TREND SURFING: EFFECTIVE AND EFFICIENT RETRIEVAL OF UNUSUAL TEMPORAL TRENDS

Jing Ao, Kara Schatz and Rada Chirkova

Department of Computer Science, North Carolina State University,
Raleigh, North Carolina, USA

ABSTRACT

Locating unusual temporal trends in data cubes is a recurrent task in a variety of application domains. We consider a version of this problem in which one looks for the data dimensions that are best correlated with the given unusual temporal trends. Our goal is to make such data-cube navigation in search of unusual temporal trends both effective and efficient. Challenges in achieving this goal arise from the rarity of the trends to be located, as well as from the combinatorics involved in locating in data cubes nodes with unusual trends. We show that exhaustive solutions are worst-case intractable, and introduce tractable heuristic algorithms that enable effective and efficient data-cube navigation in a particular manner that we call trend surfing. We report the results of testing the proposed algorithms on three real-life data sets; these results showcase the effectiveness and efficiency of the algorithms against the exhaustive baseline.

KEYWORDS

Unusual temporal trends and associated dimensions, effective and efficient navigation in the data cube, criteria for trend unusualness.

1. INTRODUCTION

Date	Source	Serotype	Ampicillin (AMP) MIC	Ampicillin (AMP) R/S/I
2/8/2009	Human	Salmonella A	<1	S
2/8/2009	Human	Salmonella A	>32	R
2/8/2009	Human	Salmonella A	16	Int
2/8/2009	Human	Salmonella A	>32	R
2/8/2009	Human	Salmonella B	<1	S
2/8/2009	Human	Salmonella B	>32	R
2/8/2009	Human	Salmonella C	>32	R

Fig. 1: This spreadsheet shows, for each biological sample, its collection date, source, serotype (bacterium) type, minimum inhibitory concentration (MIC) values for drug Ampicillin, and the interpretations (R/S/Int) of the MIC values, with R standing for (serotype) "resistant" (to the drug), S for "susceptible," and Int for "intermediate."

Locating unusual temporal trends in the given data is an important component of the data-analysis framework in a variety of application domains [1–3]. For instance, being able to find unusual delay patterns for airline flights over time opens the door to the analysis and eventual rectification of the underlying problems. As another example, in studying the efficacy of antimicrobial drugs on bacteria, researchers could be interested in finding

David C. Wyld et al. (Eds): MLNLP, NWCOM, DTMN, ASOFT, SIGPRO, AIFZ, CSITY, CLSB - 2023
pp. 103-123, 2023. CS & IT - CSCP 2023 DOI: 10.5121/csit.2023.131608

uncommon temporal patterns of bacterial resistance to drugs. Consider Fig. 1, which shows data that could be collected about the degree of antibiotic resistance of bacteria in culture samples taken from humans. Spreadsheets with data of this form are manually put together by researchers all over the world, with summarized reports on the data published annually in a number of countries, see, e.g., [4, 5].

It is unfortunately typical for bacteria to become resistant to antimicrobials over time. However, in some cases a bacterium can start out being susceptible to a particular drug,

