CEUR-WS.org/Vol-2840/short7.pdf

Exploring Data Using Patterns: A Survey and Open Problems

Lukasz Golab
University of Waterloo, Canada
lgolab@uwaterloo.ca

Divesh Srivastava
AT&T Chief Data Office, USA
divesh@att.com

ABSTRACT

We present a survey of data exploration methods that extract multi-dimensional patterns from datasets consisting of dimension and measure attributes. These patterns are designed to summarize common properties of tuples associated with particular values of the measure attributes. We provide a categorization of the characteristics of patterns produced by various solutions to this problem, we point out the pros, cons and performance optimizations of existing methods, and we suggest directions for future research.

KEYWORDS

Data exploration, Data summarization, Data explanation, Data cube, Pattern mining

1 INTRODUCTION

Data volumes have been growing rapidly in recent years. As a result, data-intensive methods are now common in many contexts, including business, science, and public governance. This motivates the need for tools that allow users who are not necessarily data management experts to explore large datasets. Such tools range from visualization and aggregation to flexible search interfaces such as keyword search in structured databases.

In this paper, we focus on the exploration of datasets containing dimension attributes and binary or numeric measure attributes. In traditional business datasets, dimension attributes often describe products or employees, and measure attributes indicate sales totals or salaries. In Internet-of-Things (IoT) and infrastructure monitoring, dimension attributes may describe device properties and measure attributes correspond to performance statistics. In Web datasets, dimension attributes may describe products, with user ratings as measure attributes. Additionally, in any of these applications, derived measure attributes may exist, e.g., a binary attribute denoting whether a given record was determined to be an outlier or to contain an error.

The *data cube* has traditionally been used to explore these kinds of datasets, by allowing users to aggregate, roll-up and drill-down using various subsets of group-by attributes. However, in large-scale databases, the data cube may be very large and may not immediately reveal interesting patterns and trends. This motivates the need for more sophisticated exploration tools [17].

We observe that a large body of recent work on exploring multi-dimensional and OLAP datasets proposes methods to identify interesting fragments of the data, summarized using patterns over the values of the dimension attributes. We make the following contributions towards an understanding of these data exploration methods.

(1) We survey recent work on data exploration using multi-dimensional patterns and propose a categorization based on the properties of patterns suggested for exploration: *coverage*, *contrast*, and *information*.
(2) We analyze the pros, cons and performance optimizations of existing work, and we suggest open problems for future research.

© Copyright 2021 for this paper held by its author(s). Use permitted under Creative Commons License Attribution 4.0 International (CC BY 4.0).

2 BACKGROUND

We are given a dataset S with a set D of dimension attributes and a set M of measure attributes (also referred to as outcomes in some prior work [6]). Let $D_1, D_2, \ldots, D_d$ be the d dimension attributes and let $M_1, M_2, \ldots M_m$ be the m measure attributes. For now, we assume, as in the majority of previous work, that the dimension attributes are categorical, and we will comment on ordinal and numeric dimension attributes in Section 4. Measure attributes may be binary or numeric.

Table 1 shows a flight dataset that will serve as a running example. For each flight, the dataset includes a record id, followed by three dimension attributes, Day of the week, flight Origin and flight Destination, as well as two measure attributes, a numeric attribute denoting how late the flight was and a binary attribute denoting whether the flight was full.

Let $dom(D_i)$ be the active domain of the ith dimension attribute. A pattern p (referred to as a rule in some prior work [8]) is a tuple from $dom(D_1) \cup \{*\} \times \cdots \times dom(D_d) \cup \{*\}$, i.e., from the data cube over the dimension attributes, with '*' denoting all possible values of that attribute. A tuple $t \in S$ matches p, denoted by $t \asymp p$, if $p[D_j] =$ '*' or $t[D_j] = p[D_j]$ for each dimension attribute D_j. For example, tuple 4 from Table 1 matches the patterns $(*, *, *)$ and $(*, *, London)$, but does not match the pattern $(Fri, *, *)$. Some approaches [10] support richer patterns, with disjunctions and dimension hierarchies.

