BigIN4: Instant, Interactive Insight Identification for Multi-Dimensional Big Data

Qingwei Lin[1], Weichen Ke[2], Jian-Guang Lou[1], Hongyu Zhang[3],
Kaixin Sui[1], Yong Xu[1], Ziyi Zhou[1], Bo Qiao[1], Dongmei Zhang[1]
[1]Microsoft Research, Beijing, China
[2]Peking University, Beijing, China, [3]The University of Newcastle, Australia
{qlin,jlou,kasui,yox,v-ziz,v-boqiao,dongmeiz}@microsoft.com
weichenzero@gmail.com,hongyu.zhang@newcastle.edu.au

ABSTRACT

The ability to identify insights from multi-dimensional big data is important for business intelligence. To enable interactive identification of insights, a large number of dimension combinations need to be searched and a series of aggregation queries need to be quickly answered. The existing approaches answer interactive queries on big data through data cubes or approximate query processing. However, these approaches can hardly satisfy the performance or accuracy requirements for ad-hoc queries demanded by interactive exploration. In this paper, we present BigIN4, a system for instant, interactive identification of insights from multi-dimensional big data. BigIN4 gives insight suggestions by enumerating subspaces and answers queries by combining data cube and approximate query processing techniques. If a query cannot be answered by the cubes, BigIN4 decomposes it into several low dimensional queries that can be directly answered by the cubes through an online constructed Bayesian Network and gives an approximate answer within a statistical interval. Unlike the related works, BigIN4 does not require any prior knowledge of queries and does not assume a certain data distribution. Our experiments on ten real-world large-scale datasets show that BigIN4 can successfully identify insights from big data. Furthermore, BigIN4 can provide approximate answers to aggregation queries effectively (with less than 10% error on average) and efficiently (50x faster than sampling-based methods).

CCS CONCEPTS

• **Information systems** → **Data management systems**;

KEYWORDS

Insight identification, approximate query processing, data cube, interactive data analytics.

ACM Reference Format:
Qingwei Lin[1], Weichen Ke[2], Jian-Guang Lou[1], Hongyu Zhang[3], Kaixin Sui[1], Yong Xu[1], Ziyi Zhou[1], Bo Qiao[1], Dongmei Zhang[1]. 2018. BigIN4: Instant, Interactive Insight Identification for Multi-Dimensional Big Data. In *KDD '18: The 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, August 19–23, 2018, London, United Kingdom.* ACM, New York, NY, USA, 9 pages. https://doi.org/10.1145/3219819.3219867

1 INTRODUCTION

The era of big data offers new opportunities for business intelligence. It is now increasingly important to find insights from the abundant multi-dimensional data in the elastic storage systems in order to make better and faster decisions. Insights are interesting facts that can be extracted from data and can help users obtain a deep understanding of the data. For example, insights from a tablet sales dataset can be "the total sales of company A have been increasing over the past five years in the USA", or "tablets with small sizes dominate the Australia market in 2017", and so on.

Insight identification has been a tedious task for data scientists because insights are not straightforward to be discovered. Users have to manually enumerate a set of dimensions and aggregate functions before they can find interesting results. It is desirable to have a tool to give automated insight suggestions interactively on an interested set of conditions specified by the user to relieve the hard trial-and-error process. Oftentimes a first set of query results obtained from a data source may spark user interest for a second query. The users may specify new filters if they would like to drill-down the results or explore a different set of dimensions if they fail to find interesting results on the current space. An interactive system that supports both automated insight suggestions and user exploration can greatly expedite the insight identification process.

A major challenge for such a system on big data is the long query latency. Even the fastest database systems can take hours or days to answer the simplest queries [1]. This is a critical problem for our scenario. On one hand, automated insight mining needs to issue a lot of aggregation queries to extract insights from the results. The waiting time before getting the suggested insights would be unacceptable if these queries cannot be answered efficiently. On the other hand, when users look for interested dimensions or drill-down the query results, it is important to give answers to the exploration queries within a short time (usually in seconds) to keep users engaged. Slow interactions will severely interrupt the user's interactive analysis. [14].

Much research work has been conducted to make query processing faster and enable interactive exploration on big data. One common technique is to create pre-aggregated data cubes for the dataset so that answers for queries can be directly found from one of the cubes. However, it is expensive to cover all possible aggregation queries over big data with data cubes. Another technique is called

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

ACM ISBN 978-1-4503-5552-0/18/08…$15.00
https://doi.org/10.1145/3219819.3219867

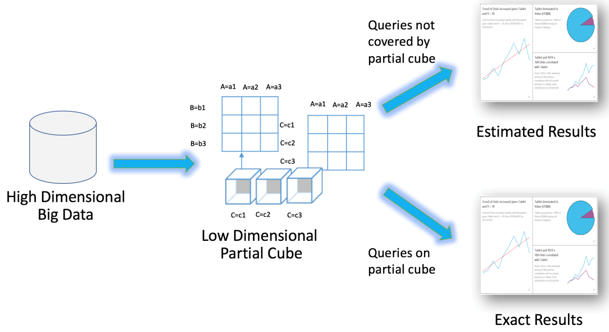

Figure 1: An overview of the proposed approach

Approximate Query Processing (AQP) [7], which focuses on trading off the accuracy of the answers for less response time and storage space. AQP can provide approximate answers to queries within a short time, which is suitable for interactive analysis. Traditional AQP methods include sampling, histograms, and wavelets [5]. Although effective, most existing methods require prior assumptions on either data distribution or query workloads, which may not be applicable to ad-hoc insight queries over real-world datasets.

In this paper, we propose BigIN4, an interactive insight identification system for multi-dimensional big data. BigIN4 can give suggested insights automatically by iteratively enumerating subspaces. The OLAP engine of BigIN4 can answer aggregation queries over big data in seconds with estimated statistical intervals by combining data cube and approximate query processing techniques. Different from existing data cube based OLAP engines, BigIN4 only requires a set of low dimensional data cubes for a dataset while putting no restriction on queries. As shown in Figure 1, if BigIN4 fails to find a data cube to answer the query, it will call DynamicBayes, a Bayesian Network based algorithm proposed in this paper, to obtain an approximate answer from the data cubes. Unlike many of the approximate query processing engines, it does not require any prior knowledge about the data distribution or query distribution.

