Dictionary<Measure, AggrParams> params,
    OrderByType orderBy);
```

Table 3. Examples of query and aggregation result

subspace	expanding	breakdown	params	orderBy	Aggregated result
{China}	null	Year	{Sales, SUM}, {Units, SUM}	Ascend	{China, [Sales, (2009:1.3) (…) (2016:12.3)], [Units, (2009: 6,403) (…) (2016:13,432)]}
{China}	Country	Year	{Sales, SUM}	Ascend	{China, [Sales, (2009:1.3) (…) (2016:12.3)]} {USA, [Sales, (2009:2.7) (…) (2016:11.8)]}…

3.1.3 *Extensibility*

QuickInsights is designed to be extensible to support new types of insights easily. The extensibility of *QuickInsights* largely relies on the unified definition of insights (Definition 4). Specifically, since each insight subject is formulated as $\{subspace(s), breakdown, measure(s)\}$, thus the aggregation results of an insight subject can be represented by a common data structure, which can be reused for any new insight type. An example of adding a new insight type is depicted in Appendix.

3.1.4 *Pruning*

As depicted in Figure 3, we applied three pruning criteria (pruning1, 2, 3) to boost performance: pruning1 prunes out significant portion of search space, and pruning2 and pruning3 reduce the cost of insight evaluation.

pruning1: We prune out any insights with impact smaller than a given threshold. An insight with impact below the threshold becomes less important and thus less interesting, so we adopt pruning1 to eliminate unimportant tasks. Furthermore, considering the anti-monotonic condition of impact (Lemma 1), any descendant subspaces can also be discarded from the SubjectSearcher module safely. In current implementation, we set the threshold to 0.01.

pruning2: For each insight type, we use a size-k buffer to keep the top-k scored insights. Considering $score_t = impact \cdot significance_t < impact$ (because significance is bounded within 0 and 1), so if impact of current insight candidate is already smaller than the score of k*th* insight, its further evaluation is saved. Furthermore, since each task knows what types of insight it needs to evaluate, if insight evaluation can be pruned on all the needed types, then data query can be saved and the task is discarded.

pruning3: When a sibling group contains only one subspace, further insight evaluation becomes trivial (because this subspace is identical to its parent subspace and thus implies duplication), hence unnecessary. So after data query, if there is only one item among the sibling group, we avoid further insight evaluation.

3.2 Easily Inferable Insights Elimination

We illustrate how to improve insight quality by detecting and eliminating EIIs (i.e., Easily Inferable Insights) incurred by functional dependency (FD in short).

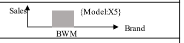
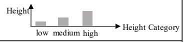
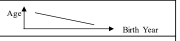
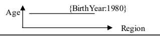
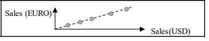
3.2.1 *FD Induced EII*

Definition 5 (functional dependency). A functional dependency FD: X → Y means that the values of Y are determined by the values of X, where X and Y are two sets of columns (i.e., dimensions or measures) [12].

FD is a commonly existing relationship in multi-dimensional data, e.g., in Table 1, Country → Region. FDs reflect certain hierarchical structure or consistent relationship across columns.

Definition 6 (FD of insight subject). We pick all the columns that appear in an insight subject as $Col \coloneqq \{s_1, \ldots, s_p, d, m_1, \ldots, m_q\}$, where $s_1 \sim s_p$ are the dimensions appearing in subspace(s), d is breakdown, and $m_1 \sim m_q$ are q measures. If $\exists\, X \subset Col, Y \subset Col, X \cap Y = \emptyset, s.t. X \rightarrow Y$, we say $X \rightarrow Y$ is a FD of this insight subject.

Table 4. Taxonomy of trivial insights

ID	Form of Functional-Dependency	Trivial insight description	Example
ID1	$\{s_1, \ldots, s_p\} \rightarrow d$	Only one item in sibling group	
ID2	$\{m_1, \ldots, m_q\} \rightarrow d$	Fixed x-y axis relationship	
ID3	$d \rightarrow \{m_1, \ldots, m_q\}$	Fixed x-y axis relationship	
ID4	$\{s_1, \ldots, s_p\} \rightarrow \{m_1, \ldots, m_q\}$	Flat line	
ID5	$\{m_1, \ldots, m_i\} \rightarrow m_j$	Fixed x-y axis relationship	

Based on Definition 6, we notice that FD of insight subject would bring up EIIs.

Definition 7 (FD induced EII). An insight is called an FD induced EII (or EII in short) if its aggregate values exhibit pre-determined relationships thus providing trivially useful information for the purpose of data analysis.

We carefully inspect all possible FDs incurred in insight subject, and come up to five forms of FD that would induce EIIs, as shown in Table 4 (ID1 ~ ID5). The details of how ID1~5 induce EII are shown in Appendix.

3.2.2 *Efficient FD Checking*

Given an insight candidate, we need to check if there exists FDs to satisfy any of ID1~5 in Table 4 thus to avoid further insight evaluation. The checking can be generalized as determining whether $\{d_1 \sim d_i\} \rightarrow d_j$ is held or not. On the other hand, such determination requires knowing the FDs that are globally held in a given dataset, and such FDs can be obtained from data schema or can be pre-calculated using FD mining techniques such as [13]. Thus, we formulate the problem as:

Problem 1 (checking functional dependency). Given a set of FDs $\{X_1 \rightarrow Y_1\}, \dots, \{X_t \rightarrow Y_t\}$, check if $\{d_1 \sim d_i\} \rightarrow d_j$ is held or not.

This problem can be solved by leveraging two axioms in the field of FD theory: *Reflexivity* and *Transitivity* [12]. Roughly, if $d_j \in \{d_1 \sim d_i\}$, the $\{d_1 \sim d_i\} \rightarrow d_j$ is true (Reflexivity). Otherwise, find $X = \bigcup_i \{X_i | d_j \in Y_i\}$, and check if $\{d_1 \sim d_i\} \rightarrow X$ (Transitivity). This process repeats recursively until an empty set is reached. The pseudo code of an efficient algorithm (IsDependent) of FD checking is shown in Appendix due to page limit.

Lemma 2. Time complexity of IsDependent is $O(lD) \ll O(D^2)$. Details of the proof are available in Appendix.