Let $sup(p)$ be the support of p in S, i.e., the number of tuples matching p, and let $sup_r(p)$ be the number of tuples matching p and satisfying the predicate r over the measure attributes. For example, $sup(*, *, London) = 4$ and $sup_{Full=0}(*, *, London) = 1$. Furthermore, let $\theta_r(p) = \frac{sup_r(p)}{sup(p)}$, which is the fraction of tuples matching p that also satisfy the predicate r. For example, $\theta_{Full=1}(*, *, London) = \frac{3}{4}$.

Let $sum^{M_i}(p)$ be the sum of the values of a measure attribute M_i over all the tuples matching p. Let $sum_r^{M_i}(p)$ be the sum of the values of a measure attribute M_i over all the tuples matching p and satisfying the predicate r over the measure attributes. For example, $sum^{Late}(*, *, London) = 20 + 15 + 19 + 7 = 61$ and $sum_{Full=0}^{Late}(*, *, London) = 7$.

We survey solutions to the following data exploration problem: given a dataset S, produce a set or a list of patterns P over the dimension attributes of S, as defined above, that summarize common properties of tuples sharing the same values of the measure attribute(s). Users may then inspect the patterns and explore tuples covered by the patterns. They may then extract patterns corresponding to a smaller subset of the data that was found to be interesting during the earlier exploration step, and so on.

The number of patterns in P should be limited to direct the user's attention to the most important or interesting regions of the data. This limit may be set explicitly by the user (in terms of the maximum number of patterns in P) or implicitly by retrieving

Table 1: A flight dataset

id	Day	Origin	Dest.	Late	Full
1	Fri	SF	London	20	1
2	Fri	London	LA	16	1
3	Sun	Tokyo	Frankfurt	10	1
4	Sun	Chicago	London	15	1
5	Sat	Beijing	Frankfurt	13	1
6	Sat	Frankfurt	London	19	1
7	Tue	Chicago	LA	5	0
8	Wed	London	Chicago	6	0
9	Thu	SF	Frankfurt	15	1
10	Mon	Beijing	SF	4	0
11	Mon	SF	London	7	0
12	Mon	SF	Frankfurt	5	0
13	Mon	Tokyo	Beijing	6	0
14	Mon	Frankfurt	Tokyo	4	0

the fewest possible patterns that jointly satisfy some property such as covering some fraction of the data.

This data exploration problem has the following applications.

- *Explaining the results of aggregate queries.* Suppose a data analyst issues the following query over Table 1: SELECT SUM(Late) FROM S. Suppose the analyst wishes to understand why the result, of 145, is so high. Here, interesting patterns are those which cover tuples that make a significant contribution to the result, i.e., those with a high $sum^{Late}()$ such as $(*, *, London)$. The analyst may then zoom into flights landing in London and investigate potential reasons for the lengthy delays.
- *Analyzing outliers and data quality issues.* Suppose we have a binary measure attribute denoting whether a given tuple contains an error or is an outlier. This attribute could be created manually by domain experts or automatically by identifying tuples that violate data quality rules or deviate from the expected distribution. We may wish to produce patterns that summarize the properties of erroneous tuples to help determine the root cause of data quality problems.
- *Feature selection and explainable AI.* Before building prediction models, a data scientist may explore interesting patterns to understand which dimension attributes are related to the measure attribute that is to be predicted. Furthermore, suppose a data analyst wants to understand how a black-box model makes classification decisions. Here, the dimension attributes are the features given to the model as input, and, as the measure attribute, the analyst records the predictions made by the model. The analyst may then want to find interesting patterns that explain the prediction decisions. For example, in Table 1, the pattern $(Mon, *, *)$ is associated with tuples having $Full = 0$, suggesting that flights scheduled on Mondays are usually not full[1].

