18th International Conference on Information Fusion
Washington, DC - July 6-9, 2015

Business Data Fusion

Surya Yadav, Gautam Shroff, Ehtesham Hassan and Puneet Agarwal
TCS Innovation Labs, New Delhi, India

***Abstract*—Enterprise business intelligence usually relies on data from multiple sources being carefully joined based on common attributes and consolidated into a common data warehouse. This process is often plagued by difficulties and errors in resolving join-attributes across sources while consolidating information into a data warehouse. Moreover, it may often be impossible to accurately join data from diverse external data sources. Nevertheless, each such data source can still provide useful information on correlations amongst the attributes it captures, and enterprises are increasingly looking to replace the traditional data warehouse with 'data lakes' based on new technology, such as Hadoop, in order to derive statistical insights. We describe an approach for 'business data fusion' applicable in such a scenario: We define 'distributional queries' and their utility in multiple scenarios, including for correlating diverse data sources, and show that these are equivalent to probabilistic inference. In order to efficiently execute such queries, relationships and correlations across data sources are summarized via a Bayesian network, which is learned in an expert-guided manner so as to incorporate domain knowledge. We present empirical results of our approach applied to (a) summarize large volumes of vehicular multi-sensor data in a sensor-data-lake, to efficiently provide probabilistic answers to support engineering analysis without repeatedly accessing the raw data; and (b) demonstrate how potentially diverse and unrelated public and private data sources can nevertheless be approximately and efficiently joined to derive useful statistical insights via distributional queries implemented using Bayesian inference.**

I. Introduction & Motivation

The traditional analytics life-cycle in large enterprises goes as follows: operational data is extracted from transactional systems, such as point-of-sale, inventory etc., and loaded into a data warehouse, undergoing a multitude of transformations, including de-normalization, aggregations, dropping attributes, etc. Great care is taken to ensure that related pieces of data can be joined using common attributes, thereby enabling 'business intelligence' queries on the data warehouse, or on subsets called 'data marts'. Sometimes the analytics life-cycle ends here, with operational and strategic decisions being supported by querying the past data, or 'looking in the rear view mirror'.

Of course, in order to look ahead, organizations increasingly employ predictive analytics to varying degrees, using statistics, data mining, and machine learning techniques. For such purposes, selected slices (or 'data cubes') are extracted using relational (i.e., SQL) queries and loaded into statistical analysis tools such as SAS or SPSS, in order to perform regressions, time-series forecasting, or similar predictive analyses to support 'looking ahead'.

In sharp contrast, modern 'web companies' such as Google or Facebook follow a different approach. Data is maintained in a large distributed file system, typically based on 'big-data' technologies such as Hadoop, or Google's GFS: There is no concept of a carefully curated data warehouse. Analytical queries, be they relational or statistical in nature are carried out directly on this common data store, typically in parallel using the map-reduce programming paradigm and its many extensions: Such techniques have been shown to be better suited for statistical processing and queries, which often touch large tracts of the data, as compared to say traditional index-based database query systems.

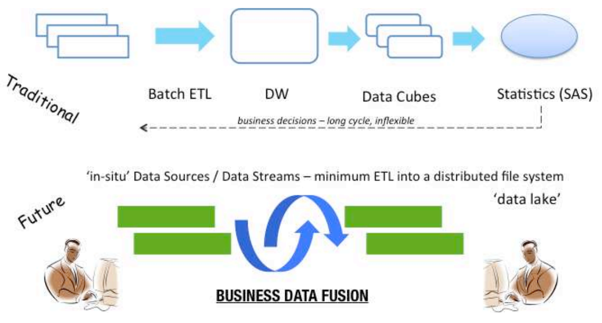

Traditional
Batch ETL
DW
Data Cubes
Statistics (SAS)
business decisions – long cycle, inflexible
'in-situ' Data Sources / Data Streams – minimum ETL into a distributed file system
Future
'data lake'
BUSINESS DATA FUSION

Fig. 1. Traditional BI vs Data Lake

The above approach has many advantages: The time taken to design and implement a large enterprise data warehouse is often measured in years; this is saved completely. Next, predictive analytics can be performed on the entire data rather than only selected subsets that are limited in size by the inherent in-memory architecture of most statistical packages. Moreover, the results of such analysis are themselves stored in the same global data store, and can be used by others as inputs for further analysis. Finally, additional data elements, be they fresh data sources or merely new attributes can be easily added without having to worry about the integrity of a common data warehouse schema.

For the above reasons, many traditional enterprises such as retail chains, banks, and manufacturers are beginning to employ an analytics architecture similar to that used by the web companies, loosely referred to as a **data lake** (as opposed to the more traditional data warehouse). Figure 1 illustrates the difference between a traditional BI life-cycle and a data-lake.

Still, fundamental challenges still remain, for which the data lake alone is no panacea: (i) Even if disparate data sources are stored in a data lake, the issue of how these should be joined together remain. (This has prompted some prominent database researchers to refer to the data-lake phenomenon as a *data swamp* [1].) (ii) Further, when data volumes are very large, queries can still take inordinate amounts of time unless backed by sufficiently powerful hardware. (iii) Lastly, as the number of attributes grows the well known problems of high dimensionality appear: In particular, data cubes defined by very selective constraints on a large number of attributes may be empty, i.e., there may not be any actual instances observed in the data available. Nevertheless, conclusions about such subsets are in fact possible using statistical rather than query-based analysis.

II. Key Contributions

In this paper we describe a **business data fusion** architecture to address each of the challenges described above. We pre-processes data stored in a data-lake based on a Bayesian network defined on all the attributes involved in the data, possibly spanning disparate sources. Each attribute forms a random variable in this Bayesian network. (The network itself can be either crafted using domain-based understanding of dependencies and correlations, or using traditional structure-learning approaches, or a combination.)

Attributes in disparate data sources that can be directly mapped to each other are assigned to the same network variable. For example, items or parts in different data sources that are described by an industry-standard coding scheme could be treated as a single item variable. Attributes that can only be related approximately are retained as separate network variables, with their mutual correlations being captured in the conditional probabilities of the Bayesian network. For example, different surveys might have captured data using different spatial tesselations of geographical regions, e.g. counties vs zip codes; nevertheless, mutual overlaps between such regions can be computed using maps, i.e., GIS layers. Thus the conditional probability of a particular county lying in each possible zip code can be computed using a map. Similarly, product-related data codified using different product-category definitions could be approximately related using *sample* instances that are codified across multiple categorizations.