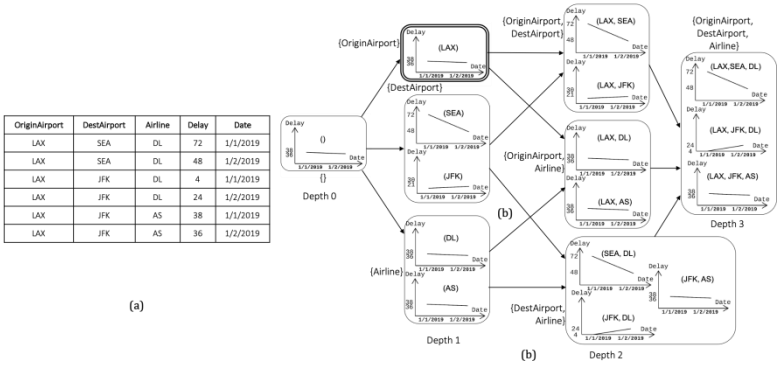

Fig. 2: Fact table F with nontemporal dimensions OriginAirport, DestAirport, and Airline, temporal dimension Date, and dependent attribute Delay (a); search space of temporal trends for F , with a node for each subset of the set {OriginAirport, DestAirport, Airline} (b). The node at depth 0 in the search space shows the data from (a) aggregated into a single temporal trend, and the node at depth 3 shows the trends in (a) itself. The {OriginAirport}-node at depth 1 (highlighted) shows a typical temporal trend for that depth, while the {DestAirport}-node immediately below it shows another typical trend labeled with the value JFK of DestAirport, as well as a trend labeled with the value SEA of DestAirport, which is an unusual trend for depth 1.then become resistant, and then become susceptible to the same drug again for a variety of reasons. (We will refer to such a trend as the S->R->S trend.) Studying this and other unusual temporal trends can lead to the more effective use of existing or emerging drugs that would prevent antimicrobial resistance in a wider range of bacteria.

By their nature, unusual temporal trends are rare and thus not easy to find. Further, in analyzing temporal trends it is often important to understand which of the dimen- sions of the data are correlated with the given rare trend. That is, how the data have been grouped to obtain a particular trend and which values of the grouping attributes are responsible for its unusualness can be critical in understanding and addressing the under- lying problems. E.g., in the airline-flights domain, is it specific airports, specific airlines, or a combination that is associated with the given flight-delay pattern? (See Fig. 2.) In antimicrobial-resistance research, is it the specific bacteria, sample sources, antimicrobial drugs, or a combination that can help us understand the given S->R->S trend?

To enable domain experts to find answers to such questions, we specify the scenario of interest as follows. We assume that experts work with a fact table that has dimensions, including a temporal dimension, and a dependent attribute of interest. **The problem of interest** in this scenario is to retrieve unusual temporal trends from the data cube [6, 7] induced by the fact table, together with the grouping attributes and their values that can help in understanding the trends and making appropriate decisions in the data-analysis lifecycle. To find temporal trends of interest, experts would navigate the data cube in some fashion,

grouping and aggregating the data along the way as appropriate. **Our research goal** in this paper is to make such navigation more effective and efficient.

Challenges in addressing the problem of retrieval of unusual temporal trends from data cubes arise from the rarity of the trends to be located, as well as from the combinatorics involved in finding the relevant data-cube nodes. Another key challenge lies in determining what makes a temporal trend unusual for a given task in a given domain. In this paper, we address these challenges by presenting a suite of approaches that enable effective and efficient **trend surfing** of the data cube, which exposes the temporal trends in the cube in a particular way. The approaches accommodate in a flexible way tools and metrics that can help users decide which trends are unusual for their task and domain purposes.

The problem of effectively and efficiently retrieving unusual temporal trends, together with the grouping attributes and their values that can help in understanding the trends, is novel to the best of our knowledge. The problem that resembles ours the closest is considered in [8], which presents a visual tool, Metro-Viz, for assisting data scientists in effectively and efficiently analyzing anomalies within individual large-scale time-series data sets. In contrast, in this current work we focus on user navigation among nodes of a data cube in search of rare temporal trends of interest to the user. Thus, the problem and methods of [8] are complementary to those considered in this current paper.
Our contributions are as follows.

- We formalize navigation in the search space of temporal trends in the data cube induced by the given fact table whose dimensions include a temporal dimension;
- We specify the problem of retrieval of unusual trends in the search space of temporal trends, and provide considerations and criteria for helping users determine which temporal trends could be unusual for their purposes;
- We consider exhaustive approaches for solving the trend-retrieval problem, and show that they are worst-case intractable;
- We propose tractable heuristic *trend-surfing* solutions for the trend-retrieval problem, in which the dependent attribute of interest can be either numeric or categorical; and
- We report experimental results on three real-life data sets; the results showcase the effectiveness and efficiency of our proposed approaches against the exhaustive baseline.