3 SURVEY

In this section, we provide a categorization of previous work on data exploration using multi-dimensional patterns based on the pattern properties and ranking strategies used for pattern selection. We categorize these properties into three types: those focusing on coverage, contrast and information. Table 2 categorizes the surveyed methods and lists their motivating applications, as mentioned in the corresponding papers.

[1]Model explanations may be global (to summarize how classification decisions are made) or local (to explain why a specific record was classified in a particular way). The explanations discussed in this paper are examples of global explanations.

Table 2: Methods surveyed

Method	Approach	Applications
CAPE [14]	Contrast	Explaining queries
Data Auditor [9]	Coverage	Data quality analysis
Data X-ray [19]	Contrast	Data quality analysis
DIFF [2]	Contrast	Outlier analysis
Explanation tables [7]	Information	Feature selection
Macrobase [1]	Contrast	Outllier analysis
MRI [5]	Coverage	Explaining queries
RSExplain [15]	Contrast	Explaining queries
Scorpion [20]	Contrast	Outlier analysis
Shrink [10]	Information	Explaining queries
Smart Drilldown [11]	Coverage	Explaining queries
SURPRISE [16]	Information	Explaining queries

3.1 Methods Based on Coverage

The goal of these methods is to identify patterns that cover tuples with certain values of the measure attribute. We discuss three coverage-based methods: Data Auditor [9], MRI [5], and Smart Drilldown [11].

Suppose we are interested in covering tuples having $Full = 1$ in Table 1, as a way to summarize the characteristics of full flights. A simple coverage-oriented approach is to sort the patterns according to $sup_{Full=1}()$ and output the top-ranking patterns. Ignoring $(*, *, *)$, which always covers everything but is not informative, the top candidates are $(*, *, London)$ and $(*, *, Frankfurt)$, which cover three full flights each, followed by the following patterns that cover two such tuples each: $(Fri, *, *)$, $(Sun, *, *)$, $(Sat, *, *)$ and $(*, SF, *)$.

There are two problems with this simple approach: there may be many patterns with a nonzero $sup_{Full=1}(p)$, and some of these patterns may also cover tuples with other values of the measure attribute (here, $Full = 0$). To reduce the size of the output and to ensure that each pattern co-occurs with the specified value of the measure attribute, Data Auditor solves the following set cover problem. Continuing with our example, Data Auditor requires a minimum threshold for $\theta_{Full=1}(p)$, i.e., the fraction of tuples covered by p that correspond to full flights. Suppose we require $\theta_r(p) \geq 0.75$. This threshold defines the candidate sets for the set cover problem. The set cover objective is to select the fewest such patterns that together cover a specified fraction of tuples in S having $Full = 1$. Suppose this coverage fraction, which would again be set by the user, is 0.5. The goal is then to find the fewest patterns, as defined above, to cover half the full flights.

The set cover problem is NP-hard, and Data Auditor uses the standard greedy heuristic that achieves a logarithmic approximation ratio in the size of the solution: it iteratively chooses the pattern that covers the most uncovered tuples (having the desired value of the measure attribute), until the desired fraction of such tuples has been covered. In our example, the first pattern added to P is either $(*, *, London)$ and $(*, *, Frankfurt)$ - they each cover three full flights and their $\theta_{Full=1}(p)$ values are 0.75 each. Suppose the set cover algorithm selects $(*, *, Frankfurt)$. Since there are seven full flights in the dataset and our coverage threshold is 0.5, we need to cover one more full flight. In the next iteration, the pattern that (has $\theta_{Full=1}(p) \geq 0.75$ and) covers the most remaining full flights is $(*, *, London)$ and the algorithm terminates, with P consisting of these two patterns.

The next method, MRI, finds k patterns that cover a user-specified fraction of the data and satisfy additional properties related to the variance of the measure attribute within each pattern. The motivating example for MRI was to explain queries over product reviews, with the dimension attributes corresponding to information about the reviewers (such as their gender and age) and the numeric measure attribute corresponding to the average rating. Minimizing variance amounts to returning patterns (having high coverage and) describing reviewers with similar opinions. For example, when applying MRI to Table 1 with $Late$ as the measure attribute, the pattern $(*, *, Frankfurt)$ is preferred over $(*, *, London)$. Both patterns cover four tuples, but the former has a lower variance of the $Late$ attribute within the covered tuples.