We have implemented BigIN4 and evaluated it on ten real-world large-scale datasets. Our experiments show that BigIN4 can successfully identify insights (with a precision of 95.88% and a recall of 96.07% on average) over high dimensional big data with billions of records and tens of columns within 4.4 seconds. Moreover, BigIN4 can provide approximate answers to aggregation queries effectively (with less than 10% error on average) and efficiently (50x faster than sampling-based methods). We have also successfully applied BigIN4 to industrial practice and multiple teams in Microsoft have used it to analyze their business data.

In summary, in this paper we make the following contributions:

- We develop BigIN4, which supports instant, interactive insight identification from multi-dimensional big data. Our experimental results show that BigIN4 can identify insights effectively and efficiently.
- We propose to combine data cube and AQP techniques in order to give approximate answers to aggregation queries quickly. More specifically, we propose DynamicBayes, a new AQP method that decomposes the high dimensional queries into several low dimensional ones. Unlike the previous approaches, it does not require any prior knowledge about future queries, nor any assumptions about the data distribution.
- We have successfully deployed BigIN4 inside Microsoft. We also report the lessons learned during the deployment in this paper.

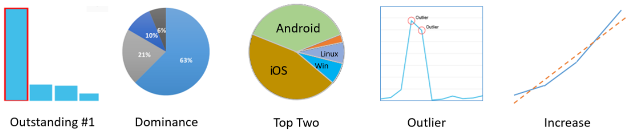

Figure 2: Example of insight types

The organization of the paper is as follows. We introduce background and motivation of our work in Section 2. Section 3 describes our approach. Section 4 presents our experimental design and results. In Section 5, we present the success stories of our approach in industrial practice. Section 6 surveys related work followed by Section 7 that concludes this paper.

2 BACKGROUND AND MOTIVATION

2.1 Insight Identification

In the era of big data, there is an increasing demand on analyzing data with large volumes in the business world. These data can be collected from daily sales transactions, user click streams, application download logs, and so on. It is critical for a company to discover the insights hidden in these raw data in order to improve their business. Figure 2 illustrates some insight types. In BigIN4, we define insights as facts specified by the end users according to their analysis needs. Some examples of insight definitions used in this paper are as follows:

- *Outstanding #1*: a subspace is remarkably different from others in terms of its aggregated measure value.
- *Dominance*: a majority of the total value can be attributed to a single factor when broken down by another dimension.
- *Top Two*: two members of a dimension have much larger values than other members of that dimension.
- *Outlier*: specific times with values significantly different from the others in time series data.
- *Increase/Decrease*: upward or downward trends in time series data.

Users can pick the insights they would like to show and specify the threshold for each type of insight. They can also define more customized types of insights. We refer the readers to [21] for more details of the advanced insights.

2.2 Data Exploration

Insight identification is done by interactive data explorations on multi-dimensional data. By "Exploration" we mean a series of aggregation queries like `SELECT SUM(Sales) WHERE Brand="Dell" GROUP BY Region` over multi-dimensional data. Multi-dimensional

Table 1: Sample Tablet Sales Data

Year	CPU	OS	Region	Brand	Sales
2000	ARM	iOS	China	Apple	17014
2000	x86	Windows	US	Amazon	657
2001	ARM	Linux	China	Dell	39633
2001	ARM	Android	US	Dell	37814
2001	ARM	Android	US	Amazon	45859
2002	ARM	iOS	China	Apple	76464
2002	x86	Windows	US	HP	725

Table 2: Data cube on {CPU, Brand}

CPU	Brand	SUM(Sales)	Count
ARM	Amazon	45859	1
ARM	Dell	77447	2
ARM	Apple	93478	2
x86	Amazon	657	1
x86	HP	725	1

data is a prevalent type of data, where a data column is either a dimension or a measure. Dimensions are used to group and filter the data records. The values of dimensions are either categorical (e.g., "Brand") or ordinal (e.g., "Priority"). Measures reflect quantitative properties that are related to the analytical task (e.g., "Sales"). The values in measure columns are typically numerical values, which can be computed by a set of aggregate functions (such as sum, count, etc.). Answering analytical queries on multi-dimensional data is called On-Line Analytical Processing (OLAP), which is a field that has been extensively studied in the database community.

Building data cubes is a method for optimizing OLAP engines. A data cube is a table that contains the pre-aggregated results on a subset of the columns. Usually, a series of cubes are built for one dataset, so that the answer to a query can be directly found out from one data cube without doing online aggregation. However, a critical drawback of this method is that it suffers from the curse of dimensionality. A query is *answerable* only when its referring columns are covered by one of the existing data cubes. In order to answer all kinds of queries, the number of cubes grows exponentially with the number of dimensions, which is extremely expensive for both computing and storage, especially for big data. Reducing the cost of building data cubes often restricts the dimensions that can be asked together, making some queries "unanswerable". Once an "unanswerable" query is posed to an OLAP engine, it has to be processed by performing online aggregation on the raw data, which would be prohibitively slow for interactive data exploration.

2.3 A Running Example

To better motivate our work, let us consider a Tablet Sales table as shown in Table 1, which has the following columns: *Year, CPU, OS, Region, Brand, Sales. Sales* is the measure column and all the other columns are dimension columns.

Many insights can be discovered from the data in Table 1. For example, "*ARM CPU* is the outstanding #1 in the *US* market", "*Android* dominated the market in *2001*", "The total sales of tablets have been increasing in *China*", and so on. To identify these insights, we need to issue many SQL-based aggregation queries (in the form of `SELECT Aggr(Measure) WHERE {K=v}`) involving many different combinations of dimension attributes. As an example, the insight "The total sales of tablets have been increasing in *China*" can be extracted by querying `SELECT SUM(Sales) WHERE Region="China" GROUP BY Year`. It would be tedious for a data scientist to enumerate the dimension attributes manually to find insights. BigIN4 can give automatic insight suggestions interactively on a user's interested set of conditions by searching through the attributes. We will explain the details in Section 3.1.

In this paper, we use *Q({K=v})* (or *Q({v})* when no ambiguity is introduced) to denote the answer for an insight query `SELECT Aggr(Measure) WHERE {K=v}`. The constraints *{K=v}* also define a *subspace S*, which contains all the records in the dataset satisfying the constraints. We use `SUM` for `Aggr` if not otherwise specified in this paper.