Paper outline. Section 1.1 reviews related work, and Section 2 specifies the trend- retrieval problem. Section 3 introduces our trend-surfing framework, and Section 4 presents approaches for assisting user navigation with trend surfing. In Section 5 we describe our implementation and report the results of experiments on three real-life data sets.

1.1. Related Work

Exploration of Time Series: Time-series exploration has recently attracted significant attention, see, e.g., works [8–15] focused on developing techniques for assisting experts in processing time-series data. The solutions range from novel indexing techniques [12, 13] to techniques for performing fast similarity search among time series [9], to user assistance in identifying relationships between different time-series data sets [14, 15], to languages and methods for helping users query time series using shapes of interest [10, 11]. None of these works focus on identifying unusual time series in the data. Metro-Viz [8] presents an approach for enabling better human-machine interaction for identifying anomalies in time series. We posit that the results of [8] and of this current paper are complementary, in that our approach concentrates on automatically suggesting unusual time series (temporal trends)

to the users. Results have also been reported on domain-specific detection of anomalous time series, see, e.g., [16, 17]. In contrast, our approach is domain independent and can apply to temporal data arising from diverse application scenarios.

Data-View Recommendation: Our work in this paper is related to the topic of automati- cally identifying and recommending interesting data views to facilitate data exploration, with the intent of maximizing insights from, and/or the value of, the underlying data to the users, see, e.g., [18–26]. Existing approaches are often based on data cubes [6, 7] and work with a variety of granularity levels of the views to be identified. For instance, [20–22, 25, 26] focus on identifying the aggregate queries whose results will be recommended to users as data views, while [23, 24] recommend subsets of results of aggregate queries.

Existing works in this area also vary in their definitions of scoring functions that de- termine the level of interestingness of candidate views. Options that have been considered include the level of deviation of a view from reference views [20], as well as the outputs of statistical analyses [23, 24] or of techniques based on machine learning [25], among oth- ers [22, 21]. The scoring functions tend to be completely or partially predefined in these works, whether based on the data-analytics experience of the authors, on the results of user studies, or on feedback from real-word data analysts [26]. In contrast, our approaches in this paper accommodate users' subjective interpretation of unusual views of their data, by accepting user-specified black-box scoring functions as indicators of the users' preferred definition of unusualness. Further, existing works in this area do not explicitly consider the temporal dimension of the given data, and typically do not automatically generate tem- poral trends for individual data-cube nodes. In some of the existing projects it is possible to examine the temporal dimension of the data as a regular dimension in the data cube. Unlike our work, these works support neither comparisons of large numbers of temporal trends, nor determinations of which temporal trends are unusual.

2. THE TREND-RETRIEVAL PROBLEMS

2.1. The Roadmap

We consider three versions of the problem of retrieval of unusual temporal trends. The first version is exhaustive; we show it to be worst-case intractable in Section 2.5. In the second version, the navigation in the search space of temporal trends is user driven, with the user selecting for exploration any number of nodes in the data cube that they deem necessary. Accordingly, the goal nodes in the search, that is, the nodes that contain the unusual temporal trends that the user deems interesting, are determined solely by the user. In this version of the problem, users may or may not be assisted in their navigation with suggestions of the next data-cube node to explore, see Section 4 for a discussion.

The third version of the problem is fully automated, both in the navigation and in the selection of the goal nodes and target (unusual) temporal trends. Clearly, in this version of the problem, some criteria on the goals and targets must be specified ahead of time. Our criteria of choice in this paper are the depth of the search in the search space, as well as objective criteria for ranking the target trends among the other trends at the same search depth. We discuss our ranking criteria in Sections 2.3–2.4, and report in Section 5 the results on the extent to which the solutions returned by our proposed approaches of Sections 3.1–3.3 satisfy these criteria.