Smart Drilldown also combines coverage with additional pattern properties. Smart Drilldown produces k patterns. In every iteration of the algorithm, the chosen pattern maximizes the following objective: the number of tuples not yet covered multiplied by the *weight* of the pattern corresponding to some measure of interestingness. One simple weighting function proposed in [11] is the number of non-star values in the pattern (i.e., more specific patterns are considered to be more desirable). In Table 1 for example, this weighting function prefers $(Tue, Chicago, LA)$ over $(Tue, *, *)$ – both of these patterns have a support of one, but the former has more non-star values.

3.1.1 Performance Optimizations. A performance bottleneck in general set cover problems results from having to keep track of the number of uncovered elements that can be covered by the candidate sets - this changes in every iteration, whenever a new set is added to the solution. Data Auditor relies on the hierarchical nature of patterns and does not generate all the patterns that serve as input to greedy set cover a priori. Instead, patterns are generated on-demand, only after all of their supersets have already been considered. For example, a pattern such as $(Fri, *, London)$ would only be generated after all of its supersets - including $(Fri, *, *)$ and $(*, *, London)$ - have already been considered. Until then, $(Fri, *, London)$ can be safely ignored: it cannot cover more elements than its supersets and therefore is guaranteed to not be selected by the greedy set cover heuristic at this time. This optimization can greatly reduce the number of generated patterns whose coverage (of uncovered elements) must be re-computed while constructing the solution.

3.1.2 Pros and Cons. One advantage of coverage-based methods is conciseness: by design, they identify the fewest possible patterns that cover the desired fraction of tuples of interest. On the other hand, in Data Auditor, some trial-and-error may be required on the user's part to select good values for the two required thresholds. For example, a high value of θ will ensure that each pattern co-occurs mainly with the specified value of the measure attribute, but will disqualify more patterns from consideration. Similarly, MRI and Smart Drilldown require users to set parameters for coverage and other pattern properties.

3.2 Methods based on Contrast

This is the largest group of methods, consisting of CAPE [14], Data X-ray [19], DIFF [2], Macrobase [1], RSExplain [15] and Scorpion [20]. (DIFF is a recent solution that generalizes many of these related methods). Contrast-based methods usually assume a binary measure attribute, and select patterns co-occurring with one value of the measure attribute but not the other. These patterns reflect the contrast between tuples having different values of the measure attribute.

One could argue that interpretable classifiers such as decision trees and rule-based methods (see, e.g., [12]) can also be used for contrast-based data exploration. These methods identify patterns of values of the feature attributes that have high discriminative power in terms of the class variable (in our case, the binary measure attribute). These patterns are therefore likely to provide contrast as well. However, classification algorithms focus on out-of-sample predictive power and include optimizations such as rule pruning to avoid overfitting. On the other hand, the methods covered in this survey focus explicitly on identifying a concise set of interesting fragments of the data for user exploration.

Contrast-based methods can also explain the results of aggregate queries. Consider the following query over Table 1: SELECT SUM(Late) FROM S WHERE Full=1. Here, the measure attribute corresponding to the quantity being aggregated. We then set the binary measure attribute to one for all tuples that participate in the query (i.e., tuples that match the WHERE predicate), and we select patterns of tuples that contribute to the result of the query (but would not contribute had the query been issued against the other tuples in the dataset).

Below, we explain the pattern ranking strategies used by contrast-based methods, with Table 1 as a running example.