Conditional probabilities in the Bayesian network are computed via pre-processing. Queries are performed on the network using probabilistic inference. Wherever possible, a tree-structured network is used, even if it is only an approximation to the actual conditional independence relationships between attributes: In a tree-structured Bayes net each conditional probability table is two-dimensional, making inference efficient. In many ways, the pre-processing for computing conditional probabilities is similar to calculating materialized views in a database, albeit in a probabilistic manner.

Our approach address the main challenges described in the previous section as follows: (i) Attributes in disparate data sources are approximately joined in a principled manner using conditional probabilities derived from other appropriate sources as available, e.g., maps, sample categorizations, etc. (ii) When volumes are large, compressing data into conditional probability tables accelerates query times in much the same manner as materialized views; (However, our approach additionally incorporates approximate correlations between attributes as well as can rely on an approximate, e.g. tree-structured, network to further minimize query execution. Of course, unlike a database, we only provide approximate results.) (iii) Our approach obviates the problem of there being possibly no actual instances satisfying complex query conditions: probabilistic inference will always yield an answer, which is the best probabilities answer possible based on the assumptions underlying the Bayesian network itself. In summary our approach for **business data fusion** using Baeysian networks allows for approximate joins, accelerates query processing on voluminous, i.e., 'long' data sets, as well as addresses the obvious challenges faced in querying very 'wide' data.

We submit that when fast, approximate results are sufficient (as is the case in predictive analytics as opposed to regulatory reporting) our approach makes the enterprise data lake a viable and useful proposition as it marries approximate data mapping and probabilistic query processing in one framework based on the well established principles of Bayesian inference.

The remainder of the paper is organized as follows: We begin by placing our contributions in the context of related work in Section III. Next we formally describe the business data fusion problem in Section IV, by introducing the concept of 'distributional queries', with examples, and showing their equivalence to probabilistic inference. We describe our approach to efficiently compute distributional queries via Bayesian networks in Section V. In Sections VI-A and VI-B we describe performance and accuracy of our technique with experimental results for two cases: First, a real-life scenario of large-scale sensor data analysis where our technique was used to enable real-time 'what-if' queries without touching the raw data. Next, we describe a hypothetical scenario where disparate data sources that are not naturally related could nevertheless be approximately joined to provide insight. We conclude in Section VII by summarizing our results and also suggesting how to extend our approach to cover data mining in addition to query processing.

III. Related Works

Fusion of data from multiple sensors [2], [3], [4], [5], [6] has been addressed via models such as JDL[7], Dempster-Shafer model [8] as well as Bayesian models[9]. However, this stream of work is focused towards object/target identification, threat assessment in defence arena, or classification. Ye et al.[10] in the field of bio-informatics have attempted joining multiple datasets using kernel learning, for diagnosis of Alzemier disease but targeted at classification. In contrast, we focus on joining seemingly disparate datasets so as to be able to answer 'distributional' queries, as defined above.

Many aspects of our approach is similar to using materialized views [11] to speed up subsequent query processing: A materialized view calculates and stores sets of aggregations on joins of multiple tables, whereas in our approach we compute conditional probability tables as parameters in a Bayesian network. Materialized views aim at efficiently but exactly answering a specified class of relational queries: In the approach proposed by Harinarayan et al. [11], specific cells of a data-cube are pre-computed so that a large set of queries can be exactly and efficiently computed. In contrast, we focus on being able to approximately answer any distributional query on a collection of datasets. Additionally, because our approach employs Bayesian data fusion, we return a posterior conditional distribution even in cases where there are no records in the collection of datasets satisfying all the query conditions.

Our approach is most similar to that of Margaritis et al. in [12] who also use Bayesian inferencing to answer database queries, while observing that Bayesian inference also underlies cost-based query optimization *inside* many database systems. We extend their work by introducing fusion of seemingly disparate datasets that have no exact join key or mapping between tuples. Additionally we further employ a SQL engine, albeit on a database of parameter tables, to execute probabilistic inference: thus also pointing to possible connections between how SQL queries are executed efficiently and techniques for inference on Bayesian networks.

A comprehensive survey of data fusion approaches has been given in [13], [14], however they assume the availability of a join key between target datasets, or an exact mapping table between the join keys of target datasets. Data fusion in the GIS field is concerned with matching of images taken at different from different angles and at different times [15], or with multi source classification [4]. Fusion of seemingly disparate datasets, when an exact mapping table is not available, has also performed using multiple different techniques such as, using approximate string match [16] or using logical constraints [17].

By focusing on distributional queries that are equivalent to probabilistic inference, we submit that our approach takes a step towards bringing together the database and machine-learning approaches to deriving insights from data, and in doing so also provides a lens through which to also correlate seemingly disparate datasets, assuming they each comprises sample instances from a common underlying population of object or entities.

IV. The Problem

A. Distributional queries on database

Consider the scenario of data from multiple sensors, such as those now commonly present in most modern vehicles, aircraft and similar complex machinery, as illustrated in Table I: Each record of raw data is tuple of real numbered sensor values. In discretized form, i.e., as D in the table, real values of each sensor are converted to bin numbers; for example, the ES sensor might be discretized using ten bins - 100 to 200, 200 to 300, etc. (Note: exactly such a dataset has been discussed in detail in the Section VI.)

Engineering analysis of such data usually involves computing and visualizing the distribution of a particular sensor's values, i.e., how often each bin is populated in the data. Equally interesting is the joint distribution of two sensors, i.e., how often each pair of bin-values are populated, resulting in a two dimensional display. (Figures 4 illustrate such distributions.)

TABLE I
Example 1: Vehicle sensor data

Raw Data					D: Discretized Data				
t	ES	TF	NT	CP	t	ES	TF	NT	CP
10	701	56.91	3.36	8	10	7	6	1	8
11	702.3	57.69	10.34	8	11	7	7	5	8
12	698.4	58.63	14.91	8	12	6	8	5	8
13	697	59.41	19.73	8	13	6	9	7	8
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮
99	700.4	57.84	6.84	11	99	7	7	2	11

More importantly, it is especially of interest to determine the distribution of one or more target set of sensors given certain conditions on the remaining sensors: For example, the distribution of TF (total fuel consumption) when ES (engine speed) is low but NT (net torque) is high.

In general, let the dataset be denoted as $D(\mathbf{a})$, where **a** denotes M attributes $a_1 \dots a_M$; in the above, these correspond to different sensors. Further, suppose each attribute a_i takes n_i discrete values, $b_{i1} \dots b_{in_i}$. The computation of a distribution for attribute a_i can be expressed in relational algebra as computing n_i queries on D of the form

$$G_{count()}\sigma_{[a_i=b_{ij}]}D \tag{1}$$

for each $j = 1 \dots n_i$. Here, $G_{count()}$ refers to aggregate function $count()$ under a select operation(σ) with condition $a_i = b_{ij}$.