When data navigation involves grouping and aggregation, one has to compute the relevant data-cube nodes [6, 7] from scratch along the way, or to retrieve precomputed

nodes whenever available. For this part of the search for unusual temporal trends, we rely on the mature state of the art in data-cube exploration, see, e.g., [6, 7, 27]. Our criterion of effectiveness and efficiency of the search is the number of nodes of the data cube that need to be examined to find an acceptable solution. Our goal is to minimize this number.

In the remainder of this section we define the search space of temporal trends for a data set (Section 2.2), discuss criteria for unusualness of temporal trends (Section 2.3), formalize the problems of finding unusual trends that we focus on in this paper (Section 2.4), and discuss optimal solutions to the problems (Section 2.5). The worst-case intractability of the optimal approaches will lead us to discuss, in Sections 3–4, our proposed heuristic approaches, with their effectiveness and efficiency results presented in Section 5.

2.2. The Search Space of Temporal Trends

Consider a relation *R* that has several dimension attributes of interest, a dependent (measure) attribute of interest, and a valid-time *temporal (dimension) attribute*, whose values are time points, such as years, dates, or timestamps. We are interested in studying *tempo- ral trends* that can be extracted from *R*, intuitively by tracking the value of the dependent attribute for all the available values of the temporal dimension, with the rest of the dimension values fixed after potentially grouping and aggregating the data in the relation. To formally define temporal trends, we start from a relational fact table, *R*, with several dimensions, including a valid-time [28] temporal dimension *T* whose values are time points, as well as a dependent attribute of interest, *M*. We distinguish between the temporal dimension *T* and the remaining *nontemporal* dimensions G1, G2, . . ., Gn of *R*, to which we will refer collectively as G. In the data cube [6] over the fact table *R*, consider nodes that can be constructed by evaluating SQL queries of the form

```
SELECT Gi, Gj, ... , Gk, T, AVG(M) FROM R GROUP BY Gi, Gj, ... , Gk, T;
```

here, Gi, Gj, . . ., Gk are some elements of the set G of all nontemporal dimensions of *R*. We require each such query to (i) have the temporal dimension *T* in the GROUP BY and the SELECT clause, and to (ii) have AVG(M) in the SELECT clause. (This exposition can be naturally generalized to arbitrary aggregation functions.) For each such SQL query, we index the query output on *R* with the set H of all nontemporal GROUP BY attributes of the query, and call each such relation with index H ⊆ G *the* H-*node of the data cube for R*.

We define the *multidimensional T-space* T (*R, AV G*) for temporal fact table $\boldsymbol{R}$ as the set of H-nodes of the data cube for *R*, for all subsets of the set G of nontemporal dimensions of *R*. Consider an arbitrary element $\boldsymbol{E}$ of T (*R, AV G*), which is the H-node of the data cube for $\boldsymbol{R}$ for some H = { Gi, Gj, . . ., Gk }. The relation $\boldsymbol{E}$ has attributes, from left to right, Gi, Gj, . . ., Gk, T, and AVG(M). Given an arbitrary tuple $s = (\mathbf{gi,gj,}. . .\mathbf{,gk,t,a})$ in *E*, we can associate with *s* the following $(\mathbf{gi,gj,}. . .\mathbf{,}gk)$-*projection query* on *E*:

```
SELECT * FROM E WHERE Gi = gi AND Gj = gj AND ... AND Gk = gk;
```

Consider the relation $\mathbf{E(gi,gj,}. . .\mathbf{,gk)}$ that results from applying the $(\mathbf{gi,gj,}. . .\mathbf{,gk)}$-projection query on *E*. E(gi,gj,. . .,gk) includes all the time points and the corresponding values of the dependent attribute that are associated in the relation *E* with the values gi, gj,. . ., gk of Gi, Gj, . . ., Gk. Recall that $\boldsymbol{E}$ is indexed with a subset H = { Gi, Gj, . . ., Gk } of the set G of nontemporal dimensions of *R*. We call the relation E(gi,gj,. . .,gk) *the temporal trend for tuple* $(\mathbf{gi,gj,}. . .\mathbf{,}gk)$ *in the* H-*node of the data cube for R*.