Here D is the number of columns, and $l = \max_i |X_i|$, where $|X_i|$ refers to the cardinality of a set X_i. In general, the FDs obtained from data schema describes FD relationship between a small set of dimensions, thus $l \ll d$.

3.3 Batched Query & Cache

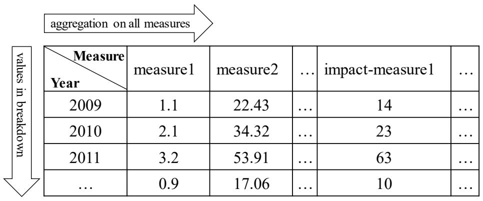
Data query occupies the majority of computational cost of *QuickInsights*. Next, we illustrate our considerations and approach on query optimization to significantly save the computational cost.

3.3.1 Caching

As depicted in Figure 3, the Subject-Searcher module, the AutoImpact module, and Tasks issue data queries. Subject-Searcher uses queries to enumerate subspaces, AutoImpact needs query results on impact-measures to assign impact to each subspace, and Tasks issue queries for insight evaluation. These modules would generate duplicate queries, e.g., query {China}⨁Year can be used for insight evaluation, while Subject-Searcher also needs resultant subspaces for search space exploration, and the impact of resultant subspaces is obtained from AutoImpact by aggregating all impact-measures. Thus, cache mechanism is needed, and the cache unit needs to be designed to facilitate the requirements of these modules, as depicted in Figure 4. Here the cache unit is 2-dimensional aggregation results grouped-by breakdown, and across all measures (both insight measures and impact-measures), and the corresponding lookup key for each cache unit is indicated by $s \oplus D$. Such granularity is necessary for the needs of all the modules.

3.3.2 Smart Batching

A typical multi-dimensional dataset contains a huge number of $s \oplus D$ combinations, and requires a large number of data queries, which would lead to significant performance impact. On the other hand, by inspecting the issued queries from *QuickInsights*, we find that the generated subspaces exhibit strong relationships with each other, which provides opportunity to reduce the number of queries.


aggregation on all measures
values in breakdown

Measure / Year	measure1	measure2	…	impact-measure1	…
2009	1.1	22.43	…	14	…
2010	2.1	34.32	…	23	…
2011	3.2	53.91	…	63	…
…	0.9	17.06	…	10	…

Figure 4. Example of cache unit: breakdown⊗measure

Definition 9 (level-2 sibling group). A set of subspaces form a level-2 sibling group if they can be generated by a level-2 group-by: $s \oplus D_1 \oplus D_2$.

E.g., when we have the following three query requests: {China}⨁Year, {USA}⨁Year, {India}⨁Year, they can be covered by a level-2 group-by: {*}⨁Country⨁Year, thus the corresponding subspaces belong to a level-2 sibling group. Batching these three queries together would take advantage of spatial locality across multiple related queries in data, thus improving query performance.

However, one problem arises from batching: higher level group-by would introduce additional aggregation results which may never be used. In the mentioned example, {*}⨁Country⨁Year obtains the necessary results for the three requests, but it also obtains results for all countries besides China, USA, and India. In addition, considering *QuickInsights* typically runs within a time budget, only a portion of a whole search space can be inspected. Therefore, we prefer conducting a batched query *on-demand* rather than exhaustive pre-fetching in the beginning to mitigate the issue of querying useless results.

As depicted in Table 2, query API of *QuickInsights* considers an expanding dimension as an additional group-by for batching purpose. We notice that using the latest breakdown as an expanding dimension can fully leverage spatial locality, and pre-fetched results can also be effectively utilized for later tasks.

From another perspective, column cardinality together with pruning1 will affect the utility of batched query. For example, if there are >1000 distinct values in City for {China}, the batched query by expanding on City generates 1000 subspaces, but at most 100 subspaces has impact > 0.01 due to the Pigeonhole Principle, thus most (>90%) prefetched subspaces are useless which makes this query very ineffective. Therefore, when the number of subspaces generated by a breakdown exceeds a threshold, we don't use it for expanding dimension.

With these considerations, we name our approach *smart batching.* The approach aims to reduce number of data queries, while the pre-fetched results are effectively utilized. Considering page limit, we put the examples, pseudocode of *QuickInsights*' query logic with both cache and smart batching in Appendix.

4. EVALUATION

We evaluate the effectiveness and efficiency of *QuickInsights* quantitatively on real datasets (Section 4.1). We further evaluate the usefulness of *QuickInsights* in assisting data analysis through two user studies on expert users and non-expert users, respectively (Section 4.2).

4.1 Evaluation on Real Datasets

4.1.1 Setup

Datasets. We evaluate *QuickInsights* on 447 real datasets. These multi-dimensional datasets are collected with assistance from partnering Microsoft teams. The datasets cover various domains such as sales, weather, market, healthcare, etc. Their scales are quite variant, with the size ranging from 8.8KB to 386.2MB, and the dimensionality varying from tens to hundreds. Some of the datasets are available on our website [26].

Environment. All experiments are conducted on a machine with 3.6GHz Intel Core i7-4790 processor, and 16GB RAM. *QuickInsights* is deployed upon a SQL Server Analysis Service (SQL Server 2016 RTM, version: 13.0.1601.5, Tabular Mode).

Configuration. We set the configuration of *QuickInsights* as follows: #worker threads = 8; maximum dimensionality of explored subspace = 2 since output insights with high-dimensionality subspaces are less informative for common usages; we set COUNT as impact-measure for all datasets for simplicity, because setting different impact-measure has little affect to efficiency evaluation.

4.1.2 Design

We aim to evaluate *QuickInsights* from three perspectives: overall effectiveness, effectiveness for EIIs elimination and mining efficiency. To make the experimental results measurable, we set *golden set* of each dataset as the obtained insights from *QuickInsights* with time budget set to ∞, denoted as O_i, where i indicates the index of a dataset. More specifically, we set the number k of top-k buffers (as depicted in Figure 3, we maintain a top-k buffer for each type of insight) to 10, and O_i is the union of insights from all the buffers after insight mining is finished with an unbounded time budget.

Overall effectiveness. To evaluate the overall effectiveness of *QuickInsights*, we define metric $cov_i(t) = \frac{|O_i(t) \cap O_i|}{|O_i|}$, where $O_i(t)$ is the set of output insights when time budget is set to *t*. Thus $cov_i(t)$ is the coverage of "good" insights of $O_i(t)$.