More generally, for k target attributes $a_{i_1} \dots a_{i_k}$, a k dimensional **distributional query** under conditions Q, where Q specifies a set of conditions on some or all of the remaining $n - k$ attributes can be computed by executing $n_{i_1} \times n_{i_2} \times \dots n_{i_k}$ queries, each of the form:

$$G_{count()}\sigma_{[a_{i_1}=b_{i_1 j_1}, a_{i_2}=b_{i_2 j_2}, \dots, a_{i_k}=b_{i_k j_k}, Q]}D \tag{2}$$

for each possible combination of $j_1 \dots j_k$, i.e., where every j_l can take n_l values $1 \dots n_l$.

For example the distribution of ES sensor under the conditions that TF lies in its bin 7 or 8, and CP lies in its bin 5, can be obtained by evaluating ten relational queries of form:

$$G_{count()}\sigma_{[ES=j, TF\in[7,8], CP=5]}D \tag{3}$$

for the ten bins of ES, $j = 1 \dots 10$ (here we have assumed that ES has been discretized into ten bins).

Note that each set of distributional queries can be computed in a single pass over the entire database, or using an index of some form if the condition Q is highly selective. Of course, when data volumes become very large, having to access the data for each query becomes a significant overhead, especially when queries are not highly selective, making indexes irrelevant and necessitating a scan through the entire data. (Further, as pointed out by Jacobs [18], often even loading datasets in to a traditional database is not worth the benefit of rapid querying using indexes.)

Joining Multiple Datasets: In the above scenario, all the data was present in a single table. In case data comes from diverse sources, an additional complication arises, i.e., that of joining different sources based on common or related attributes. Consider the example of data of interest for a marketing professional: e.g., income from census data(D_1), profession from a marketing survey(D_2), and location of a person over time from a mobile operator(D_3), as shown is Table II, details of which are discussed later in Section VI. (Note that dataset may have been collected by surveying or monitoring a different set of people, albeit in the same overall geography, i.e., the data represents the same underlying 'ground truth', and each sample is assumed to be equally unbiased.)

In this scenario, we have three databases D_1, D_2, and D_3 representing the income, location, profession of people in different regions, captured by the attributes R_1, R_2 and R_3. NOTE however, even if the same geographical segmentation is used for each of the region attributes, we cannot use this common region attribute to meaningfully join the three tables, since each of the joins $D = D_1 \bowtie_{R_1=R_2} D_2$, $D_2 \bowtie_{R_2=R_3} D_3$, or $D_1 \bowtie_{R_1=R_3} D_3$ are many-many relationships. So, while these joins are defined, they do not serve any meaningful purpose.

TABLE II
EXAMPLE 2: DISPARATE MARKETING DATASETS

D_1		D_2		D_3	
R_1	Income	R_2	Location	R_3	Profession
1	110089	2	Resturant	1	Lawyer
2	116702	1	School	3	Surgeon
1	103868	2	Hotel	3	Architect
1	135433	2	School	2	Farmer
⋮	⋮	⋮	⋮	⋮	⋮
2	147453	1	Shop	2	Architect

Instead, what we are really interested in is some mechanism of querying a *hypothetical* joined dataset D, such as depicted in Table III, which might have been possible to compute if each of the tables had a common attribute that uniquely identifying an individual. Unfortunately, no such attribute is available.

We submit that it is still possible to derive meaningful insights from such disparate datasets, albeit under some strong assumptions. Suppose we view each of the datasets D_i as *random* samples from the hypothetical distribution D. Unfortunately, each such sample is itself incomplete, with two out of three attributes missing, as shown in Table III. What is the best that can be done, if anything at all?

Note that this is an extreme example of a missing-data problem, commonly encountered in machine-learning. Such situations are usually tackled by attempting to fill in the missing values as best as possible using other attributes present. So, for example, one might try to fill in the Location and Profession columns by computing those that maximize the conditional probability $P(Location, Profession|Income, Region)$. In general, we try to fill in missing attributes with the combination that maximizes their joint probability conditioned on

TABLE III
EXAMPLE 2: HYPOTHETICAL JOINED DATASET

D			
R	Income	Location	Profession
1	110089	?	?
2	116702	?	?
1	103868	?	?
1	135433	?	?
⋮	⋮	⋮	⋮
2	?	Restaurant	?
1	?	School	?
2	?	Hotel	?
2	?	School	?
⋮	⋮	⋮	⋮
1	110089	?	Lawyer
3	116702	?	Surgeon
3	103868	?	Architect
2	135433	?	Farmer
⋮	⋮	⋮	⋮

whatever attributes are observed:

$$\arg\max P(\forall a_i \in \mathcal{M} | \forall a_j \in \mathcal{O}) \quad (4)$$

where $\mathcal{M}, \mathcal{O}$ stand for the missing and observed attributes respectively.

In practice, this may be difficult to compute without even further assumptions. Nevertheless, supposed we are able to assume that all the attributes of interest are independent of *each other*, given those that are observed: (In our example above, this translates to Location, and Profession being independent of each other at least *within* each Income, Region combination. In other words, any correlations between the missing variables can be explained by those that are observed (this not neccesarily a valid assumption, of course). This is the *naive Bayes* assumption commonly used in data fusion and machine learning, using which (4) becomes:

$$\arg\max \Pi_{\forall i \in \mathcal{M}} P(a_i | \forall a_i \in \mathcal{O}) P(\forall a_i \in \mathcal{O}) \quad (5)$$

In our specific example this means filling in missing values by maximizing the product of $P(P|I,R)$, $P(P|I,R)$ and $P(I,R)$ instead of $P(L,R|R,R)$ (P, L, I, R being Profession, Location, Income and Region, respectively).

Incongruous Join Keys: Further, it is important to note that if each of the datasets D_i are truly disparate, each dataset may use *different* geographical segmentations for their respective region attributes. So $R1 = 1$ and $R2 = 1$ may not in fact be the same region. Further, one dataset may have used of regions coarser than the other, and regions from different datasets may overlap with each other in practice, e.g. zip codes and counties. In such a situation the above procedure for filling in missing values of D also breaks down.

Nevertheless, provided additional data is available, we can still proceed: For example, we might augment these three datasets with an additional dataset that helps derive the relationship between the different region attributes. Such a dataset could, for example, be derived from a map, where all the regions are marked out. We randomly sample points on such a map, noting for each point all three of its region values, i.e., according to $R1, R2$ and $R3$. The joined table D would now have additional rows where only these three region attributes are filled, and such rows would provide the link for estimating missing values. However, since the linkages are now indirect, and, as we show below, it is easier to address the problem using full probabilistic inference rather than via first filling in missing values followed by executing relational queries.