To enable quick query answering for interactive exploration, an OLAP engine may build a set of data cubes for the dataset offline. Suppose we have the following data cubes available in the cluster memory: *{CPU, Brand}*, *{CPU, OS}*, *{Year, CPU, Region}*, and *{Brand, OS, Region}*. The query *Q(2001, China)* is answerable because the columns mentioned in this query *{Year, Region}* is a subset of an existing cube *{Year, CPU, Region}*. In this case, it can be answered in a very short time. On the other hand, *Q(ARM, iOS, Apple, US)* is unanswerable, because there is no cube covering the column set *{CPU, OS, Brand, Region}*. This query has to be executed on raw data in common OLAP systems. We propose an algorithm to give an estimated answer in seconds to the unanswerable queries in Section 3.2.

3 THE PROPOSED APPROACH

In this section, we describe the details of BigIN4, which consists of two major components: insight mining and query answering.

3.1 Insight Mining

An insight derived from a multi-dimensional dataset is associated with a subspace. The subspaces form a lattice, where each node indicates a subspace and each edge indicates a parent-child relationship. Figure 3 shows an example of the subspace lattice. Given a multi-dimensional subspace S by the user, BigIN4 searches for insights by enumerating subsequent subspaces, aggregation functions, and insight types. BigIN4 enumerates the subspaces in a top-down order, starting from the user-defined subspace and iteratively adds new dimensions and values to it until a user-specified depth is reached. The insights are identified through the query answering module (which will be described in Section 3.2) in each iteration. The general insight mining algorithm is shown in Algorithm 1.

To improve the search efficiency, we also design a suite of performance optimization techniques (e.g., pruning, ordering, sibling cube, and computation sharing). For details, we refer the readers to [21], which is dedicated to the extraction of top k insights from multi-dimensional data.

3.2 Query Answering

A key problem for BigIN4 is how to estimate the answer for an unanswerable query. We propose *DynamicBayes*, a novel AQP algorithm

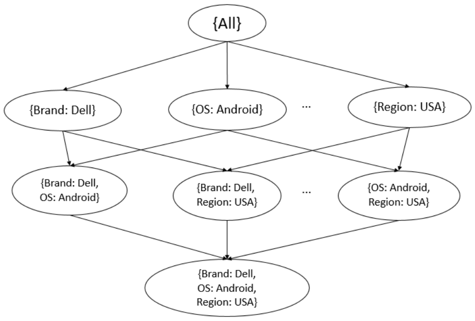
{All}
{Brand: Dell}
{OS: Android}
...
{Region: USA}
{Brand: Dell, OS: Android}
{Brand: Dell, Region: USA}
...
{OS: Android, Region: USA}
{Brand: Dell, OS: Android, Region: USA}

Figure 3: An example of subspace lattice

```
Algorithm 1: InsightMining
Input: S: a user-defined subspace, $\tau$: search depth
Output: Insights: the identified insights
Insights = $\Phi$
$D \leftarrow$ Enumerate all descendants of $S$ with depth $\tau$
foreach $s$ in $D$ do
    $C \leftarrow$ Determine a set of aggregation functions for $s$
    foreach $c$ in $C$ do
        $T \leftarrow$ Determine a set of insight types for $c$ and $s$
        foreach $t$ in $T$ do
            $i \leftarrow$ Identify an insight based on $< s, c, t >$
              through Query Answering
            $Insights \leftarrow Insights \cup < i >$
        end
    end
end
return Insights
```

which can decompose the unanswerable queries into several low dimensional answerable queries to estimate the answer through an online constructed Bayesian Network (BN). The only requirement for DynamicBayes is that all 2-d queries are answerable, which can be easily satisfied when building the data cubes offline.

We treat each dimension A as a random variable and each attribute a as a possible outcome of it. For a query *Q(A=a)*, we define $P(a) = \frac{Q(a)}{N}$, where $N = \sum(Measure)$ for all records. $P(a)$ can be viewed as the probability of event *A=a*. DynamicBayes answers *Q(a)* by estimating *P(a)* since N remains constant all the time. Consider the running example described in Section 2.3, if we estimate *P(ARM, Amazon) = 0.21*, *Q(ARM, Amazon)* can be answered with:

$$Q(ARM, Amazon) = P(ARM, Amazon) \cdot N \approx 45814.$$

The intuition of query decomposition is that some dimensions can be approximately viewed as (conditional) independent under certain circumstances. If the dependency between events *A=a* and *B=b* is weak given event *C=c*, the joint probability *P(A=a, B=b, C=c)* can be estimated with $P(a,b,c) \approx P(a|c) \cdot P(b|c) \cdot P(c)$. If all the low dimensional queries are answerable, the answer can be estimated in a very short time by looking them up in the data cubes. Formally, if we have a BN for a set of events $W = \{w_1, w_2 ... w_n\}$, we can estimate the joint probability of $P(W)$ efficiently with:

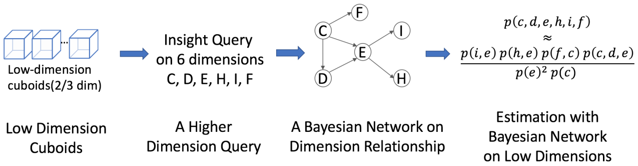
Low-dimension cuboids(2/3 dim)
Insight Query on 6 dimensions C, D, E, H, I, F
C
F
E
I
D
H
$p(c,d,e,h,i,f) \approx \frac{p(i,e)\,p(h,e)\,p(f,c)\,p(c,d,e)}{p(e)^2\,p(c)}$
Low Dimension Cuboids
A Higher Dimension Query
A Bayesian Network on Dimension Relationship
Estimation with Bayesian Network on Low Dimensions

Figure 4: The decomposition of a high dimensional query into low dimensional ones

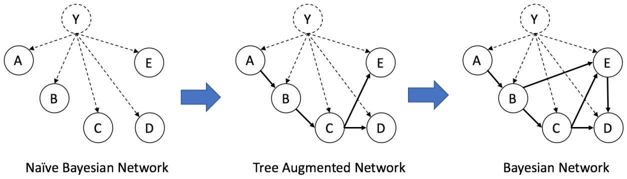

Figure 5: Different kinds of Bayesian Models

$$P(W) \approx \prod_{w \in W} P(w|IN(w))$$

where $IN(w)$ is the direct predecessors of node w in the BN. For example, consider the query *Q(ARM, iOS, Apple, US)* in Section 2.3. Suppose the BN is the one shown in Figure 6(a), we can give an estimated answer to it with:

$$P(ARM, iOS, Apple, US) \approx \frac{P(ARM, US) \cdot P(iOS, US) \cdot P(Apple, US)}{P(US)^3}$$

Therefore, it is intuitive for us to build a BN to extract the dependency between the dimensions in the data. Figure 4 outlines the decomposition of a high dimensional insight query with a BN.