Effectiveness of EIIs elimination. To improve the quality of output insights, *QuickInsights* exploits the FD checker to avoid yielding EIIs. To demonstrate the effectiveness of such improvement, we assess the insights mined when the FD checker is turned off.

FD checker enabled vs. disabled. Among the whole datasets, there are 218 ones with input FDs according to the data schema. Evaluation of the FD checker is therefore conducted on this subset because the other datasets have no effect. We compare the results when the FD checker is disabled to the *golden set* by two metrics:

$$cov\_FD_i(t) = \frac{|O\_FD_i(t) \cap O_i|}{|O_i|}, fp\_FD_i(t) = \frac{|O\_FD_i(t) \backslash O_i|}{|O\_FD_i(t)|}$$

where $O\_FD_i(t)$ is the set of output insights when the FD checker is disabled. $cov\_FD_i(t)$ reflects the coverage of insights when the FD checker is disabled. $fp\_FD_i(t)$ reflects the estimated ratio of trivial insights in $O\_FD_i(t)$. This is because $O\_FD_i(t) \backslash O_i$ indicates the set of insights being eliminated by golden set, which must be FD induced EIIs.

Mining efficiency. *QuickInsights* exploits best-first prioritization and smart-batching to boost mining performance. Thus, the evaluation of mining efficiency mainly is conducted on these two techniques. We propose the below evaluation metrics.

Best-first prioritization enabled vs. disabled. We implement a priority queue (by using impact as priority) to prioritize insight evaluation tasks. To assess the effectiveness of such a strategy, we compare the coverage of output insights by replacing the priority queue with a FIFO queue. The metric is defined as: $cov\_Priority_i(t) = \frac{|O\_Priority_i(t) \cap O_i|}{|O_i|}$, where $O\_Priority_i(t)$ is the set of output insights when best-first prioritization is disabled.

Smart-batching enabled vs. disabled. We assess the efficiency improvement of smart-batching from two aspects: coverage when smart-batching is disabled, and the utilization of the cache:

$$cov\_Batching_i(t) = \frac{|O\_Batching_i(t) \cap O_i|}{|O_i|}, cache\_utility_i(t) = \frac{|Hits(t)|}{|Cached(t)|}$$

where $O\_Batching_i(t)$ is the set of output insights when smart-batching is disabled, $Cached(t)$ is the set of total cached items (See Figure 4 for definition of cache unit), and $Hits(t)$ is the set of cached items that are utilized.

In our experiment design, we vary time budget t from 5 seconds to 25 seconds, and so each evaluation metric generates a curve with respect to time budget on a specific dataset. Evaluation results are analyzed by averaging on all datasets, and via comparison between different curves.

4.1.3 Results

Below are the results of our experiments.

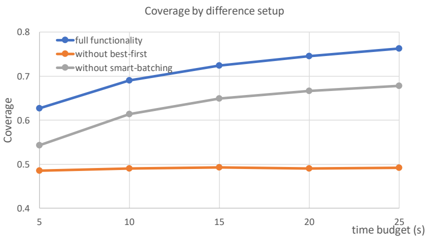


Figure 5. Average coverage by varying time budgets

Result of overall effectiveness. The curve at top of Figure 5 presents the coverage of "good" insights (i.e., *golden set* insights) mined in different time budgets. Each data point is an average of the coverage over totally 447 datasets. The coverage increases as more time budget is given, which is reasonable since more search spaces can be explored and evaluated, and more hard-to-find insights can be discovered. Moreover, the coverage ranges from 0.6 to 0.8. For example, when the time budget is set to 5 seconds, the coverage is 0.63, which indicates that even when the response time is very quick, more than 60% of the insights returned by *QuickInsights* are truly top-scoring ones.

Results of FD checker enabled vs. disabled. Table 5 depicts the average coverage of output insights when the FD checker is disabled (third row), and ratio of the EIIs (fourth row). Since this evaluation is conducted on 218 datasets that have FDs as input, we also list the corresponding coverage when the FD is enabled (second row) for comparison. As shown in Table 5, coverage of good insights decreased about 3% consistently when the FD checker is disabled. Moreover, value of $fp\_FD(t)$ is around 25% when the FD checker is disabled, i.e., when users inspect the output insights, one out of four will be easily inferred. So disabling FD checker will significantly decrease the user experience of *QuickInsights*.

Table 5. Results of disabling the FD checker

Time budget (s)	5	10	15	20	25
cov	0.49	0.57	0.62	0.65	0.67
$cov\_FD(t)$	0.46	0.55	0.60	0.62	0.64
$fp\_FD(t)$	0.28	0.25	0.25	0.23	0.22

Results of best-first prioritization enabled vs. disabled. As depicted in Figure 5, the bottom curve presents the coverage of good insights when best-first prioritization is disabled. Compared to the top curve (with best-first prioritization enabled), we can see that the gain of the best-first prioritization is significant. Without such a mechanism, the insight mining procedure seems to be trapped into massively worthless search spaces, making the curve rather flat. To increase coverage to around 63%, it needs much more time than 25 seconds, while the same coverage is achieved only in 5 seconds if best-first prioritization is enabled. We can see that the performance gain by using best-first prioritization is huge.

Results of smart-batching enabled vs. disabled. As depicted in Figure 5, the middle curve presents the coverage of good insights when smart-batching is disabled. Compared to the top curve (with smart-batching enabled), smart-batching contributes to about 10% coverage increase. From a performance perspective, it takes about 15 seconds to achieve 63% coverage (the dashed line in Figure 5) when smart-batching is disabled, which is about **three** times slower than when smart-batching is enabled.

Table 6. Cache utilization ratio

Time budget (s)	5	10	15	20	25
cache_utility	38%	41%	43%	44%	44%

Table 6 illustrates the cache utilization with varying time budgets, which reflects how many pre-fetched queries are reused in further insight evaluation. We can see that the ratio of utilization is relatively stable at 40%. The ratio is impacted by the near-timeout batched queries, which generate lots of unused cached items.

4.2 User Study

We conduct user studies to understand usefulness of insights generated by *QuickInsights*.