For a given temporal fact table *R*, a node of the multidimensional *T* -space T (*R, AV G*) can, in general, contain multiple temporal trends. Our final step in formalizing the *search space* A(*R, AV G*) *of temporal trends for R* consists in assigning each node of T (*R, AV G*), with its index H and with all its temporal trends, to the section of the search space whose *(grouping) depth* in the space is the cardinality of the set H. There are two special boundary cases: First, all the raw data of the fact table *R* can be found in the only node in the space at depth *n*, where *n* is the number of nontemporal dimensions of *R*. Second, the search space has exactly one node at depth zero with a single trend that averages the values of the dependent attribute grouped only by the temporal dimension.

As an example, Fig. 2(b) shows all the nodes and their navigation links in the search space for the fact table of Fig. 2(a). Navigation from node *A* at depth *k* in the search space A(*R, AV G*) to node *B* at depth *k* + 1 of A(*R, AV G*) is performed by adding a single extra nontemporal dimension to the SELECT and GROUP BY clauses of the SQL query that defines node *A*; we say that *B* is a *child node of node A*.

2.3. Which Temporal Trends Are Unusual?

We now discuss which temporal trends could or should be considered unusual. We take the general view [1–3] that the notion of outlier is subjective. For the specific purpose of this work, we assume that domain experts rate temporal trends arising from the temporal fact table of interest by using a *scoring function*, which gives each temporal trend in a set of trends an *outlier score*. We now provide an intuition for outlier scoring, and then cover our requirements on scoring functions and give specific examples.

To develop an intuition for outlier scoring in a set of temporal trends, let us consider an (extreme) special case, in which all the tuples in the given fact table *R* have the same temporal value. Let *N* be an arbitrary node in the search space of temporal trends for *R*; *N* is a relation with attributes Gi, Gj, . . ., Gk, T, AVG(M), where Gi, Gj, . . ., Gk are some nontemporal dimensions of the fact table *R*, T is its temporal dimension, and M is the measure attribute of interest. (See Section 2.2 for the details.) Then all the temporal trends in the node *N* can be visualized as points in a multidimensional space with dimensions Gi, Gj, . . ., Gk, AVG(M). We next compute the *center of mass* $C_M(N)$ *for the node N* as the average of the values of AVG(M) in all the trends of *N* . Then the *Euclidean outlier score* for each temporal trend in the node *N* can be computed as the Euclidean distance between the point for the trend and the hyperplane for the center of mass $C_M(N)$ in the multidimensional space. Clearly, the larger the difference between the value of AVG(M) in a temporal trend *tr* and the center-of-mass value $C_M(N)$, the higher the Euclidean outlier score for the trend *tr*. We take as *unusual* those temporal trends in the node *N* whose outlier score is unusually high compared to the other trends; depending on the cutoff value for unusualness, there may be more or fewer unusual trends in any given node. See Fig. 3 for an illustration of center of mass for the case of one grouping attribute G; ignore for now the discussion of local and nonlocal unusualness of temporal trends.

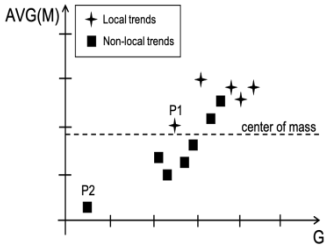

Fig. 3: Example of local vs global unusualness of temporal trends using a fact table with a nontemporal dimension G (the X-axis), dependent attribute M (see AVG(M) in the Y-axis), and a temporal dimension with a single temporal value. While $P1$ is the most unusual trend within its data-cube node, it is very close to the center of mass of all the given trends. A trend $P2$ that is not local to that node is the most unusual trend globally.