In the following sections, we will first introduce the dynamic construction of BN and then describe DynamicBayes in detail.

3.2.1 Dynamic Construction of Bayesian Network. We use a special form of network, Tree Augmented Network (TAN) [4] as the skeleton of our BN. Figure 5 illustrates a Naive Bayesian Network. a TAN, and a general Bayesian Network. TAN assumes that each event depends on at most only one other event (so the network is a tree). The reason we start from TAN is that although it is possible to learn an optimal BN from data (such as the one proposed in [3]), the process is too expensive for high-dimensional big data. Besides, since usually we can only afford to maintain a set of low dimensional data cubes, the complexity of our BN is largely limited. Under these conditions, TAN is a good balance point between model complexity and computation efficiency. After the skeleton is built, we continue to add more edges to it to maximize the use of information stored in the data cubes, while ensuring all the potential sub-queries are answerable.

We learn the structure of the TAN from the information theory perspective. Specifically, for a query *Q({K=v})*, we regard the conditions *{K=v}* as the nodes and the absolute value of pairwise mutual information (PMI) between them as the weight of the edges. PMI is

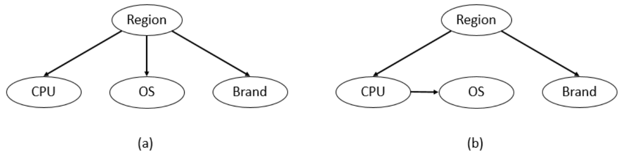

Figure 6: Two different Bayesian models for one query

defined as:

$$PMI(a,b) = \log(\frac{P(a,b)}{P(a) \cdot P(b)}).$$

A high absolute value of PMI(a, b) indicates a strong correlation between events *A=a* and *B=b*. Conversely, a low absolute value of PMI(a, b) suggests that they are nearly independent. Therefore, we aim to keep the edges with high PMI when building our network.

One unique feature of our approach is that it builds the BN completely online, after a query is submitted to BigIN4. The reason we design our network in a dynamic manner is that the statically constructed BN based on column-wise mutual information is not optimal for specific queries, especially those specifying a small subspace. Since mutual information tends to ignore elements with small fractions, queries specifying these elements will be answered poorly because the columns are wrongly decomposed. For example, the mutual information *I(CPU, OS)* is only 0.038 in Table 1, suggesting that *CPU* and *OS* are approximately independent. The reason for that is *"ARM"* dominates the *CPU* column, taking more than 99% share for the total measurement, and *"ARM"* does not have an apparent relationship with *OS*. Therefore, a static BN would consider *CPU* and *OS* independent. A possible static BN for the sample data could be Figure 6(a), where *CPU, OS,* and *Brand* all depend only on *Region*.

However, consider the following query: *Q(x86, Windows, Microsoft, US)*. We can find from Table 1 that when *CPU* is *x86*, all the tablets are Windows operating systems, which means that *"x86"* and *"Windows"* are strongly correlated. In fact, PMI(*x86, Windows*) is 5.06, much higher than 0.038. With the statically constructed network, the system will still consider *CPU* and *OS* are independent for this query. As a result, it could give incorrect answers to it. A better network for this query is shown in Figure 6(b).

The root cause of this problem is that traditional Bayesian methods aim at approximating the discrete possibility distribution of the full given dataset instead of optimizing the answers for each aggregation query. Therefore, we propose building the network dynamically as a better solution for an AQP module.

3.2.2 The DynamicBayes Algorithm. There are three steps in DynamicBayes as illustrated in Figure 7. First, we construct a PMI matrix to depict the mutual dependency relationship between query dimensions. Calculating the PMI matrix is essentially answering all 2-d combinational sub-queries of the referred column pairs, so it can be done very quickly by consulting the data cubes. Second, based on the PMI matrix, we create a maximum spanning tree as the network skeleton. Finally, we add more dependency edges to the network as long as all the potential low dimensional queries are answerable. An edge $x \longrightarrow y$ can be added if $\{x, y\} \cup IN(y)$ is covered by one of the data cubes and no cycle is introduced after adding this edge. The details of the algorithm are described in Algorithm 2.

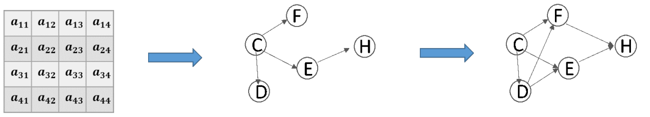
a_{11} a_{12} a_{13} a_{14}
a_{21} a_{22} a_{23} a_{24}
a_{31} a_{32} a_{33} a_{34}
a_{41} a_{42} a_{43} a_{44}
F
C
H
E
D
F
C
H
E
D

Figure 7: The outline of DynamicBayes Algorithm

Besides the estimation value, we also provide the bounds of the true value. The basic idea here is that the bound of a high dimensional query can be deduced by a set of lower dimensional queries (or a superset of the query subspace). Suppose the subspace specified in the query is s. The upper and lower bounds for estimated values are as follows:

Upper bound:

$$P(s) \leq \min_{s \subseteq S} P(S)$$

Lower bound:

$$P(s) \geq \max_{s \subseteq S} P(S) - UpperBound[P(S-s)]$$

Since these bounds always hold true, they can also be used to bound the estimated answer given by the Bayesian Network.

Algorithm 2: DynamicBayes

```
Input: Q{K=v}: an aggregation query
Output: BN: the built Bayesian Network
G = Φ
foreach $(a, b)$ in $v \times v$ do
    $e = Edge(start = a, end = b, weight = |PMI(a, b)|)$
    Add e to G
end
BN=MaximumSpanningTree(G)
Sort G.E according to weight in descending order
foreach e in G.E do
    if Answerable({e.start, e.end} ∪ IN(e.end)) and no cycle
      after adding e then
        Add e to BN
    end
end
return BN
```

4 EXPERIMENTS

4.1 Experimental Design

To evaluate the proposed algorithm, we have conducted experiments on ten large-scale real-world datasets as shown in Table 3. These datasets contain multi-dimensional data collected from multiple Microsoft large-scale online service products. In order to better demonstrate the performance of the AQP module in BigIN4, we only create all the 2-d cubes for each dataset. In other words, we build M^2 small cubes for a dataset with M dimensions.