Risk ratio is used by Macrobase; a related metric called Diagnosis Cost is used by Data X-ray. It is the ratio of the following two probabilities: 1) the probability that a tuple with a particular value is covered by the given pattern, and 2) the probability that a tuple with this particular value occurs outside this pattern. In our example, $risk_{Full=1}(p) = \frac{\theta_{Full=1}(p)}{\frac{sup_{Full=1}(*,*,*)-sup_{Full=1}(p)}{(sup_{Full=1}(*,*,*)-(sup_{Full=1}(p))+(sup_{Full=0}(*,*,*)-sup_{Full=0}(p))}}$. For instance, $risk_{Full=1}(*, *, London) = \frac{0.75}{0.4} = 1.875$, whereas $risk_{Full=1}(*, SF, *) = \frac{0.5}{0.5} = 1$. This indicates that $(*, *, London)$ represents full flights better than $(*, SF, *)$.

Mean shift (supported by DIFF) computes the ratio of the mean of the measure attribute values co-occurring with the two values of the binary measure attribute. In our example, $mean^{Late}_{Full=1}(p) = \frac{sum^{Late}_{Full=1}(p)/sup_{Full=1}(p)}{sum^{Late}_{Full=0}(p)/sup_{Full=0}(p)}$. For instance, $mean^{Late}_{Full=1}(*, *, London) = \frac{54/3}{7/1} = 2.57$, indicating that full fights to London have delays that are 2.57 times longer than non-full flights to London.

Intervention is used by RSExplain; a related metric called Influence is used by Scorpion. It measures the ratio of contribution towards the numeric measure attribute for tuples occurring with the different values of the binary measure attribute. In our example, $intervention^{Late}_{Full=1}(p) = \frac{sum^{Late}_{Full=1}(*,*,*)-sum^{Late}_{Full=1}(p)}{sum^{Late}_{Full=0}(*,*,*)-sum^{Late}_{Full=0}(p)}$. For instance, $intervention^{Late}_{Full=1}(*, *, London) = \frac{108-54}{37-7} = 1.8$. In other words, if flights to London were removed from the dataset then full flights would have delays on average 1.8 times longer than non-full flights. On the other hand, $intervention_{Full=1}(*, SF, *) = \frac{108-35}{37-12} = 2.92$, meaning that removing flights departing from SF from the dataset would create a greater *contrast* between the delays of full and non-full flights.

We also point out CAPE, whose goal is to find patterns whose measure attribute values *counterbalance* those of the pattern given as input. For example, the average flight delay in Table 1

Table 3: An explanation table of size four for the binary measure attribute Full

Day	Origin	Dest.	Full
*	*	*	0.5
Mon	*	*	0
*	*	London	0.75
*	*	Frankfurt	0.75

Table 4: An explanation table of size four for the numeric measure attribute Late

Day	Origin	Dest.	Late
*	*	*	10.4
*	*	London	15.3
Fri	*	*	18
Sat	*	*	16

is 10.4, but it is only 5.2 for tuples covered by $(Mon, *, *)$. To counterbalance this low value, CAPE suggests related patterns such as $(Fri, *, *)$ and $(Sat, *, *)$, whose average delays are higher.

3.2.1 Performance Optimizations. Contrast-based methods output patterns whose scores (e.g., risk ratios) are above a user-supplied threshold. A general optimization used by DIFF is to also require a minimum support threshold for the extracted patterns. This enables pruning optimizations similar to those used by the Apriori algorithm for frequent itemset mining [3]. For example, if $(Wed, *, *)$ fails to satisfy the minimum support threshold, then all its subsets, such as $(Wed, London, *)$, can be ignored.

3.2.2 Pros and Cons. By design, contrast-based methods are useful when exploring differences between data subsets – something that coverage-based methods do not implicitly optimize for. On the other hand, these methods may not guarantee concise results. One exception is Data X-ray, which performs a set-cover-like operation on the extracted patterns as a post-processing step to eliminate redundant patterns.

3.3 Methods based on Information

Finally, we discuss three methods that select patterns based on the information they provide about the distribution of the measure attribute: Explanation Tables [7], SURPRISE [16] and Shrink [10].