4.2.1 Methodology

QuickInsights is designed to serve both expert users and non-expert users. The usage scenarios and requirements vary among different user groups, e.g., expert users would like *QuickInsights* to aid their further data analysis and decision making, while non-expert users would want to gain a better understanding of data. Thus, we conducted two user studies for expert users and non-expert users.

User study for expert users. We invite six participants from three business groups in Microsoft to participate in this user study: HR (Human Resource), IT and UR (University Relationship). In each group, we select two data analysts whose daily work is data analysis.

For each group, we ask the participants to provide one dataset of their own, since users would provide reasonable feedback on the datasets that they care about. The datasets are required to be stored in Microsoft Excel spreadsheets, in the form of multi-dimensional table. In addition, we require that the datasets have different levels of familiarity to the corresponding groups. Specifically, HR participants provide a recently-conducted survey dataset for which they have no idea about the content (Not familiar), while UR participants give us a dataset which they have already conducted comprehensive analysis (Very familiar), and IT participants give us a new dataset but they have conducted analysis on similar datasets before (Moderate). Table 7 lists the information of the three datasets.

Table 7. Datasets for user study on expert users

Dataset	#row	#col	Familiarity	Description
HR_data	351	10	Not familiar	Internal survey results on a specific HR service
IT_data	353,686	9	Moderate	GPU usage data of servers, collected every 10 minutes in one month
UR_data	1202	14	Very familiar	Records of hired interns in recent 4 years

We provide a questionnaire for each group, which contains 15 insights randomly selected among the insights obtained by running *QuickInsights* on the corresponding dataset. For each insight, we design three questions for scoring:

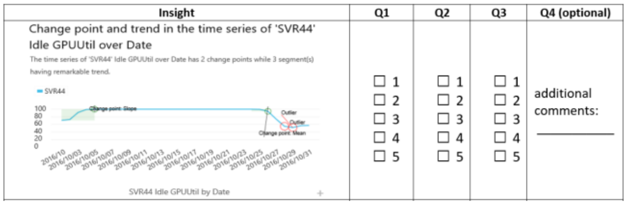
Q1: *How interesting do you feel of this insight*?

Q2: *How helpful is this insight for you to understand the data characteristic, such as distribution, anomaly or correlation, etc.*?

Q3: *To what extent do you feel interested to take follow-up actions, such as sharing with others, pinning to a dashboard, or conducting drill-down analysis*?

Specifically, Q1 targets obtain an overall impression of the insight from users; Q2 is designed to evaluate whether the insight is helpful for better data understanding or not; and Q3 is used to evaluate the actionability of the insight. Participants are asked to answer each question on a 5-point Likert Scale from “the least interesting/helpful” (1) to “the most interesting/helpful” (5). In addition, we allow users to provide free-text comments on each insight. We provide a text description along with a chart to represent each insight. Figure 6 shows a snapshot of an example insight and the corresponding questions.

Our user study is conducted by interviewing the three groups separately. Each session consists of three stages. In the first stage, each participant briefly describes his/her experience and the role of data analysis, and we introduce *QuickInsights* and the process of the user study. We also educate them how to interpret an insight from its text description and visual representation. In the second stage, participants assign scores to the questions for each insight. They are encouraged to provide additional comments as well. In the last stage, we ask participants about their overall feedback, and whether they would use *QuickInsights* for their analytical tasks. Each session lasts about one hour on average.



Insight	Q1	Q2	Q3	Q4 (optional)
Change point and trend in the time series of 'SVR44' Idle GPUUtil over Date The time series of 'SVR44' Idle GPUUtil over Date has 2 change points while 3 segment(s) having remarkable trend.	□ 1 □ 2 □ 3 □ 4 □ 5	□ 1 □ 2 □ 3 □ 4 □ 5	□ 1 □ 2 □ 3 □ 4 □ 5	additional comments: ____________

Figure 6. Example of questions for a ChangePoint insight

User study for non-expert users. We invite 30 participants (18 males) to participate in this user study. The participants are employees or interns from Microsoft. They have certain data analysis needs in daily work, but none of them are professional data analysts. To minimize potential bias, we select the participants with diverse roles and experiences. Detailed user profiles are shown in Appendix.

Table 8. Datasets for user study on non-expert users (275/5 means 275 rows, 5 columns)

Dataset	Schema	data scale	Description
Movie	Snowflake schema with 6 tables	65 columns, largest table has >70,000 rows	Worldwide movie sales from 1985~2016
CarSales	Single table	275/5	Car sales of different Brands, Models, etc. in past years
Emission	Single table	41,156/7	The emission of $CO_2/SO_2/NO_x$ in past 25 years in USA
Census	Single table	90/6	A census dataset mainly focuses on marriage status

Since non-expert users normally do not have dedicated analytical tasks, we select four datasets from public domains, which are common, and easy-to-understand. Table 8 lists the information of these datasets. We generate insights from these datasets via Power BI (thus under same configuration) and present them to users. The study design is an easier version compared with the user study for expert users. Specifically, the questionnaire contains 10 insights randomly selected from the results of running *QuickInsights* on the corresponding dataset. For each insight, we only ask the participants Q1 and Q2 but discard Q3, since the typical scenario for non-expert users is knowledge discovery and data understanding.

4.2.2 Key Findings

We identified five key findings from the two user studies.

***Finding 1: QuickInsights demonstrates its usefulness for general data analysis for both two types of users*.**

The expert users provided positive feedback on the overall satisfaction of *QuickInsights*. All three groups agreed that *QuickInsights* provides valuable information to aid their analytical tasks. In addition, some participants even provided “out-of-scope” feedback, such as improvements of visualization design, feature request of insight sharing, etc. This finding also indicates the effectiveness of our scoring function, since the insights in user study are the ones with highest scores.