We implemented BigIN4 on a cluster of three machines, each equipped with 256GB memory, 32 CPU cores (2.60GHz) and a 40TB

Table 3: The evaluation datasets

No.	Name	#Rows	#Dims	Cube Size	Ratio
A	Sales	1.12B	9	249K	0.02%
B	Subscription	1.31B	12	438K	0.03%
C	LifetimeEvents	852M	14	765K	0.09%
D	CrpApiUsage	2.51B	13	1349K	0.05%
E	OnlineAct	1.24B	26	3562K	0.29%
F	IntentionalAct	1.32B	10	247K	0.02%
G	BizChat	2.48B	13	523K	0.02%
H	ApiQosEvent	924M	24	1035K	0.11%
I	SharedFile	1.18B	31	2047K	0.17%
J	JobMetaData	2.16B	25	1046K	0.05%

disk. We deployed Hadoop and Spark on this cluster and all the experiments are conducted on it. The datasets are stored in the HDFS with 3 replicas. To facilitate the examination of insights and the exploration of data, we also develop a drag-and-drop based GUI frontend for BigIN4.

To evaluate our approach, we design experiments to address the following research questions:

RQ1: Can BigIN4 identify insights effectively and efficiently? This RQ evaluates the ability of BigIN4 to successfully identify insights from multi-dimensional datasets. We show that insight identification can be accomplished even without perfect answers to the queries. We perform the evaluation for each insight type separately and calculate the percentage of insights that are successfully identified and the total time spent on insight identification. For evaluation, we compare the results of BigIN4 with the results obtained from SparkSQL, which is one of the most widely-used systems for big data analytics. The latter serves as the ground truth for insight queries but takes much longer time to complete. In order to obtain the ground truth in a reasonable time, we limit the maximum search depth to 4 in this experiment. We use the default thresholds for the parameters for defining insights.

RQ2: Can BigIN4 answer aggregation queries effectively and efficiently? In this RQ, we further evaluate the effectiveness of BigIN4 in answering queries against big data. We address two major questions regarding our DynamicBayes algorithm:

- DynamicBayes does not learn the optimal Bayesian Network. Is the learned network adequate to describe the dependency among the dimensions?
- DynamicBayes is an online algorithm. Is the overhead of building the network small enough for interactive queries?

The queries we use in the experiments are randomly generated from a typical OLAP query template: `SELECT SUM(Measure) WHERE condition GROUP BY column`. The total number of columns in `WHERE` and `GROUP BY` is randomly determined from 2 to 7, and the value for filter columns are also randomly picked from all the possible values appeared in the dataset. In total, we generate 3000 unique queries for each dataset. We filter out queries specifying an empty subspace.

We compare our algorithm with three baselines: 1) uniform sampling (US), which builds one uniform sample; 2) stratified sampling (SS), which builds a stratified sample on all columns; and 3) multiple stratified sampling (MSS), which builds a set of stratified samples in the way similar to the optimization framework used in BlinkDB [1]. We do not consider history workloads in deciding which samples to build because we do not have prior knowledge about future queries. The size of the samples for each dataset is the same as the total size of the cube set.

RQ3: Can DynamicBayes outperform the statically constructed Bayesian Network? In this RQ, we compare the effectiveness of DynamicBayes (constructing a Bayesian Network online) and a static Bayesian Network (constructing a Bayesian Network offline), and evaluate the usefulness of DynamicBayes in BigIN4. We show the necessity of building the network online in this RQ.

Evaluation Metrics: We evaluate the effectiveness of insight identification (RQ1) using the Precision and Recall metrics, which are widely adopted in machine learning and data mining.

We evaluate the effectiveness of BigIN4 in approximate query processing (RQ2 and RQ3) using the SMAPE (symmetric mean absolute percentage error) metric, which is an average error rate defined as:

$$SMAPE = \frac{1}{n}\sum_{t=1}^{n}\frac{|E_t - A_t|}{|E_t| + |A_t|}$$

where E_t is the estimated answer and A_t is the ground-truth value. SMAPE is a fair metric for error evaluation because it is symmetric for both underestimation and overestimation. The value of SMAPE is from 0 to 1. The smaller the better.

4.2 Experimental Results

RQ1: Can BigIN4 identify insights effectively and efficiently?

Table 4 shows the precision and recall values achieved by BigIN4. Clearly, BigIN4 can achieve comparative results as SparkSQL within a much shorter time. As ground truth, the precision and recall achieved by SparkSQL are all 100%. BigIN4 achieves above 92% precision (on average 95.88%) and above 93% recall (on average 96.07%) in identifying all kinds of insights. The average time required by BigIN4 to identify all insights is only 4.4 seconds, while the time required by SparkSQL is 17.7 minutes. The speedup is 241x. Overall, our experimental results confirm the effectiveness and efficiency of BigIN4 in identifying insights.

RQ2: Can BigIN4 answer aggregation queries effectively and efficiently?

Table 5 shows the evaluation results for all the 10 datasets. On average, our approach performs much better than the sampling approaches (0.09 vs. 0.19 in terms of error rates). BigIN4 also gives more stable results than sampling approaches as indicated by its low standard deviation of error rates.