Table 3 shows an explanation table of size four (i.e., containing four patterns) for the binary measure attribute Full based on Table 1. In addition to values of the dimension attributes, each explanation table pattern also includes the fraction of matching tuples that have $Full = 1$. The first pattern in an explanation table is always the all-stars pattern, and, in this example, it also indicates that half the flights in the entire dataset are full. The next pattern suggests that no flights on Mondays are full, and the last two patterns indicate that three-quarters of flights to London and Frankfurt are full.

In Table 4, we show an explanation table of size four for the measure attribute Late based on Table 1. Here, each pattern includes the average value of Late across its matching tuples. Again, we start with the all-stars pattern, which states that flights are 10.4 minutes late on average. The next pattern indicates that flights to London are 15.3 minutes late on average, and so on.

The greedy heuristic for constructing explanation tables used in [6–8] iteratively selects the pattern that contains the most information about the distribution of the measure attribute. To do so, the algorithm maintains a *maximum-entropy* estimate of the distribution based on the patterns that have been added to the explanation table so far. It also keeps track of the *Kullback-Leibler* (KL) divergence between this estimated distribution and the true distribution. To quantify the information contained in a candidate pattern p, the algorithm computes the reduction in KL-divergence if p were to be added to the explanation table.

Table 5: A summary of size two considered by Shrink [10]

Day	Origin	Dest.	Late
Fri,Sun,Sat,Thu	*	*	15.4
Tue,Wed,Mon	*	*	5.3

Returning to Table 3, the greedy algorithm starts by inserting the pattern $(*, *, *||0.5)$. At this point, knowing only this one piece of information, the maximum-entropy estimate of the distribution of $Full$ is to assign $Full = 0.5$ to every tuple in Table 1[2]. Next, it turns out that $(Mon, *, *||0)$ gives the greatest reduction in KL-divergence. Based on this new pattern, the maximum-entropy estimate for tuples 10 through 14 in Table 1 changes to $Full = 0$. This revision causes the estimates of the first nine tuples to change (from 0.5 to $\frac{7}{9}$) in order to maintain consistency with the first pattern, which asserts that $Full = 0.5$ on average over the entire table. Given the updated maximum-entropy estimate, the next pattern with the greatest reduction in KL-divergence is $(*, *, London||0.75)$, and so on.

Similar reasoning can explain how Table 4 was created. The first pattern asserts that flights are late by 10.4 minutes on average. Given this estimate, $(*, *, London||15.3)$ provides the most information about the distribution of $Late$. The maximum-entropy estimate of $Late$ is now updated accordingly, That is, tuples 1, 4, 6 and 11, corresponding to flights to London, receive an estimate of 15.3, and the remaining tuples receive an estimate of 8.4 to maintain consistency with the first pattern. The next most-informative pattern is then selected, and so on.

SURPRISE is a similar method, whose goal is to identify surprising fragments of a dataset where the distribution of the measure attribute is different than what the user has seen so far. Suppose the user queries Table 1 and finds out that flights are 10.4 minutes late on average. SURPRISE finds the most informative *non-overlapping* and *contained* patterns, i.e., those which lead to the greatest reduction in KL-divergence between the true distribution of Late and the maximum-entropy estimated distribution. Restricting the output to such patterns makes it easier to update the estimated distribution. In our example, the most informative pattern is $(*, *, London)$ and its most informative subset is $(*, SF, London)$.

Finally, we discuss Shrink, which produces k non-overlapping patterns that summarize the distribution of the measure attribute with (approximately) minimal sum of squared errors. Shrink allows patterns with disjunctions of values and dimension hierarchies, and uses a greedy approach in which cells in the data cube are merged to produce patterns with the smallest squared errors. Table 5 shows an example of a summary with two patterns that may be considered by Shrink based on Table 1. Given the average values of the Late measure attribute reported by the summary, the sum of squared errors is 77.1.

[2] $Full$ is a binary attribute that can only be zero or one. However, for the purpose of measuring the divergence between the true and the estimated distributions, the maximum-entropy estimates are allowed to be real numbers between zero and one.