This special case extrapolates naturally to the general case, in which a node in the search space of temporal trends for a fact table can have multiple temporal values. In that general case, we still use the (appropriately generalized) notions of *center of mass* and of *Euclidean outlier scoring* for temporal trends, and also extend these notions from the case of trends in specific nodes in the search space to arbitrary sets of given temporal trends. Consider now an arbitrary scoring function, which provides a nonnegative numeric "outlier" score to each trend in a given set of temporal trends. Such a function can be taken off the shelf, as in the case of the above Euclidean outlier scoring, or it can be provided by domain experts for the task and fact table at hand. While we view the given scoring functions as black boxes, our approach to ruling out "absurd" functions is by declaring a scoring function to be *reasonable* if it satisfies the condition of being distance proportional *in the expectation*. That is, an outlier-scoring function f is *distance proportional with confidence level* L if, for each pair (a, b) of trends in a given set of temporal trends, we are confident at the confidence level L that the score $f(a)$ for the trend a is higher than $f(b)$ for trend b if and only if the Euclidean outlier score for a is higher than that for b. Common outlier-scoring functions include Principal Component Analysis (PCA) [29], k-Nearest Neighbors (kNN) [30], Clustering-based Local Outlier Factor (CBLOF) [31], and Isolation Forest [32]. We use PCA, kNN, and CBLOF in our experiments, see Section 5. We have verified that these outlier-scoring functions are all distance proportional with confidence level 95%. While our approaches could still work with scoring functions that do not satisfy this distance-proportionality requirement, observing the requirement confers to our approaches the expectation of reasonableness of their outlier scoring.

2.4. The Problem Statements

We now specify the problems that we focus on in this paper. While both problem statements are general, in that they refer to an arbitrary aggregation function *agg*, in Sections 3–5 we will focus on the versions of the problems in which *agg* is the function *AVG*.

Problem statement 1: Retrieval of unusual temporal trends at a given depth in search space of trends. Given (i) a temporal fact table R with n nontemporal dimensions, a temporal dimension, and a dependent attribute, (ii) an aggregation function *agg*, (iii) an outlier-scoring function f, (iv) the target grouping depth k, $0 \leq k \leq n$, and (v) an unusualness criterion *cr*. Our goal is to identify one or more temporal trends $tr_1, tr_2, \ldots, tr_p$ in nodes at depth k in the search space of temporal trends built for R using the function *agg*, such that each tr_i, $1 \leq i \leq p$, satisfies the criterion *cr* with respect to (w.r.t.) the function f.

That is, given a temporal fact table about facts in some application domain, our goal is to return to domain experts unusual temporal trends at a level of granularity that is appropriate for their analysis tasks, i.e., at the level of groupby and aggregation on the input fact table. The question is, of course, what makes a trend unusual in a given application domain. Intuitively, we strive to return to domain experts the temporal trends with the highest outlier scores as given by the outlier-scoring function chosen by the experts. That is, we can consider the outlier-scoring function itself to be a proxy for the definition of unusualness in the domain. There are two problems with this direction of thinking: First,

domain experts may have trouble selecting the right outlier-scoring function for their domain. Second, an optimal approach to finding the most unusual trends might not be time efficient. Indeed, as we show in Section 2.5, exhaustive search for the most unusual temporal trends at a depth of interest turns out to be intractable for a range of depths in the search space of trends for the input fact table.

To address these challenges in determining unusual trends in a specific application domain, we develop automated domain-independent approaches for retrieving unusual temporal trends from a given temporal fact table. Toward addressing the first challenge, we provide to domain experts a library of outlier-scoring functions along with the expectations of their reasonableness, see Section 2.3. Toward addressing the second challenge, we specify two unusualness criteria for temporal trends at a given depth:

A temporal trend is unusual if it is one of the top 5% outlier-scoring trends (we choose the 5% range as the commonly used cutoff value in statistical tests); and Alternatively, a temporal trend is unusual if it is an element of the top-scoring outlier cluster, where the cluster is determined by an off-the-shelf clustering method.