Table 6 shows the average response time for queries in BigIN4. The results show that BigIN4 is 5000x faster than SparkSQL (8.3ms vs. 42.3s). BigIN4 is also 50x faster than sampling-based methods. The average response time for each dataset is less than 20ms, further justifying that BigIN4 provides quick response to the queries. BigIN4 is faster than sampling because BigIN4 stores aggregated results instead of samples. For answerable queries, these results can be quickly found out without online aggregation. Although the Bayesian Network is calculated online, all the data needed for

Table 4: The identification of insights

Type	BigIN4			SparkSQL
	Precision	Recall	Time	Time
Outstanding #1	96.9% (31/32)	100% (31/31)	2.8s	8.4m
Top Two	92% (23/25)	95.8% (23/24)	4.7s	7.8m
Increase	96.4% (27/28)	93.1% (27/29)	6.2s	32.1m
Decrease	100% (16/16)	94.1% (16/17)	5.7s	29.8m
Dominance	97.7% (42/43)	97.7% (42/43)	3.2s	8.6m
Outlier	92.3% (24/26)	96% (24/25)	3.9s	19.6m
Average	95.88%	96.07%	4.4s	17.7m

Table 5: The SMAPE of different AQP methods

Dataset	US		SS		MSS		BigIN4	
	Avg.	Std.	Avg.	Std.	Avg.	Std.	Avg.	Std.
A	0.47	0.37	0.35	0.31	0.16	0.14	0.10	0.08
B	0.55	0.24	0.36	0.26	0.21	0.10	0.13	0.02
C	0.34	0.35	0.31	0.25	0.15	0.06	0.02	0.02
D	0.53	0.27	0.19	0.22	0.21	0.13	0.15	0.06
E	0.40	0.38	0.32	0.26	0.18	0.10	0.07	0.02
F	0.43	0.41	0.35	0.27	0.19	0.11	0.05	0.03
G	0.31	0.37	0.26	0.23	0.23	0.15	0.07	0.02
H	0.42	0.36	0.31	0.18	0.20	0.04	0.09	0.03
I	0.55	0.41	0.36	0.32	0.24	0.08	0.12	0.12
J	0.29	0.35	0.18	0.21	0.16	0.05	0.11	0.07
Average	**0.43**	0.35	**0.30**	0.25	**0.19**	0.10	**0.09**	0.05

that computation can also be directly found from the cubes, which introduced minimal overhead.

We also give the error distributions on dataset A (Figure 8). The distribution is similar on other datasets. BigIN4 can answer 82% queries with less than 10% errors. An interesting fact is that BigIN4 gives perfect answers to 77% of the queries whose dimensionality ranges from 2 to 7 using only information from 2-d data cubes.

Overall, the evaluation results confirm the effectiveness and efficiency of the proposed approach in answering aggregation queries.

RQ3: Can DynamicBayes outperform the statically constructed Bayesian Network?

In this experiment, we compute the mutual information and build the Bayesian Network with our algorithm in an offline manner. We answer all the queries that need to be estimated with statically constructed Bayesian Network. The average SMAPE values of the two algorithms on the 10 datasets are shown in Table 7. Dynamic BN outperforms the static one on all datasets. The average SMAPE achieved by Static BN and Dynamic BN is 22.01% and 9.21% respectively. The results justify our view that static Bayesian Network cannot describe the relationship properly under a certain subspace given by the query.

Table 6: Average query response time of three methods

Dataset	SparkSQL	MSS	BigIN4
A	65.6s	515.7ms	14.3ms
B	23.9s	345.3ms	6.6ms
C	27.3s	276.8ms	6.2ms
D	92.7s	756.4ms	18.2ms
E	51.6s	672.3ms	12.1ms
F	43.8s	364.5ms	4.1ms
G	23.2s	186.8ms	3.4ms
H	36.8s	465.6ms	6.7ms
I	27.3s	275.3ms	4.4ms
J	30.6s	286.7ms	6.8ms
Average	42.3s	414.5ms	8.3ms

Table 7: The SMAPE comparison between statically and dynamically constructed Bayesian Network

Dataset	Static	Dynamic	Dataset	Static	Dynamic
A	20.32%	10.49%	F	9.67%	5.28%
B	30.27%	13.32%	G	9.62%	7.17%
C	18.65%	1.69%	H	23.76%	8.92%
D	34.29%	14.82%	I	32.89%	11.64%
E	20.46%	6.74%	J	20.17%	11.12%

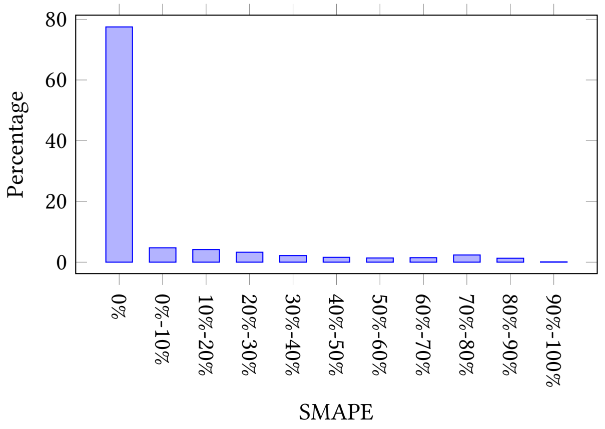

Figure 8: The error distribution on dataset A

4.3 Discussions

In this section, we discuss the results and point out some assumptions in our approach:

- We find that high dimensional big data tends to be sparsely distributed. Usually, an attribute in one dimension could be completely determined by another dimension. In the Tablet Sales example, when *CPU="x86"*, *OS* can only be *"Windows"*. Therefore, a query specifying *CPU="x86"* and *OS="Windows"* can be reduced to a query specifying only *CPU="x86"*. Our DynamicBayes algorithm is very good at discovering such determination relationships because the PMI between such dimensions is high. By automatically grouping these columns together, BigIN4 can implicitly achieve dimension reduction on queries.

- We only consider pairwise PMI. Some situations may require calculating high dimensional PMI to better describe the complex dependency among the dimensions. However, our experiments, along with previous experiments with Bayesian Network, have shown that this scenario is not common in reality [3, 4].
- DynamicBayes assumes that for two subspaces s and S, if $s \subseteq S$, then $Q(s) \leq Q(S)$. In this way, some aggregation functions, like `MIN` and `AVG`, cannot be directly processed by DynamicBayes, nor if the data contains negative values in the measure column. Nevertheless, negative values can be converted to positive ones by adding an offset to all the values; `MIN` can be turned to `MAX` by mapping each value v to $1/v$; and `AVG` can be calculated as `SUM/COUNT`.
- We target mainly at aggregating SQL queries on a single fact table. We do not support nested queries or join queries. Moreover, BigIN4 primarily supports point queries. Although our method potentially supports range queries, we believe more optimizations remain to be done for them.

5 SUCCESS STORIES AND LESSONS LEARNED

Hundreds of big datasets from multiple Microsoft products were imported into BigIN4. People used BigIN4 include program managers, developers, testers, salespersons, DevOps, data scientists, and researchers. The sizes of data analyzed by BigIN4 range from a few gigabytes to hundreds of terabytes and their row numbers range from 10^7 to 10^{11}. Most datasets have 10 - 30 columns. The widest dataset has more than 200 columns.