In general we can formulate the problem scenario as follows: Given datasets D_i's, we assume them to have some level of overlap in terms of attributes $\mathbf{a}_i$: For every i there exists a j, such that $\mathbf{a}_i \cap \mathbf{a}_j \neq \phi$, as seen in the above example. Further, if this is not true i.e., there is no explicit overlap between two or more datasets, we assume that we can augment our data with additional datasets so that this is the case.

Last but not least, we also assume that our collection of datasets are fully *connected*: If we define a graph having the D_is as nodes and an edge between D_i and D_j if these share at least one common attribute, then the collections of datasets are *connected* if this graph is fully connected, i.e., comprises of one connected component. In case this is not true, we once again assume that we can augment our collection of datasets to make it true.

We concern ourselves with distributional queries on a connected collection of datasets, where the attributes of each dataset have been discretized. As in the example above, such a collection can be viewed as comprising of independent samples of a 'joined' dataset D, where a chain of common attributes connects the tables.

B. *Distributional queries as probabilistic inference:*

As already mentioned, we can view the joined dataset $D(\mathbf{a})$ of Table III in the discussion above as comprising of samples from a joint distribution $P(\mathbf{a})$ across random variables corresponding to the attributes $\mathbf{a}$.

Distributional queries are then equivalent to probabilistic inference, i.e., the set of queries (1) can be viewed as computing the probability distribution $P(a_i)$ by marginalizing the joint distribution $P(\mathbf{a})$. Similarly, the general distributional query executed by the set (2) is computing the conditional posterior probability:

$$P(a_{i_1}, a_{i_2}, \ldots, a_{i_k} | Q) \tag{6}$$

This can be computed from the joint distribution P by marginalization under the evidence Q, i.e., classical probabilistic inference:

$$P(a_1 \ldots a_k | Q) \approx \sum_{\mathbf{a} \setminus \{a_1, \ldots a_k\}, Q} P(\mathbf{a}) \tag{7}$$

($\approx$ since this needs to be normalized).

Since the datasets are 'connected', and under appropriate assumptions of conditional independence, some coming from domain knowledge, and others forced upon us by the circumstances of what datasets D_i are actually available to us, we can factor the joint distribution in (6) using a single Bayesian network so

$$P(\mathbf{a}) = \Pi_i P(a_i | \mathtt{Pa}(a_i)) \tag{8}$$

where $\mathtt{Pa}(a_i)$ denotes the parents of variable a_i in the Bayesian network. We can combine (6) and (8) above to efficiently compute distributional queries, as detailed later in Section V.

C. *Querying for values using distributions*

Note that once we view the problem of distributional queries from the perspective of inference on the joint probability distribution over attributes $\mathbf{a}$, we can also answer value-based queries for continuous variables. This is required in many situations, for example instead of the two dimensional join distribution of NT and TF, we might want to see a distribution of NT vs TF, i.e., the actual values of NT for different TF values. This is the kind of query a traditional scatter-plot might convey, or a traditional 'business intelligence' query on the 'average NT for each bin of TF'.

Well, in the language of probability, such queries are easily expressed in terms of expected values, which can be computed using distributional queries: The average NT for each bin b_{TF_i} of TF can be expressed as $E[NT | TF = b_{TF_i}]$, which is computable from the conditional distribution of NT given TF as follows:

$$\sum_j v(b_{NT_j}) P(NT = b_{NT_j} | TF = b_{TF_i}) \tag{9}$$

where $v(b_{NT_j})$ is say, the midpoint of bin b_{NT_j} of NT; e.g. if $b_{NT_j} \equiv NT_1 < NT < NT_2$, then $v(b_{NT_j}) = (NT_1 + NT_2)/2$. Similarly, in the case of categorical variables, it is easily possible to compute the most likely location for a person of high income to visit by maximizing:

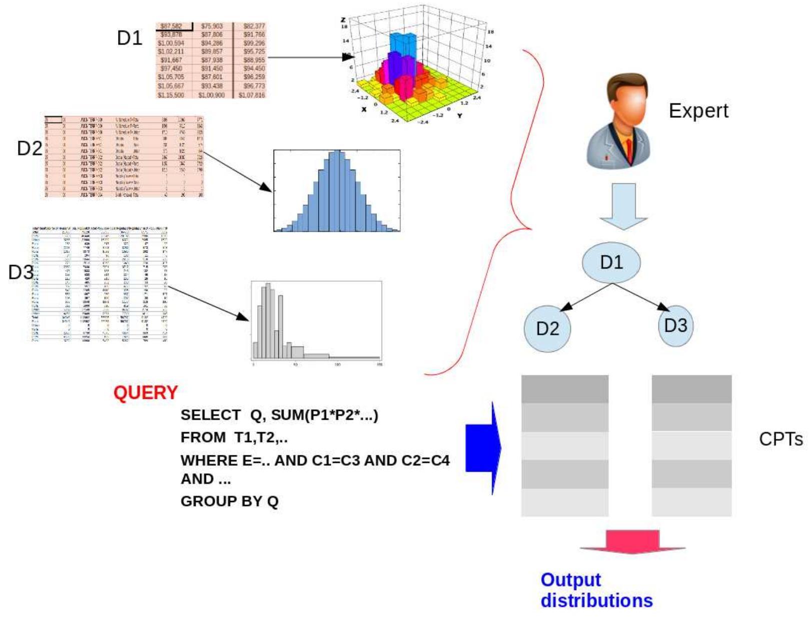
$$\arg \max_{Location} P(Location | Income \in [100000, 130000]) \tag{10}$$

V. Our Approach

A distributional query on a collection of datasets essentially can be computed via conditional inference on the joint probability distribution of attributes $P(\mathbf{a})$. Note that, in case of diverse data sources as discussed in marketing data example in the previous section, the attribute set $\mathbf{a}$ would have been suitably augmented with additional attributes, to ensure that the collection of datasets is connected.

We model a Bayesian network (BN) approximating the joint distribution, with each node representing an attribute. In the case of diverse datasets, the structure of such a BN may constrained by the limited conditional distributions computable using the available datasets D_i. If D is available fully, we construct a BN using domain knowledge along with constraints on its structure that make it easy to evaluate, e.g., ensuring it is close as possible to a tree, etc.