We posit that these two unusualness criteria are natural and domain independent, and present experimental results in Section 5.

As discussed in Section 2.1, we are also interested in user-driven approaches to selecting unusual trends from the data cube for the given data set. For the user-driven version of the problem of trend retrieval, the problem statement is as follows:

Problem statement 2: User-driven retrieval of unusual temporal trends from search space of trends. Given (i) a temporal fact table $\boldsymbol{R}$ with n nontemporal dimensions, a temporal dimension, and a dependent attribute, (ii) an aggregation function *agg*, (iii) an outlier-scoring function $\boldsymbol{f}$, (iv) an unusualness criterion *cr*, and (v) a node N at depth k ($0 \leq k \leq n - 1$) in the search space of temporal trends built for $\boldsymbol{R}$ using the function *agg*. Our goal is to identify a node $\boldsymbol{N'}$ at depth $k + 1$ in the search space, where $\boldsymbol{N'}$ contains one or more temporal trends $tr_1, tr_2, \ldots, tr_p$ such that each trend $\boldsymbol{tr_i}$, $1 \leq \boldsymbol{i} \leq p$, satisfies the criterion *cr* w.r.t. the scoring function $\boldsymbol{f}$. Here, the node N mentioned in (v) is the node at which the user resides at a given point in their exploration of the search space, and the output node $\boldsymbol{N'}$ is the suggested node for the user to examine next.

2.5. Exhaustive Retrieval of Unusual Trends

Given a fact table R with n nontemporal dimensions, a brute-force procedure for retrieving any number of top-unusual trends at a given depth k in the search space for R would rank all the trends in the nodes of the space at depth k using the given outlier-scoring function, and then return the needed number of top-ranked trends. The solution would clearly be optimal, given its exhaustive nature. Unfortunately, the approach can be intractable:

Theorem 1. *For a range of target depths k around depth $\frac{n}{2}$ in the search space of trends for fact table $\boldsymbol{R}$ with n nontemporal dimensions, the time complexity of the brute-force procedure is in* $\Omega(a^n)$, $a > 1$.

That is, for any depth k that is "close enough" to the depth $n/2$ of the search space of trends for the fact table R, the runtime of the brute-force procedure is exponential in n.

The importance of this result for our problem statements of Section 2.4 is in the following.

For the problem of retrieving unusual trends at a given depth of the search space, it means that for some depths (those that are "close enough" to the depth *n*/2), the exhaustive optimal approach, which would consist of one call to the brute-force procedure, would be intractable. For user-driven retrieval of unusual trends, the implications are even stronger, as the exhaustive approach addressing this problem would be calling the brute- force procedure at each successive depth. Due to this analysis, we give up on exhaustive solutions and, in the remainder of the paper, explore approximate solutions instead.

3. THE TREND-SURFING SOLUTIONS

As we saw in Section 2.5, the exhaustive approach to retrieving unusual temporal trends from a given fact table is worst-case intractable. Accordingly, we resort to heuristic-based solutions. In particular, we switch from the criteria *cr* for top-unusual trends in the problem statements of Section 2.4 to *local best-effort* criteria. The intuition is for the proposed approaches to do the best they can locally at each step. In Section 5.2 we will discuss the performance of the proposed approaches with respect to the criteria *cr* of Section 2.4.

In this section we focus on solutions to the problem of retrieving unusual temporal trends at a given depth in the search space of trends for the given fact table, see Problem 1 in Section 2.4. Intuitively, our proposed approaches start out from the single most aggre- gated node with a single temporal trend at depth zero of the search space, and navigate iteratively between adjacent depths of the space, always in the direction of the "highest- outlier" node at the next depth. We could view this way of traversing the search space of temporal trends as "surfing" the search space. For this reason, we call the proposed approaches *TrendSurfer I* and *TrendSurfer II*. The difference between the approaches is in their criteria for the highest-outlier node at the next depth, see Section 3.2.

3.1. The Trend-Surfing Framework

Algorithm 1: Retrieving unusual trends via trend surfing.