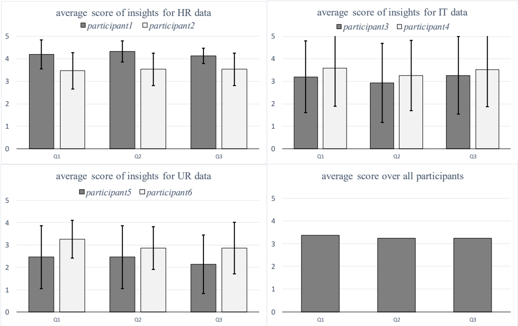


Figure 7. Statistics of scores from expert users

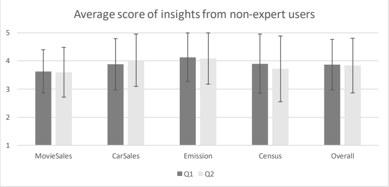


Figure 8. Statistics of scores from non-expert users

Figure 7 depicts the statistics of the scores from all expert participants. The error-bar indicates the standard deviation across 15 insights. The top-left chart illustrates the scores from HR participants. The average score on three questions are high and stable. The top-right chart illustrates the scores from IT participants, which has the largest deviation compared to HR or UR. In fact, the scores for most insights are either close to 1 (least interesting) or close to 5 (most interesting). The IT participants patiently provided comments on the insights with extreme scores, from which we learned that the IT analysts have very specific analytical tasks. Therefore, the insights are either valuable or less useful. The bottom-left chart is the scores from the UR group. The average score given by UR participants is the lowest compared to the scores from HR and IT participants. Based on feedback, the major reason is that they are very familiar with the dataset, thus most of the lower-scored insights are compliant with their prior knowledge. These observations are expected since the typical *QuickInsights* scenario targets users who are not familiar with dataset.

As shown in Figure 8, non-expert participants also provide very positive feedback on the overall satisfaction of *QuickInsights*. In addition, 11 out of 30 non-expert participants wrote down additional feedback, and quite a few pointed out that *QuickInsights* is really helpful on knowledge discovery.

Finding 2: Certain insight types would be favored for some domain-specific analysis tasks. We obtained this finding from the user study with expert users. One typical task of the IT group is to monitor GPU usage of various service jobs running over multiple servers, to detect which servers are overloaded (with high GPU usage) or idle (with low GPU usage), and reallocate jobs accordingly. The insight (with *ChangePoint* type) shown in Figure 6 is valuable (with a score 5 for all three questions) to them, since it discovered Server44 kept being idle for >20 days in October, which indicated some unknown service issues. Moreover, the *Seasonality* insights are especially interesting to them. *QuickInsights* discovered GPU usage for a specific GPU Model exhibits strong seasonality pattern with period equals to 24 hours. Thus, the IT users would want to take follow-up actions to see which periods within a day had low GPU usage, so that additional service jobs can potentially be scheduled during such a period. However, any transient spikes of GPU usage (regarding to the *Outlier* insight) are uninteresting since they are not indicators of workload. One possible solution is to assign insight type-dependent weights, so that the insight types with higher weights have more chance to be mined, and with higher score.

Finding 3: Insight subjects with certain structure would be less meaningful for some domain-specific analysis tasks. Specifically, certain dimensions, measures or combination are trivial. This finding emerged based on further feedback from the IT participants in the user study with expert users. For insights with a score equal to 1, typically their combinations of $s \oplus D$ are less meaningful to users. E.g., some insights concern a specific GPU Model (one dimension) breakdown by different GPU card slots (another dimension), which makes no sense since this is a fixed hardware configuration. *QuickInsights* should take this information into account, to avoid unnecessary data queries and insight evaluation.

Finding 4: Prior knowledge is valuable for improving insight score calculation. In the user study on expert users, the UR participants shared with us their thoughts during the interview. Since they are very familiar with the dataset, most of the insights are compliant with their prior knowledge, making them less interesting. For example, “Computer Science is the major for most hired interns” is mined by *QuickInsights* as an insight, but it is not surprising to them. In contrast, the HR participants claim that similar insights are helpful for their understanding of the survey data. Since they were not familiar with the content of the dataset, no prior knowledge was built before inspecting the insights. Since *QuickInsights* provides a general mining framework, so we can incorporate prior knowledge in via customized significance calculation.

Finding5: Visualization and natural language description are important to convey insights. Based on the free-text comments, most of the negative feedback is about confusion on either the visual charts or the text descriptions. Comparing with expert participants, non-expert participants are more often be confused by the charts or text descriptions. E.g., "the outlier does not seem so significant", or "what does *repeat pattern* mean?". We believe that future work from visualization and NLP communities could be very helpful and important to better represent and convey insights to non-expert users.

5. RELATED WORK

Pattern mining on multi-dimensional data. There exists lots of work in the literature which target mining various types of interesting patterns from multi-dimensional data. Sarawagi et al. [5] aim to find exceptions in OLAP data cubes. Wu et al. [7] propose promotion analysis for business intelligence, which discovers highly ranked subspaces associated with a given promotion object. Vartak et al. [9] focus on recommending high-deviation patterns via visualization. Chen et al. [10] investigate methods for multi-dimensional regression analysis of time series stream data. Their approach can be used to efficiently detect trends or outliers from multi-dimensional data. Palpanas et al. [18] provide answers to queries and find interesting cells in a data cube by the principal of maximum-entropy. Compared to these works, we attempt to propose a unified formulation of various types of interesting pattern as insights and conduct efficient insight mining via a general and extensible mining framework. Chen et al. [8] build a fact taxonomy of interesting patterns from visual perspective. All the facts can be formulated by the definition of insights.

Interestingness measures for data mining. Silberschatz et al. [19] advocate using unexpectedness to measure the interestingness of a pattern. Unexpectedness patterns are interesting because they exhibit contrary to common knowledge and may suggest certain perspectives of data that require further analysis. This idea is conceptually compliant with our formulation of insight significance. In addition, we propose using *impact* to express the importance of a pattern, which is also a key factor contributing to the interestingness measure. Ceng et al. [20] identify 9 criteria to determine whether a pattern is interesting or not, where *coverage* and *surprisingness* are analogical to the *impact* and *significance* of *QuickInsights*. Coverage is a specific implementation of impact when COUNT is adopted as impact-measure. Tang et al. [27] propose composite extractors for discovering latent yet interesting knowledge that can be derived by higher-order calculations. *QuickInsights* can incorporate composite extractors by calculating impact and significance based on the results of composite extractors.

OLAP and cubing. The data cube modeling has been a mature area to facilitate exploratory data analysis with lots of work such as Colliat [17], and Gray et al. [23]. Instead of pre-constructing data cubes, *QuickInsights* adopts a more economical way by on-demand querying and caching. Such an approach can avoid generating too many cubes which have no chance to be used for insight evaluation, and the query performance can be further improved via smart-batching, which is guided based on the subject searching mechanism of *QuickInsights*.