For probabilistic inferencing on this BN, we translate conditional queries into SQL via an rather obvious technique that nevertheless is not often used: We use SQL for querying the CPTs considering them as set of relational tables. This approach basically presents a novel method for fusion of



```
SELECT  Q, SUM(P1*P2*...)
FROM  T1,T2,..
WHERE E=.. AND C1=C3 AND C2=C4
AND ...
GROUP BY Q
```

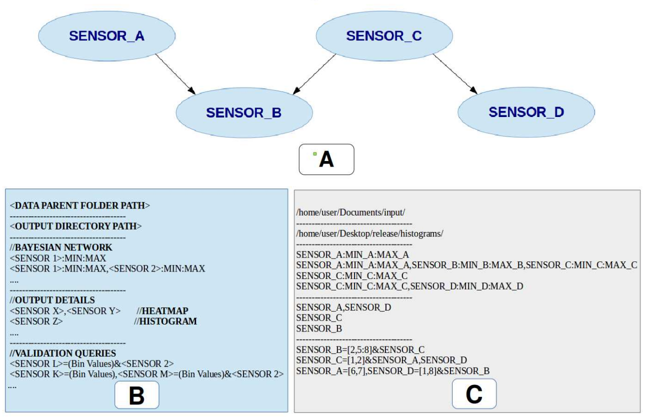
Fig. 2. Business data fusion architecture

probabilistic models using SQL engine. Note that this is not necessarily efficient unless the network structure is simple, nevertheless, we are presenting what has been used for our experimental system.

Since the BN represents an approximation to the joint distribution structure, a suitable mechanism is required for measuring the network accuracy. For validating the network structure, we compute the estimation error for selected set of typical distributional queries. Figure 2 shows the overall view of the our business data fusion architecture. The details of each block is presented in the following discussion.

A. *Computing parameters of the Bayesian network*

We anticipate the subject matter expert to specify the BN based on domain constraints. This structure and other basic details are specified in a configuration file as shown in Figure 3. Here, we show a sample BN in Figure 3.A, template for a configuration file in Figure 3.B, and sample configuration file in Figure 3.C for the BN shown in Figure 3.A. In the BN details section of this file, the details of parent sensors are given before the dependent sensor. Based on such a configuration file and observed data, we learn the conditional probability tables(CPTs) using aggregation queries similar to the one shown in Eq. 1. Here, it should be noted that in some cases the size of these CPTs can become exponentially large, we therefore limit the number of parent nodes of a node, to a maximum of three in the BN. Also, as far as possible, we model the BN as tree rather than a DAG to keep the size of the CPTs small. The conditional probability tables for each edge of the BN are easily calculated using the original data. We take Dirichlet distribution as the conjugate prior and its parameters are specified by the user.



```
<DATA PARENT FOLDER PATH>
-----------------------------------
<OUTPUT DIRECTORY PATH>
-----------------------------------
//BAYESIAN NETWORK
<SENSOR 1>:MIN:MAX
<SENSOR 1>:MIN:MAX,<SENSOR 2>:MIN:MAX
....
-----------------------------------
//OUTPUT DETAILS
<SENSOR X>,<SENSOR Y>      //HEATMAP
<SENSOR Z>                 //HISTOGRAM
....
-----------------------------------
//VALIDATION QUERIES
<SENSOR L>=(Bin Values)&<SENSOR 2>
<SENSOR K>=(Bin Values),<SENSOR M>=(Bin Values)&<SENSOR 2>
....
```

```
/home/user/Documents/input/
-----------------------------------
/home/user/Desktop/release/histograms/
-----------------------------------
SENSOR_A:MIN_A:MAX_A
SENSOR_A:MIN_A:MAX_A,SENSOR_B:MIN_B:MAX_B,SENSOR_C:MIN_C:MAX_C
SENSOR_C:MIN_C:MAX_C
SENSOR_C:MIN_C:MAX_C,SENSOR_D:MIN_D:MAX_D
-----------------------------------
SENSOR_A,SENSOR_D
SENSOR_C
SENSOR_B
-----------------------------------
SENSOR_B=[2,5:8]&SENSOR_C
SENSOR_C=[1,2]&SENSOR_A,SENSOR_D
SENSOR_A=[6,7],SENSOR_D=[1,8]&SENSOR_B
```

Fig. 3. Bayesian network configuration

B. *Using SQL for conditional queries on a Bayesian network*

The conditional probabilities learned from data are stored in a SQL database and all further probabilistic queries are then answered using this database. For example, if we translate the relational expression described in equation 3 on the vehicle sensor dataset of Table I into probabilistic inference , it amounts to computing a set of conditional probability queries:

$$P(ES = j, |TF \in [7, 8], CP = 5]) \tag{11}$$

which are then multiplied and marginalized according to (8) and (6) respectively.

The product (8) and sum (6) can be computed using SQL on the database of CPTs, for the Bayesian network in Figure 5 as follows:

```
SELECT ES, SUM (P_TF*P_NT*P_ES*P_CP)
 FROM T_TF,T_NT,T_ES,T_CP
 WHERE TF IN (7,8) AND CP = 5
GROUP BY ES
```

Here ES is queried variable, and `T_TF`, `T_NT`, `T_ES`, `T_CP` are the CPTs for TF, NT, ES, CP based on the Bayesian network (shown later in the Section VI-A). `P_TF`, `P_NT`, `P_ES` and `P_CP` are the respective probability column names, and the `WHERE` clause defines the given conditions. Using similar query, we can also compute the evidence of given conditions as shown below:

```
SELECT SUM (P_TF*P_NT*P_ES*P_CP)
 FROM T_TF,T_NT,T_ES,T_CP
 WHERE TF IN (7,8) AND CP = 5
```

where the notations are similar to the ones given above.

C. Validation of Bayesian network

As we are using probabilistic inference there is bound to be some difference between the queries when executed on the raw data and the queries executed using BN. So we give some validation queries as input in config file for measuring the error bounds. We find the output of the queries from the network and from the raw data and display the results. Then we compare the two derived distributions using KL divergence and Bhattacharyya coefficient. Let the distribution obtained from data be Y and the distribution obtained from network be Z. Since Y and Z are both discrete distributions the symmetric KL divergence between them as $D_{KL}(Y,Z) = (D_{KL}(Y \parallel Z) + D_{KL}(Z \parallel Y)/2$, where $D_{KL}(Y \parallel Z) = \sum_i Y(i) \ln \frac{Y(i)}{Z(i)}$. Also, the Bhattacharyya coefficient between Y and Z is given by $BC(Y,Z) = \sum_i \sqrt{Y(i) * Z(i)}$.

VI. Experimental Analysis

We evaluate the proposed approach of business data fusion on two scenarios described as datasets, Tables I and II. The objective is to show that this approach can provide an efficient solution for distributional querying on multiple and diverse datasets. Since the approach primarily depends on domain knowledge supplied bayesian network, the focus of experimental evaluation is on the computation of validation errors in case of set of sample querying describing different conditions. The related computational analysis presented here onwards have been obtained on an Intel Core i5 workstation with 3.2 GHz speed and 4GB RAM.