3.3.1 Performance Optimizations. In contrast-based methods, the "goodness" of a pattern usually remains static throughout the execution of the algorithm. On the other hand, the "goodness", i.e., information, of a candidate explanation table pattern is *relative* to the current maximum-entropy estimate of the measure attribute. This means that a previously un-informative pattern may become informative as more patterns are added to the explanation table. As a result, candidate patterns cannot be easily pruned. Instead, the explanation table construction algorithm in [6–8] uses *sampling* to reduce the space of candidate patterns. In every iteration of the greedy algorithm, a random sample is drawn, and the set of candidate patterns corresponds to only those patterns that have a non-zero support in the sample. The intuition is that patterns with frequently occurring combinations of dimension attribute values are likely to be sampled and also likely to contain information about the distribution of the measure attribute.

3.3.2 Pros and Cons. By design, information-based methods produce *informative* patterns that highlight fragments of the data with surprising distributions of the measure attribute. However, these methods tend to be expensive, especially as the number of dimension attributes grows.

4 CONCLUSIONS AND OPEN PROBLEMS

In this paper, we surveyed recent data exploration methods that extract interesting or informative fragments of the data, represented as patterns over the dimension attributes. We categorized these methods according to the properties of patterns they select. Below, we offer suggestions for future work in this area.

Benchmarks: A performance comparison of contrast-based methods implemented within the DIFF frameworks appears in [2]. In terms of effectiveness, prior work reports that methods based on information provide more information about the distribution of the measure attribute than coverage-based methods [6, 7]; similarly, methods based on contrast provide more precise outlier explanations than methods based on coverage [19]. Some approaches were also evaluated through user studies against simple baselines [14]. An interesting direction for future work is to develop a comprehensive benchmark to highlight the effectiveness of various types of methods in various applications.

New applications: Popular motivating applications that guided the development of prior work were outlier and data error analysis, as well as query result explanation. Recent interest in explainable AI motivates further studies on exploring the behaviour of black-box machine learning models such as neural networks using multi-dimensional patterns, as was suggested in [7]. Since deep learning methods have been successful in the context of unstructured data such as text, images and graphs, future research should investigate new ways of formulating interpretable patterns over these high-dimensional unstructured datasets.

Correlated measure attributes: Another characteristic of prior work is that it usually formulates exploration problems involving a single measure attribute. Pattern-based exploration of multiple measure attributes is an interesting area for future work.

Feature reduction: In terms of performance and scalability, the large number of possible patterns remains a challenge for many methods, especially those based on information which cannot leverage Apriori-like pruning strategies. This is an important challenge for interactive methods that allow users to continuously issue new exploration tasks. As a result, some techniques such as DIFF limit the number of dimension attributes for use in patterns and discard redundant dimension attributes such as those functionally determined by other attributes. Distributed versions of some methods, including DIFF [2] and Explanation Tables [8], have also been proposed to parallelize the search for interesting patterns. In machine learning, there exists a variety of dimension reduction methods such as Principal Component Analysis (PCA) and word embeddings. However, these methods are not known for being interpretable and thus their suitability for pattern-based exploration requires further study.

Bringing order to dimension attributes: Much of the previous work considers categorical dimension attributes. However, there exist methods for covering a multi-dimensional dataset using *hyper-rectangles* corresponding to intervals over numeric dimension attributes [13], there exists a method to cover data anomalies using intervals over numeric features [21], and explanation tables have recently been extended to support ordinal and numeric dimension attributes [18]. These extensions further increase the space of candidate patterns and require additional performance optimizations. For example, returning to Table 1, the Day attribute may lead to additional patterns with ranges or intervals such as $([Mon - Fri], *, *)$ or $([Sat - Sun], *, *)$. Techniques used for constructing optimal histograms and optimal decision trees may help to discover these types of patterns.

Exploring data evolution: Finally, recent work motivates the need for tools to explore how data (and metadata) change over time [4]. Here, patterns may summarize fragments of the data that have changed recently or are updated often.