BigIN4 has been adopted by multiple product teams in Microsoft. We share some success stories here:

Data Hub Team: the team maintains a large number of big datasets collected from multiple Microsoft services. Many teams perform analysis on those big datasets to improve the service quality. Before BigIN4 was deployed, users have to manually perform queries on the big datasets, wait a long time for the results, and refine their queries with the results from the previous queries. With BigIN4, users can receive instant and automatically suggested insights from their big datasets at any subspace they are interested in, making the analysis faster and more targeted. The team has put BigIN4 on their web portal as an advanced tool for big data analysis.

Windows Team: the team used BigIN4 to obtain insights for system-related datasets. Big data storage and computing are deployed on different platforms inside the team to meet different requirements, which brings great difficulty for BigIN4 deployment. However, after trying out BigIN4 on one platform, the team invested efforts to contribute a set of data adaptors in order to make BigIN4 work on other data platforms across the team.

The product teams also provided valuable feedback to improve our system. We share some lessons learned during the deployment:

First, we confirm that many users do not know where to start when faced with a new dataset, especially for big data. This is because users can obtain an overview of small data using tools like Microsoft Excel or Powerview, but obtaining an overview of big data is prohibitively slow. The insight mining feature of BigIN4 offers data scientists a good and quick starting point for analyzing the data. Although the insight types are pre-defined, they can inspire the users for further explorations. According to our interviews, users of BigIN4 sometimes need other types of insights (we provided interfaces for adding new insight types), but they also start with the basic insights BigIN4 suggests to them.

Second, a good data visualization component is critical for interactive insight identification. Apart from core algorithms, we also built a user-friendly UI that provides a drag-and-drop interface for typical data operations such as filtering, rolling-up, and breaking-down. Although UI is not the core of BigIN4, it played a critical role in improving the user experience.

Third, data cleaning is also important for an insight mining system. Many real-world datasets are diverse and noisy, which could bring many challenges to data analysis. Therefore, we implemented a data schema analysis module that gives the users hints on columns that might not fit for big data analysis (like IDs, JSONs, etc.). This feature not only ensured the stability and performance of the system but also helped with user adoption.

6 RELATED WORK

6.1 Insight Mining and Data Exploration

Data exploration is about efficiently extracting knowledge from data [2]. In the database community, several approaches have been proposed for efficient data exploration [22, 24, 27]. Our previous work [21] proposes an insight extraction solution, which can return top k interesting insights to users automatically.

Data cube plays an essential role in OLAP engines of many multi-dimensional data warehouses. Over the years, a large amount of work has been proposed to improve the effectiveness and efficiency of queries over data cubes. For example, Wu et al. [23] proposed an online aggregation system called COSMOS, which organizes queries into a dissemination graph to exploit the dependencies across queries. It also combines answers from ancestor nodes to generate the online aggregates for a node. Their work is effective in handling multiple aggregation queries, while we focus on giving answers to a single aggregation query. Riedewald et al. [17] proposed Iterative Data Cubes, which provides a modular framework for combining one-dimensional aggregation techniques to create optimal, high-dimensional data cubes. Their approach could also reduce query cost by creating iterative data cubes, while our work reduces query cost through approximate query processing.

6.2 Approximate Query Processing

Approximate Query Processing (AQP) [5, 7, 14] focuses on giving approximate answers in a very short time, which is especially meaningful for decision making, data visualization, and hypothesis testing [15]. We briefly introduce some widely used AQP approaches here.

Online Aggregation (OLA): In online aggregation, the query answers are continuously refined as more data is scanned [9]. Users can stop the query process at any time when they are satisfied with the accuracy. However, for some queries, OLA still requires a long time to find enough records before giving satisfactory answers. In interactive query answering where quick answers are expected for all kinds of queries, OLA is not a good solution.

Sampling: Sampling provides approximate answers by making one or more samples from the original data and doing queries on the sample instead of the full data. It is a simple yet powerful method because it avoids the curse of dimensionality. It can be easily applied to high dimensional data and can answer a wide range of queries with the same sample files. BlinkDB [1] is a representative AQP framework which builds a set of stratified samples for the original data. It exploits the past workloads and data sparseness when building the samples. However, when using sampling as the data synopsis method, the quality of answers relies heavily on the quality of samples. Uniform sampling works badly on skew data, and stratified sampling requires future queries known as a priori. Since query workloads are not available for newly imported data, sampling is not a proper solution for our scenario. Yan et al. [25] proposed a stratified sampling method which aims to solve measure skewness. However, their work is done on one-dimensional data and is hard to extend to high dimensional situations.

Histograms and wavelets: Histograms and wavelets are both data synopsis methods which aim to summarize records that are "close" with fewer records [5, 20]. Although they work well in one-dimensional scenarios, their complexity grows exponentially with the number of dimensions. Moreover, these synopsis methods mostly focus on minimizing global error and do not provide optimization for specific queries. As a result, some of the queries may suffer from a surprising high error rate.

Result Reusing: Recent work in AQP focuses on refining query accuracy with the results of the past queries. For example, IDEA [6] leverages several common patterns in data discovery to give answers with the past results without accessing the data again. Verdict [16] builds a model for underlying data distribution through "Database Learning". BigIN4 focuses on giving accurate approximate answers without knowledge about data and queries. It does not need any past query to estimate answers. Still, we can incorporate the idea of result reusing into BigIN4 in our future work.

7 CONCLUSION

Insight identification over big data has become increasingly important for business intelligence. In this paper, we present BigIN4, an interactive insight identification system for multi-dimensional big data. BigIN4 answers queries by combining data cube and approximate query processing techniques. The advantage of BigIN4 is that it does not require any prior knowledge of data or future queries. When a query cannot be directly answered by the cubes, BigIN4 will calculate a Bayesian Network for it, decompose the query into several low dimensional answerable queries, and then give an estimated answer to the original query. Experiments on real-world datasets have shown that BigIN4 can identify insights effectively and efficiently. It is able to provide approximate answers quickly with less than 10% error on average. BigIN4 is now used by multiple product teams in Microsoft to analyze their business data.