A. Distributional queries on Vehicle Sensor Data

As the machines, be they vehicles, engines or any other equipment become more and more complex they are increasingly being fitted with multiple, often hundreds of sensors. Analyzing the voluminous data produced by populations of vehicles so outfitted allows manufactures to better understand the behavior of their products in the field as well as exactly how their customers use them; information that is invaluable in determining reasons for abnormal behavior leading to faults, finding opportunities for improving design, etc.

We used a large collection of sensor data for multiple instances of an engine. The engines had more than two hundred sensors and for each sensor readings were taken for an average of half-hour run of the engine. The data consisted of such runs for over a year and was stored in csv format. It had more than twelve million records and was 15GB of size in total. This data was first converted to binary form in order to speed up the process of reading of the data. This led to data being compressed to 10GB. The binary files were then used as input data for learning the parameters of the Bayesian network, and querying on raw data.

We demonstrate our business data fusion approach for the above use-case of real-life sensor data, with four of the sensors, viz., engine speed (ES), torque (NT), total fueling (TF) and combustion control path owner (CP). As our approach requires discretized sensor values, we represent these values as set of 1-D and 2-D histograms computed as the process of statistical profiling of the data. Figure 4 shows the screenshot of these histograms in our visual analytics workbench. The Bayesian network defining connections between these sensors is shown in the Figure 5.

Figure 4 illustrates our business data fusion system in operation: An instance of distributional query can be described by range selection on selected sensors; the system highlights the initial and post-query distribution on remaining sensors. The system also shows probability of the evidence, i.e., the selections, as a vertical bar, indicating the support of the selected conditions in terms of probability.

For validation, a number of queries containing one to four sensors were formulated. These queries were then executed on the raw data and on BN. We observed an average time of $10 \sim 12$ seconds without considering the data reading time in memory (which is of 4 minutes in the present environment). The same queries when executed on the BN took less than 1 second as the query is being executed on the tables stored in SQL database. However, for learning the parameters of the BN, a linear scan of the entire raw data taking approx. 4 minutes. The errors between distribution computed by raw values, and using BN for sample queries have been shown in Table IV.

The distribution errors for queries with high evidence are close to zero. Nevertheless, the first and last row in Table IV show contrasting results, where the first query have less support in the data, but BN based querying is almost accurate as raw data. Further, for the last row, even with high support query, distribution errors are high potentially suggesting other dependencies missing in the encoded BN structure. *Nevertheless*, the direction of change between the distribution prior to and after executing the Bayesian query remains accurate, and in most situations this is what matters the most to an engineer.

B. Querying on disparate data sources: Marketing Data

While conducting market intelligence one often has access to data capturing different attributes of people from several regions typically compiled by different agencies. The situation is similar to the scenario introduced by Table II. Since each

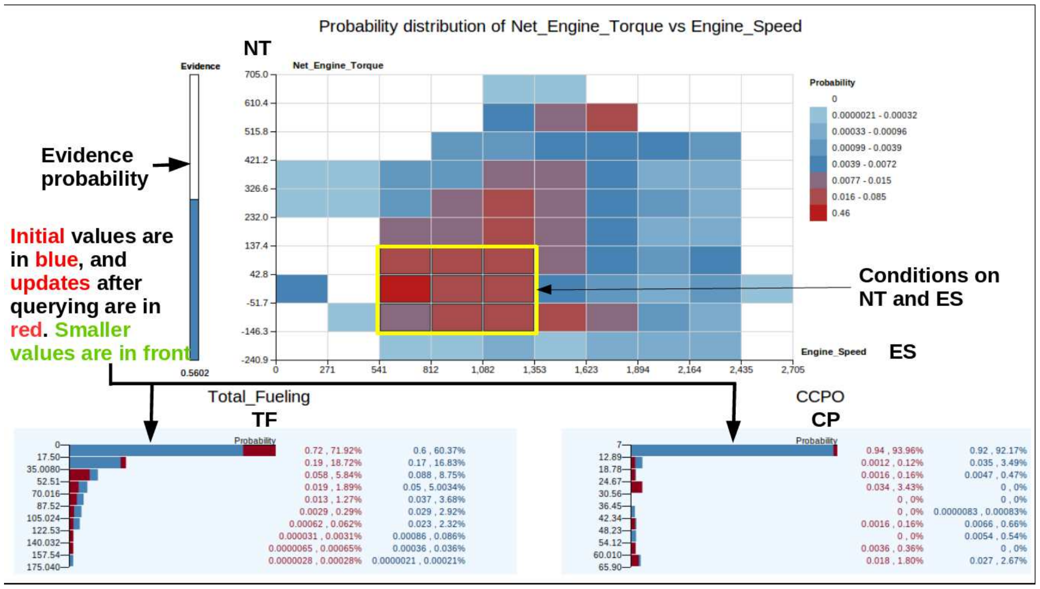


Fig. 4. The distribution of descritized ES Vs NT, TF, and CPas histograms

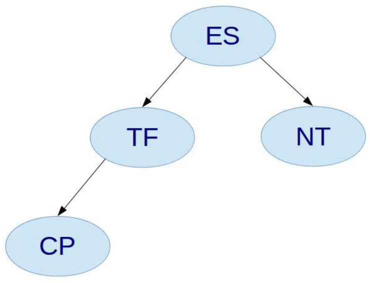


Fig. 5. Bayesian network for vehicle sensor data

TABLE IV
VALIDATION QUERIES FOR MULTI-SENSOR ANALYSIS

Condition	Query	Evidence from data	Evidence from network	KLD	BC
ES=[1,2,4:8] & NT=[2:4,6,7]	TF	0.019	0.021	0.000	0.999
TF=[0,1,5:7]	ES & NT	0.735	0.932	0.692	0.856
TF=[0:8]	CP	0.994	0.998	0.005	0.999
CP=[0:3,6:8]	TF	0.982	0.982	0.276	0.933

agency collects data in a different manner, i.e., each agency collects data from different regions, each potentially delimited differently, combining such data sources becomes an obstacle to deriving any meaningful analysis from such data. One way can be that we consolidate the data from the different subregions by ignoring subregions, i.e., using each data source only as a description of the union of all its regions. However, while this will lead to a larger and reliable dataset, it would be at the expense ignoring insights based on region specific correlations. Instead, in order to determine correlations between the regions we can mark all regions on a common map. We can then find the conditional probability distribution of each of these regions given other regions based on the geographical overlap between each pair of regions.