REFERENCES

[1] F. Abuzaid, P. Bailis, J. Ding, E. Gan, S. Madden, D. Narayanan, K. Rong, S. Suri: MacroBase: Prioritizing Attention in Fast Data. ACM Trans. Database Syst. 43(4): 15:1-15:45 (2018)

[2] F. Abuzaid, P. Kraft, S. Suri, E. Gan, E. Xu, A. Shenoy, A. Anathanaraya, J. Sheu, E. Meijer, X. Wu, J. F. Naughton, P. Bailis, M. Zaharia: DIFF: A Relational Interface for Large-Scale Data Explanation. PVLDB 12(4): 419-432 (2018)

[3] R. Agrawal, R. Srikant: Fast Algorithms for Mining Association Rules in Large Databases. VLDB 1994: 487-499

[4] T. Bleifuss, L. Bornemann, T. Johnson, D. V. Kalashnikov, F. Naumann, D. Srivastava: Exploring Change - A New Dimension of Data Analytics. PVLDB 12(2): 85-98 (2018)

[5] M. Das, S. Amer-Yahia, G. Das, C. Yu: MRI: Meaningful Interpretations of Collaborative Ratings. PVLDB 4(11): 1063-1074 (2011)

[6] K. El Gebaly, P. Agrawal, L. Golab, F. Korn, D. Srivastava: Interpretable and Informative Explanations of Outcomes. PVLDB 8(1): 61-72 (2014)

[7] K. El Gebaly, G. Feng, L. Golab, F. Korn, D. Srivastava: Explanation Tables. IEEE Data Eng. Bull. 41(3): 43-51 (2018)

[8] G. Feng, L. Golab, D. Srivastava: Scalable Informative Rule Mining. ICDE 2017: 437-448

[9] L. Golab, H. J. Karloff, F. Korn, D. Srivastava: Data Auditor: Exploring Data Quality and Semantics using Pattern Tableaux. PVLDB 3(2): 1641-1644 (2010)

[10] M. Golfarelli, S. Graziani, S. Rizzi: Shrink: An OLAP operation for balancing precision and size of pivot tables. Data Knowl. Eng. 93: 19-41 (2014)

[11] M. Joglekar, H. Garcia-Molina, A. G. Parameswaran: Interactive data exploration with smart drill-down. ICDE 2016: 906-917

[12] H. Lakkaraju, S. H. Bach, J. Leskovec: Interpretable Decision Sets: A Joint Framework for Description and Prediction. KDD 2016: 1675-1684

[13] L. V. S. Lakshmanan, R. T. Ng, C. X. Wang, X. Zhou, T. Johnson: The Generalized MDL Approach for Summarization. VLDB 2002: 766-777

[14] Z. Miao, Q. Zeng, C. Li, B. Glavic, O. Kennedy, S. Roy: CAPE: Explaining Outliers by Counterbalancing. PVLDB 12(12): 1806-1809 (2019)

[15] S. Roy, D. Suciu: A formal approach to finding explanations for database queries. SIGMOD Conference 2014: 1579-1590

[16] S. Sarawagi: User-cognizant multidimensional analysis. VLDB J. 10(2-3): 224-239 (2001)

[17] P. Vassiliadis, P. Marcel: The Road to Highlights is Paved with Good Intentions: Envisioning a Paradigm Shift in OLAP Modeling. DOLAP 2018.

[18] M. Vollmer, L. Golab, K. Bohm, D. Srivastava: Informative Summarization of Numeric Data. SSDBM 2019: 97-108

[19] X. Wang, X. L. Dong, A. Meliou: Data X-Ray: A Diagnostic Tool for Data Errors. SIGMOD Conference 2015: 1231-1245

[20] E. Wu, S. Madden: Scorpion: Explaining Away Outliers in Aggregate Queries. PVLDB 6(8): 553-564 (2013)

[21] H. Zhang, Y. Diao, A. Meliou: EXstream: Explaining Anomalies in Event Stream Monitoring. EDBT 2017: 156-167