Visualization recommendation. There has been much work [9][22][32][33][34][36][37][41][42][43] that aims to facilitate rapid visual data exploration by automatically recommending visualizations. Some recommenders, such as APT [32], SAGE [33] and Show Me [34], focus on suggestions of visual encodings. More recent work [9][22][36][37] also suggest what data to visualize. They might rank visualization candidates based on various statistical analysis to promote the visualizations with interesting patterns [35]. E.g., Voyager [36][37] suggests visualizations based on statistical properties. Some systems are designed for specific tasks and patterns. Profiler [39] finds anomalies. SeeDB [9] identifies charts that are largely deviated from a given reference. Zenvisage [22] targets charts that are similar to a given input. Some novel visual data exploration tools (e.g., Foresight [29], Voder [38], DataSite [28]) are developed based on automatic insights and visualizations. In comparison, *QuickInsights* provides a unified formulation of interesting patterns, and developed a systematic insight mining framework to automatically mine insights from data. *QuickInsights* can be leveraged by visualization recommendation systems to produce insightful visualizations that convey interesting data patterns.

6. CONCLUSION

We present a novel technique *QuickInsights* to quickly and automatically discover insights from multi-dimensional data. *QuickInsights* proposes a systematic formulation of interesting patterns in terms of *insights* and conducts efficient insight mining to discover high-quality insights. *QuickInsights* has been released as a feature of Microsoft Power BI.

Acknowledgement. We thank our partners in Microsoft Power BI team for collaboration and system integration.

REFERENCES

[1] J. Han, M. Kamber and Jian Pei. Data Mining: Concepts and Techniques. Morgan Kaufmann Publishers, 2011.

[2] D. A. Keim. Information Visualization and Visual Data Mining. TVCG, 2002.

[3] U. Fayyad, G. P. Shapiro, and P. Smyth. From Data Mining to Knowledge Discovery: An Overview. In advances in Knowledge Discovery and Data Mining, 1996.

[4] R. Amar, J. Eagan, and J. T. Stasko. Low-level Components of Analytic Activity in Information Visualization. InfoVis,'05

[5] S. Sarawagi, R. Agrawal, and N. Megiddo. Discovery-Driven Exploration of OLAP Data Cubes. In EDBT, pages 168-182, 1998.

[6] S. Sarawagi. Explaining Differences in Multi-Dimensional Aggregates. In VLDB, pages 42-53, 1999.

[7] T. Wu, D. Xin, Q. Mei, and J. Han. Promotion Analysis in Multi-dimensional Space. In VLDB, 2009.

[8] Y. Chen, J. Yang, and W. Ribarsky. Toward Effective Insight Management in Visual Analytics Systems. IEEE Pacific Visualization Symposium, 2009.

[9] M. Vartak, S. Rahman, S. Madden, A. Parameswaran, and N. Polyzotis. Seedb: Efficient Data-driven Visualization Recommendations to Support Visual Analytics. In VLDB, 2015.

[10] Y. Chen, G. Dong, J. Han, B. W. Wah and J. Wang. Multi-Dimensional Regression Analysis of Time-Series Data Streams. VLDB, 2002.

[11] Jiawei Han, Jian Pei, Guozhu Dong, and Ke Wang. Efficient Computation of Iceberg Cubes with Complex Measures. SIGMOD, 1-12, 2001.

[12] M. Y. Vardi. Fundamentals of dependency theory. In E. Borger, editor, Trends in Theoretical Computer Science, 171-224, 1987.

[13] H. Yao, H. J. Hamilton. Mining Functional Dependencies from Data. DMKD, 197-219, 2008.

[14] https://powerbi.microsoft.com/en-us/blog/announcing-power-bi-integration-with-cortana-and-new-ways-to-quickly-find-insights-in-your-data/.

[15] QuickInsights. https://www.microsoft.com/en-us/research/project/quickinsights/

[16] R. Ng, L. Lakshmanan, J. Han, and A. Pang. Exploratory Mining and Pruning Optimization of Constrained Association rules. SIGMOD'98

[17] George Colliat. OLAP, Relational, and Multidimensional Database Systems. Technical report, CA, 1995.

[18] T. Palpanas and N. Koudas. Entropy based Approximate Querying and Exploration of Data Cubes. SSDBM, 2001.

[19] A. Silberschatz and A. Tuzhilin. What Makes Patterns Interesting in Knowledge Discovery Systems. TKDE, 1996.

[20] L. Ceng, and H. J. Hamilton. Interestingness Measures for Data Mining: A Survey. ACM Computing Surveys, 2006.

[21] M. Basseville, and I. V. Nikiforov. Detection of Abrupt Changes: Theory and Application. Prentice-Hall, 1993.

[22] T. Siddiqui, A. Kim, J. Lee, K. Karrie and A. Parameswaran. Effortless Data Exploration with zenvisage: An Expressive and Interactive Visual Analytics System. VLDB, 2017.

[23] 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. DMKD, 1997.

[24] E. K. Choe, B. Lee, and M. C. Schraefel. Characterizing Visualization Insights from Quantified Selfers' Personal Data Presentations. IEEE Computer Graphics and Applications, Volume 35, Issue 4, 2015.

[25] Gartner. Magic Quadrant for Business Intelligence and Analytics Platforms. Feb 2017. https://www.gartner.com/doc/reprints?id=1-3TYE0CD&ct=170221&st=sb

[26] https://docs.microsoft.com/en-us/power-bi/sample-datasets

[27] B. Tang, S. Han, M. L. Yiu, R. Ding, and D. Zhang. Extracting Top-k Insights from Multi-dimensional Data. SIGMOD, 2017.

[28] Z. Cui, S. K. Badam, A. Yalcin, and N. Elmqvist. Datasite: Proactive Visual Data Exploration with Computation of Insight-based Recommendations. arXiv:1802.08621, 2018

[29] C. Demiralp, P. J. Hass, S. Parthasarathy, and T. Pedapati. Foresight: Recommending Visual Insights. VLDB, 2017.

[30] C. Stolte, D. Tang, and P. Hanrahan. Polaris: A System for Query, Analysis, and Visualization of Multidimensional Relational Databases. IEEE TVCG, 2002.

[31] H. Wickham. ggplot2: Elegant Graphics for Data Analysis. Springer, 2009.