Data generation: For this experiment we generated a synthetic dataset with 20 million records having four attributes of a person income, profession, location and expenditure, with region being an additional attribute augmenting the attribute set. We assume a rectangular grid on the overall geography of interest. A segmentation of this geography into regions can be defined as shown in the Figure 6. We generated four different segmentations of the overall region (geography) into two or three regions, i.e., the R_is in Table II can each have different cardinality.

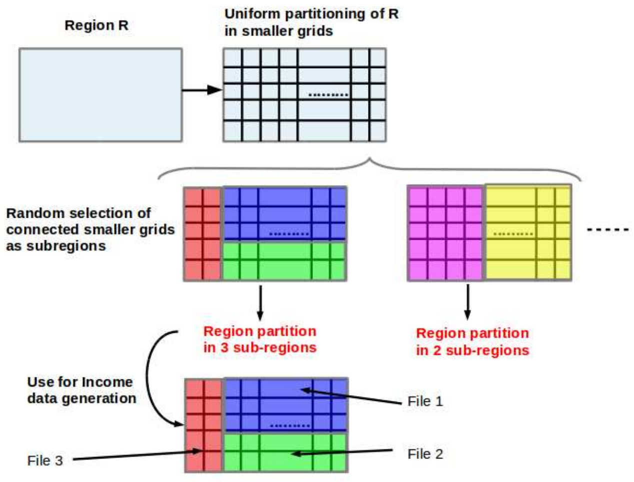

Region R
Uniform partitioning of R in smaller grids
Random selection of connected smaller grids as subregions
Region partition in 3 sub-regions
Region partition in 2 sub-regions
Use for Income data generation
File 1
File 2
File 3

Fig. 6. Synthetic data generation: grids and sub-regions

TABLE V
VALIDATION QUERIES FOR MARKETING DATA

Condition	Query	Evidence from data	Evidence from network	KLD	BC
Income = [1:4] & Profession = [0,3:5]	Expenditure	0.428	0.450	0.090	0.978
Expenditure = [0:2]	Profession & Income	0.727	0.726	0.444	0.901
Profession = [6:8]	Location	0.0465	0.0465	0.281	0.946
Location = [3,4]	Income	0.346	0.346	0.198	0.955

Attribute values for income (I), profession (P), location (L) and expenditure (E) were generated for each grid cell using different means and variances, corresponding to people with different incomes, professions, location they frequent the most, and what they spend on. Separate files were created for each dataset by randomly sampling a large number of records from base generated data, including only one of the four attributes per dataset, and tagging each record with the region corresponding to the segmentation specific for that dataset. The region segmentation for each dataset was represented by a new variable such as R_I for income distribution, which would take one of three values for a sample as shown in the Figure 6. Similarly, for location, variable R_L takes only two possible values.

Probabilistic Inference: Following the data generation process as discussed above, the Bayesian network for this case is modeled as shown in Figure 8. The segmentation of the overall geography into grid cells is assumed to be the 'map' using which we compute the overlap between two region segmentation scheme i.e., $P(\text{R\_I}|\text{R\_L})$, which is used for computing the CPTs in the Bayesian network. Distributional queries are computed using SQL as described earlier. Results of a sample query on this synthetic dataset are shown in the Figure 7, where, as earlier, values before and after executing a query are highlighted.

Errors for sample validation queries are shown in Table V. As seen, the BC distance between conditional distributions computed by the net and raw data have high overlap (though KL divergence gives high values in some cases). Once again though, the direction of the changes in each distribution before and after executing a query are the same whether one uses the original data or our probabilistic method for joining its disparate samples. Note that in practice, since the original joined data samples are assumed to be unavailable, such validations would be impossible to compute these errors; we can do such validation since we are using synthetic data.

As regards execution performance, querying each of the datasets involves loading the entire dataset into the memory while scanning each file. This takes on an average hundred seconds on the synthetic data while the time taken to query using the BN is less than a second. The time taken for querying on the BN using SQL is excluding the time taken to learn the CPTs which also takes approximately hundred seconds, which is again mostly being spent on input-output.

Summary: We have demonstrated the execution distributional queries on disparate datasets. Further, by compressing the data distributions and their correlations into a Bayesian network, we are also able to significantly improve query execution time, in much the same manner as materialized views do for standard database queries.

VII. CONCLUSIONS

We have formally defined the problem of 'distributional' queries one or more datasets, potentially arising from diverse sources, motivated by some real-life as well as potential application scenarios We have argued that under suitable assumptions this problem can be viewed as being equivalent to conditional probabilistic inference on a single dataset, or on a hypothetical 'joined' dataset in the case of multiple datasets from potentially diverse sources. We have also shown that many value-based queries, such as are often encountered in business intelligence, can also be approximately answered using by taking expectations on distributional queries.

We have presented our 'business data fusion' approach to compute distributional queries by approximating the underlying joint distribution via a Bayesian network, defined using domain knowledge as well as constraints arising from what datasets are actually available. We described our system for business data fusion that evaluates distributional queries that performs probabilistic inference using SQL queries on a database of the conditional probability tables of the Bayesian network. In particular, our system does not need to re-access the raw data once the network parameters have been learned. Experimental results were reported on a real-life sensor dataset as well as a synthetic collection of datasets illustrating a hypothetical scenario of marketing analysis, including execution performance as well as accuracy as compared to exact computation using the raw data.

We submit that our approach is well suited for approximately integrating data in a 'data lake' architecture, wherein data stored in near-raw form in a distributed file system is directly processed for insights as opposed to first integrating it into a data warehouse. In fact, as we have also illustrated, seemingly disparate datasets that would have been difficult or impossible to join using traditional database techniques, can nevertheless be processed to derive meaningful insights via distributional queries executed via business data fusion.

Our approach is potentially also usable for data mining in addition to query processing: Since able to answer distributional queries via conditional inference on a Bayesian network, we can determine the support and confidence of any particular combination of values, each of which can be expressed as distributional queries/conditional inference. Thereafter association rules, subgroups are computable using data mining techniques such as [19], [20] to efficiently search the space of combinations of attribute values.