ACKNOWLEDGEMENT

We thank the intern students Xuewei Chen, Weizi Wang, Pu Zhao, Kaize Ding, Banghuai Li, Chenggang Li, Yuchen Sun, Mingchao Sun, and Zhenzhen Wang for the development and experiments, and our product team partners for their collaboration and suggestions on the applications of BigIN4. We also thank all the members of Software Analytics team at MSRA for the discussions.

REFERENCES

[1] Sameer Agarwal, Barzan Mozafari, Aurojit Panda, Henry Milner, Samuel Madden, and Ion Stoica. 2013. BlinkDB: queries with bounded errors and bounded response times on very large data. In *Proc. of the 8th ACM European Conference on Computer Systems*. ACM, 29–42.

[2] Michael Armbrust, Reynold S Xin, Cheng Lian, Yin Huai, Davies Liu, Joseph K Bradley, Xiangrui Meng, Tomer Kaftan, Michael J Franklin, Ali Ghodsi, et al. 2015. Spark sql: Relational data processing in spark. In *Proc. SIGMOD '15*. ACM, 1383–1394.

[3] Jie Cheng, Russell Greiner, Jonathan Kelly, David Bell, and Weiru Liu. 2002. Learning Bayesian networks from data: an information-theory based approach. *Artificial intelligence* 137, 1-2 (2002), 43–90.

[4] C Chow and Cong Liu. 1968. Approximating discrete probability distributions with dependence trees. *IEEE transactions on Information Theory* 14, 3 (1968), 462–467.

[5] Graham Cormode, Minos Garofalakis, Peter J Haas, and Chris Jermaine. 2012. Synopses for massive data: Samples, histograms, wavelets, sketches. *Foundations and Trends in Databases* 4, 1–3 (2012), 1–294.

[6] Alex Galakatos, Andrew Crotty, Emanuel Zgraggen, Carsten Binnig, and Tim Kraska. 2017. Revisiting reuse for approximate query processing. *Proc. VLDB Endowment* 10, 10 (2017), 1142–1153.

[7] Minos N Garofalakis and Phillip B Gibbons. 2001. Approximate Query Processing: Taming the TeraBytes.. In *VLDB*.

[8] Jiawei Han, Jian Pei, Guozhu Dong, and Ke Wang. 2001. Efficient computation of iceberg cubes with complex measures. In *ACM SIGMOD Record*, Vol. 30. 1–12.

[9] Joseph M Hellerstein, Peter J Haas, and Helen J Wang. 1997. Online aggregation. In *Acm Sigmod Record*, Vol. 26. ACM, 171–182.

[10] Prasanth Jayachandran, Karthik Tunga, Niranjan Kamat, and Arnab Nandi. 2014. Combining user interaction, speculative query execution and sampling in the DICE system. *Proc. VLDB Endowment* 7, 13 (2014), 1697–1700.

[11] Niranjan Kamat, Prasanth Jayachandran, Karthik Tunga, and Arnab Nandi. 2014. Distributed and interactive cube exploration. In *Proc. ICDE 2014*. IEEE, 472–483.

[12] Micheline Kamber, Jiawei Han, and Jenny Chiang. 1997. Metarule-guided mining of multi-dimensional association rules using data cubes.. In *KDD*, Vol. 97. 207.

[13] Xiaolei Li, Jiawei Han, and Hector Gonzalez. 2004. High-dimensional OLAP: A minimal cubing approach. In *Proc. VLDB '04*. VLDB Endowment, 528–539.

[14] Barzan Mozafari. 2017. Approximate query engines: Commercial challenges and research opportunities. In *Proc. SIGMOD '17*. ACM, 521–524.

[15] Barzan Mozafari and Ning Niu. 2015. A Handbook for Building an Approximate Query Engine. *IEEE Data Eng. Bull.* 38, 3 (2015), 3–29.

[16] Yongjoo Park, Ahmad Shahab Tajik, Michael Cafarella, and Barzan Mozafari. 2017. Database learning: Toward a database that becomes smarter every time. In *Proc. SIGMOD '17*. ACM, 587–602.

[17] Mirek Riedewald and Divyakant Agrawal. 2005. Dynamic Multidimensional Data Cubes for Interactive Analysis of Massive Datasets. In *Encyclopedia of Information Science and Technology, First Edition*. IGI Global, 924–929.

[18] Mirek Riedewald, Divyakant Agrawal, and Amr El Abbadi. 2001. Flexible data cubes for online aggregation. In *Proc. ICDT '01*. Springer, 159–173.

[19] Mirek Riedewald, Divyakant Agrawal, and Amr El Abbadi. 2002. Managing and analyzing massive data sets with data cubes. In *Handbook of massive data sets*. Springer, 547–578.

[20] Cristina Sirangelo. 2005. *Approximate Query Answering on Multi-dimensional Data*. Ph.D. Dissertation. PhD Thesis, University of Calabria.

[21] Bo Tang, Shi Han, Man Lung Yiu, Rui Ding, and Dongmei Zhang. 2017. Extracting top-k insights from multi-dimensional data. In *Proc. SIGMOD '17*. 1509–1524.

[22] Hunter Whitney. 2012. *Data insights: new ways to visualize and make sense of data*. Morgan Kaufmann.

[23] Sai Wu, Beng Chin Ooi, and Kian-Lee Tan. 2010. Continuous sampling for online aggregation over multiple queries. In *Proc. SIGMOD '10*. ACM, 651–662.

[24] Tianyi Wu, Dong Xin, Qiaozhu Mei, and Jiawei Han. 2009. Promotion analysis in multi-dimensional space. *Proc. VLDB Endowment* 2, 1 (2009), 109–120.

[25] Ying Yan, Liang Jeff Chen, and Zheng Zhang. 2014. Error-bounded sampling for analytics on big sparse data. *Proc. VLDB Endowment* 7, 13 (2014), 1508–1519.

[26] Kai Zeng, Sameer Agarwal, Ankur Dave, Michael Armbrust, and Ion Stoica. 2015. G-ola: Generalized on-line aggregation for interactive analysis on big data. In *Proc. SIGMOD '15*. ACM, 913–918.

[27] Yan Zhang and Yiyu Jia. 2011. EProbe: An efficient subspace probing framework. In *ICTAI, 2011 23rd IEEE International Conference on*. IEEE, 841–848.

[28] Peixiang Zhao, Xiaolei Li, Dong Xin, and Jiawei Han. 2011. Graph cube: on warehousing and OLAP multidimensional networks. In *Proc. SIGMOD '11*. ACM, 853–864.