[32] J. Mackinlay. Automating the Design of Graphical Presentations of Relational Information. ACM Transactions on Graphics, 1986.

[33] S. F. Roth, J. Kolojejchick, J.Mattis, and J. Goldstein. Interactive Graphic Design using Automatic Presentation Knowledge. ACM CHI, 1994.

[34] J. D. Mackinlay, P. Hanrahan, and C. Stolte. Show Me: Automatic Presentation for Visual Analysis. IEEE TVCG, 2007.

[35] K. Wongsuphasawat, D. Moritz, A. Anand, J. Mackinlay, B. Howe, and J. Heer. Towards A General-Purpose Query Language for Visualization Recommendation. HILDA, 2016.

[36] K. Wongsuphasawat, D. Moritz, A. Anand, J. Mackinlay, B. Howe, and J. Heer. Voyager: Exploratory Analysis via Faceted Browsing of Visualization Recommendations. IEEE TVCG, 2016.

[37] K. Wongsuphasawat, Z. Qu, D. Moritz, R. Chang, F. Ouk, A. Anand, J. Mackinlay, B. Howe, and J. Heer. Voyager 2: Augmenting Visual Analysis with Partial View Specifications. ACM CHI, 2017.

[38] A. Srinivasan, S. M. Drucker, A. Endert, J. Stasko. Augmenting Visualizations with Interactive Data Facts to Facilitate Interpretation and Communication. IEEE TVCG, 2019.

[39] S. Kandel, R. Parikh, A. Paepcke, J. M. Hellerstein, J. Heer. Profiler: Integrated Statistical Analysis and Visualization for Data Quality Assessment. AVI, 2012.

[40] L. Grammel, M. Tory, and M. Storey. How Information Visualization Novices Construct Visualizations. IEEE TVCG, 2010.

[41] K. Z. Hu, M. A. Bakker, S. Li, T. Kraska, and C. A. Hidalgo. VizML: A Machine Learning Approach to Visualization Recommendation. arXiv: 1808.04819, 2018.

[42] D. Moritz, C. Wang, G. L. Nelson, H. Lin, A. M. Smith, B. Howe, and J. Heer. Formalizing Visualization Design Knowledge as Constraints: Actionable and Extensible Models in Draco. IEEE TVCG, 2019.

[43] Y. Luo, X. Qin, N. Tang, and G. Li. DeepEye: Towards Automatic Data Visualization. ICDE, 2018.

APPENDIX

Property of Impact

We restrict the impact-measures to be measures only containing non-negative values. Paper [13] provides a set of calculations to accommodate anti-monotonic condition being held by various aggregations (e.g., top-k average for AVG). The corresponding calculations are denoted as $MonoAggr_i(insight.subspace)$. thus, we define impact by the following two steps:

$$impact_i = \frac{MonoAggr_i(insight.subspace)}{MonoAggr_i(\{*\})}$$

$$impact = \max_i(impact_i)$$

Lemma. In definition 2, impact satisfies anti-monotonic condition, and it is bounded between 0 and 1.

Proof: Considering the impact-measures are restricted to only contain non-negative values, and since $MonoAggr_i(insight.subspace)$ satisfies anti-monotonic condition, thus,

$MonoAggr_i(insight.subspace) \leq MonoAggr_i(\{*\}) \rightarrow impact_i \in [0,1] \rightarrow impact \in [0,1]$, because every subspace is a subset of overall subspace {*}.

To prove anti-monotonic condition, let S and s be two subspaces where $s \in S$, then

$$impact_i(s) \leq impact_i(S)\ \forall i \rightarrow \max_i\big(impact_i(s)\big) \leq \max_i\big(impact_i(S)\big)$$

Complexity Analysis for FD Detection

Lemma. Time complexity of IsDependent (as shown in Table 9) is $O(lD) \ll O(D^2)$, here D is the number of columns, and $l = \max_i |X_i|$, where $|X_i|$ refers to the cardinality of a set X_i.

Proof: Similar to the analysis of graph traverse, we use a Boolean array *inspected* to record which column has been evaluated. Thus, each column will be evaluated at most once. And considering we inspect at most D columns, then we come up with the complexity $O(lD)$. Further considering in general, the FDs obtained from data schema describe FD relationship between a small set of dimensions, thus $l \ll d$, the proof concludes.

Example of Batched Query

For example, let a query request be {China, Android}⊕Year, and we know the subspace {China, Android} is generated from a previous query {China} ⊕ OS, then we choose OS as the expanding dimension rather than Country or any other dimensions, because all the subspaces generated by query {China}⊕OS have been inserted into the task queue (combined with all feasible breakdowns including Year), thus the query results expanded by OS will be useful for these tasks, but which is uncertain if we use other dimensions as the expanding dimension.

Pseudo Code

Table 9. Pseudo code of FD detection

```
/* check if a set of columns determine another column, given a set of
   basicFDs */
IsDependent(determinantCols, col, basicFDs)
     inspected ← {}
     return Qualify(determinantCols, col, inspected, basicFDs)

/* check if a set of columns determine another column recursively */
Qualify(determinantCols, col, inspected, basicFDs)
    /* reflexivity axiom */
    if col in determinantCols
         return true
    /* this column has already been inspected */
    if col in inspected
         return inspected[col]
    inspected[col] ← false
     /* retrieve all the determinant sets of col. it is possible that one
        column can be determined by multiple determinant sets */
     dtSets ← GetAllDeterminants(col, basicFDs)
    foreach set in dtSets
         qualify ← true
        /* if all the columns within this set can be determined,
         then col can be determined according to transitivity axiom */
         foreach newCol in set
              if Qualify(determinantCols, newCol, inspected, basicFDs) is false
                   then qualify ← false
                        break
         if qualify is true
              then inspected[col] ← true
                   return true
     return false
```

Table 10. Pseudo code of batched-query with cache

```
/* a specific query issued from QuickInsights miner layer */
Query(subspace, expanding, breakdown, params)
     ret ← Cache.Lookup(subspace, breakdown, params)
     if ret ≠ null
        return ret
     /* a special optimization for the case where breakdown is null: we swap
     breakdown and expanding to increase cache hit */
     if breakdown = null and expanding ≠ null
         newSubspace ← subspace - expanding
         ret ← Cache.Lookup(newSubspace, expanding, params)
         if ret ≠ null
             /* in case the lookup successful, we need to re-format result */
             return ExtractResult(ret)

     /* conduct real data query */
     queryResult ← DataDriver.AggregationQuery(subspace, expanding,
     breakdown, params)
     if expanding is null
         ret ← first in queryResult
         Cache.Add(subspace, breakdown, params, ret)
     else
         root ← subspace - expanding
         foreach t in queryResult
             newSubspace ← root + {expanding:t.Key}
             Cache.Add(newSubspace, breakdown, params, t.Value)
             if newSubspace is subspace
                 ret ← t.Value
     return ret
```

Example of Adding A New Insight Type

Suppose we would like to support a new insight type which is equivalent to the pattern depicted in [9], we first need to register it as a new insight type, named "HighDeviation". Considering the subject of HighDeviation insight is with single subspace and single measure, thus only the tasks with single subspace as input are allowed for its evaluation, this is reflected by a single-line checking "*case HighDeviation: return subspaces.Count==1;*" in the method *CanEvaluate* in Table 11. In the *Evaluation* method, certain statistical metrics are calculated to measure the deviation for each individual measure, and qualified insights are output.