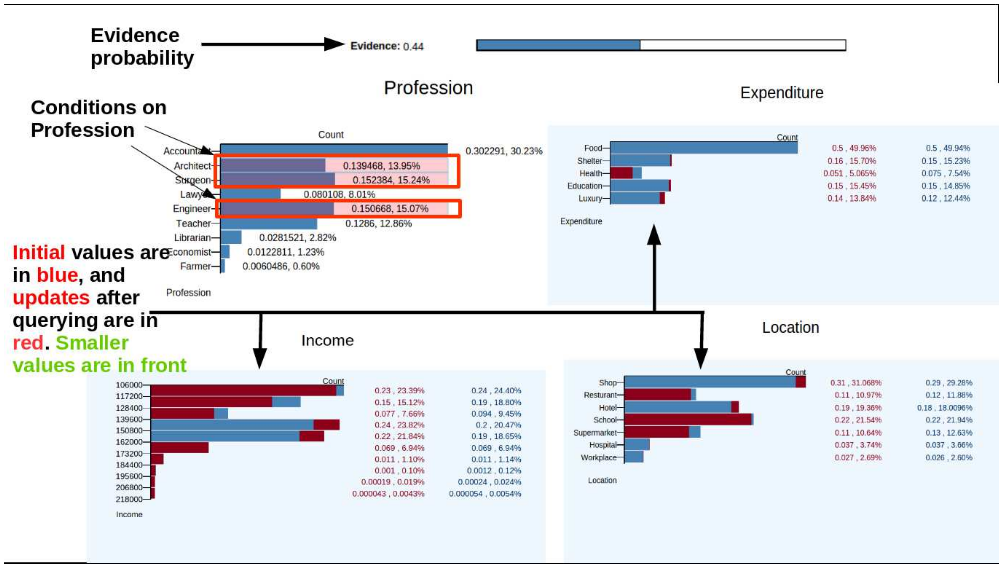


Fig. 7. The distribution of discretized income, profession, location, and expenditure

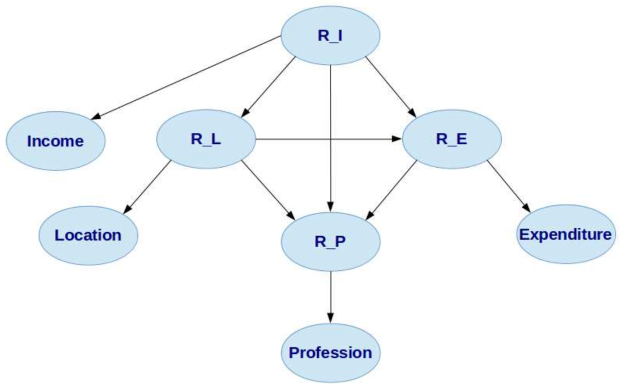


Fig. 8. Bayesian network for connecting marketing datasets

References

[1] M. Stonebraker, "Why the 'Data Lake' is Really a 'Data Swamp'," *CACM*.

[2] D. Hall and J. Llinas, "An introduction to multisensor data fusion," *Proceedings of the IEEE*, vol. 85, no. 1, pp. 6–23, Jan 1997.

[3] E. Waltz, J. Llinas *et al.*, *Multisensor data fusion*. Artech house Boston, 1990, vol. 685.

[4] J. Zhang, "Multi-source remote sensing data fusion: status and trends," *International Journal of Image and Data Fusion*, vol. 1, no. 1, pp. 5–24, 2010.

[5] M. Mutlu, S. C. Popescu, C. Stripling, and T. Spencer, "Mapping surface fuel models using lidar and multispectral data fusion for fire behavior," *Remote Sensing of Environment*, vol. 112, no. 1, pp. 274 – 285, 2008.

[6] D. P. Roy, J. Ju, P. Lewis, C. Schaaf, F. Gao, M. Hansen, and E. Lindquist, "Multi-temporal MODISLandsat data fusion for relative radiometric normalization, gap filling, and prediction of Landsat data," *Remote Sensing of Environment*, vol. 112, no. 6, pp. 3112 – 3130, 2008.

[7] A. N. Steinberg, C. L. Bowman, and F. E. White, "Revisions to the JDL data fusion model," in *AeroSense'99*. International Society for Optics and Photonics, 1999, pp. 430–441.

[8] D. M. Buede and P. Girardi, "A target identification comparison of Bayesian and Dempster-Shafer multisensor fusion," *Systems, Man and Cybernetics, Part A: Systems and Humans, IEEE Transactions on*, vol. 27, no. 5, pp. 569–577, 1997.

[9] P. Pinheiro and P. Lima, "Bayesian sensor fusion for cooperative object localization and world modeling," in *Proc. 8th Conference on Intelligent Autonomous Systems*. Citeseer, 2004.

[10] J. Ye, K. Chen, T. Wu, J. Li, Z. Zhao, R. Patel, M. Bae, R. Janardan, H. Liu, G. Alexander, and E. Reiman, "Heterogeneous Data Fusion for Alzheimer's Disease Study," in *Proceedings of the 14th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, ser. KDD '08. New York, NY, USA: ACM, 2008, pp. 1025–1033.

[11] V. Harinarayan, A. Rajaraman, and J. D. Ullman, "Implementing data cubes efficiently," *ACM SIGMOD Record*, vol. 25, no. 2, pp. 205–216, 1996.

[12] D. Margaritis, C. Faloutsos, and S. Thrun, "NetCube: A Scalable Tool for Fast Data Mining and Compression," in *Proceedings of the 27th International Conference on Very Large Data Bases*, ser. VLDB '01. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 2001, pp. 311–320.

[13] H. B. Mitchell, *Multi-Sensor Data Fusion: An Introduction*, 1st ed. Springer Publishing Company, Incorporated, 2007.

[14] J. Bleiholder and F. Naumann, "Data Fusion," *ACM Comput. Surv.*, vol. 41, no. 1, pp. 1:1–1:41, Jan. 2009.

[15] A. S. Solberg, A. K. Jain, and T. Taxt, "Multisource classification of remotely sensed data: fusion of Landsat TM and SAR images," *Geoscience and Remote Sensing, IEEE Transactions on*, vol. 32, no. 4, pp. 768–778, 1994.

[16] P. Agarwal, G. Shroff, and P. Malhotra, "Approximate incremental big-data harmonization," in *Big Data (BigData Congress), 2013 IEEE International Congress on*. IEEE, 2013, pp. 118–125.

[17] Barbara Vantaggi, "Statistical matching of multiple sources: A look through coherence," *International Journal of Approximate Reasoning*, vol. 49, no. 3, pp. 701 – 711, 2008.

[18] A. Jacobs, "The pathologies of big data," *Communications of the ACM*, vol. 52, no. 8, pp. 36–44, 2009.

[19] R. Agrawal, H. Mannila, R. Srikant, H. Toivonen, A. I. Verkamo *et al.*, "Fast Discovery of Association Rules." *Advances in knowledge discovery and data mining*, vol. 12, no. 1, pp. 307–328, 1996.

[20] S. Saikia, G. Shroff, P. Agarwal, A. Srinivasan, A. Pandey, and G. Anand, "Exploratory data analysis using alternating covers of rules and exceptions," in *Proceedings of the 20th International Conference on Management of Data*. Computer Society of India, 2014, pp. 105–108.