Table 11. Three steps for supporting a new type of insight

```
Step1: add the new insight type
enum InsightType {…, newType, …}

Step2: implement insight evaluation of the new type
List<Insight> Evaluate(List<Dictionary<Measure, AggrResult>> aggrResults);

Step3: register new insight type to task execution pre-condition
bool CanEvaluate(
List<Subspace> subspaces, Dimension breakdown, InsightType type);
```

Examples of FD Induced EII

For example, suppose there exists FD between two measures $Sales(USD) \rightarrow Sales(EURO)$ (falls into the category of ID5 in Table 4), the corresponding values only differ by a constant exchange-rate. These two measures will exhibit perfect correlation no matter breakdown by any dimension (thus the relationship is pre-determined) when drawn in a scatter plot, but clearly provides little value for analysis. The example of ID2 in Table 4 is another case about measure *height* determining dimension *Height-Category*. For example, the value of *Height-Category* is calculated by measure *height*, by setting $low = height \leq 100$, $high = height \geq 1000$, $medium = 100 < height < 1000$. Any insight describes *height* breakdown by *Height-Category* would become a trivial *Outstanding No. 1* insight: "*height* of *high* is *outstanding No. 1* among all *Height-Categories*", which is pre-determined no matter what subspace of the insight is. The details of how ID1~5 induce EII are available at website [15] due to page limit.

Profiles of Non-Expert Users

To mitigate any potential bias, we select non-expert participants by different jobs, genders, and different familiarity with data analysis, as shown in Table 12.

Table 12. Statistics of non-expert users

Job role	Count
Researcher	17
Developer	8
UX Designer	2
IT	1
Admin	1
PM	1

Gender	Count
male	18
female	12

Analysis frequency	Count
Daily	5
Weekly	8
Monthly	10
Seldom	7

Details of "Movie" Dataset

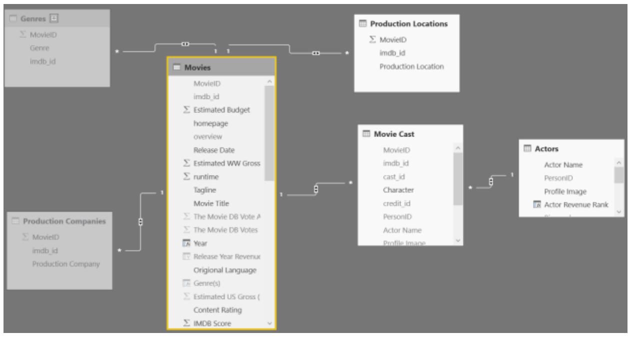


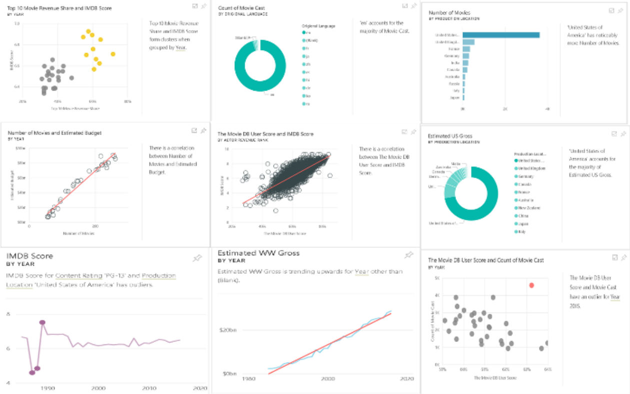
Figure 9. Snapshot of the schema of 'Movie' Dataset

Figure 9 shows a snapshot of the database schema of 'Movie' data. This is a real-world dataset, containing the various information of movie in the years from 1985 to 2016. This is a typical multi-dimensional dataset, which is formed by six tables, connected by Snowflake schema. Table 13 shows the scale of each table. There are in total about 60 dimensions, and almost every table has more than 10,000 rows. Thus the search space for QuickInsights is very large.

Table 13. Data scale of each table

Table Name	#Dimensions	#Measures	#Rows
Movies*	27	8	4740
Movie Cast	10	1	74038
Actors	8	2	39567
Genres	2	1	22470
Production Companies	2	1	22222
Production Locations	2	1	12084

When this dataset is run by Power BI, QuickInsights could generate quite a few insights within 20 seconds. Figure 10 shows nine sample insights generated by QuickInsights. We have used these insights to conduct the user study. The details of the user study are presented in Section 4.2.2.



Top 10 Movie Revenue Share and IMDB Score form clusters when grouped by Year.

'en' accounts for the majority of Movie Cast.

'United States of America' has noticeably more Number of Movies.

There is a correlation between Number of Movies and Estimated Budget.

There is a correlation between The Movie DB User Score and IMDB Score.

'United States of America' accounts for the majority of Estimated US Gross.

IMDB Score for Content Rating 'PG-13' and Production Location 'United States of America' has outliers.

Estimated WW Gross is trending upwards for Year other than (Blank).

The Movie DB User Score and Movie Cast have an outlier for Year 2015.

Figure 10. Snapshot of sampled insights recommended from Movie