Data: Temporal fact table R with n nontemporal dimensions, a temporal dimension, and a dependent attribute; outlier-scoring function f; target grouping depth k, $0 \leq k \leq n$; parameter p that determines which TrendSurfer procedure will be called as a subroutine.

Result: Temporal trend tr from H-node at depth k in $A(R, AVG)$, where tr is the best-effort top-outlier trend across all the nodes examined at depth k.

```
1  begin
2      Set up A(R, AVG) for navigation from depth 0; // search space of trends for R
3      tr_0 ← the sole trend at depth zero in A(R, AVG);
4      H ← ∅; // index of target node at depth k
5      while H.cardinality < k do
6          N ← H-node in A(R, AVG);
7          if p = 1 then
8              (H', tr') ← TrendSurferI(A(R, AVG), N, f);
9          else
10             (H', tr') ← TrendSurferII(A(R, AVG), N, f, tr_0);
11         H ← H'; tr ← tr';
12     return H, tr;
```

We now present our proposed framework for retrieving one best-effort unusual trend at a given depth in the search space A(R, AVG) of trends for the given temporal fact table R, see Algorithm 1 for the pseudocode. The first operation is to set up the search space A(R, AVG) (line 2; ignore line 3 for now). As mentioned in Section 2.1, for performing efficient

grouping and aggregations needed to construct nodes in the space A(R, AVG) we rely on the mature state of the art in exploring data cubes in data warehouses, see, e.g., [6, 7, 27]. Moreover, in our proposed approaches we *do not* precompute the entire space A(R, AVG); rather, we *open* each node in the search space only at the point at which the node needs to be explored. Accordingly, line 2 in Algorithm 1 suggests opening only the sole depth-zero node of the search space A(R, AVG). Similarly, the procedures of Algorithms 2–3 have A(R, AVG) as their input argument only for clarity of the exposition; in reality, all that needs to be passed to the procedures is a pointer to the fact table R and to any nodes that have already been opened.

The main loop in lines 5–11 of Algorithm 1 shows iterative navigation between adjacent depths in the search space A(R, AVG), starting from the single temporal trend at depth zero, always in the direction of the *top-outlier* node at the next depth. The exact notion of top outlier in the algorithm depends on the input outlier-scoring function f, as well as on the choice of the *TrendSurfer* approach **(*I* or *II*)**, governed by the input parameter p (lines7–10). To return the node to be explored next, together with its best-effort top-outlier trend, each call of the chosen *TrendSurfer* procedure opens and examines all the child nodes in A(R, AVG) of the procedure's input node. The overall iterative surfing of the search space A(R, AVG) in k steps, from depth zero to the target depth k, results in: (1) creating the k-attribute index H of the target node at depth k, one attribute per iteration, and (2) returning the best-effort top-outlier trend tr of the target H-node (line 12).

We measure the runtime complexity of Algorithm 1 in the number of nodes of the search space A(R, AVG) that must be opened (computed) for the algorithm to work. Measured this way, the runtime complexity of Algorithm 1 is by construction quadratic in the total number $n + 1$ of depths of A(R, AVG), where n is the number of nontemporal dimensions of the input fact table R:

Theorem 2. *The runtime complexity of Algorithm 1 is in* $O(n^2)$.

Recall that the runtime complexity of the exhaustive approach of Section 2.5, measured the same way, is worst-case exponential in n.

3.2. The TrendSurfer Procedures

Consider a node N at depth $i < n$ in the search space of trends A(R, AVG); we observe that if N contains trends that can be viewed as outliers for depth i in A(R, AVG), then trends in at least one *child node* of N in A(R, AVG) tend to be outliers for depth $i + 1$. This observation follows from properties of the function AVG, which we use for creating the space A(R, AVG) in the first place. Further, when a trend tr is a *global* outlier for some depth i in A(R, AVG), then tr tends to also be a *local* outlier within its node. For example, the top trend in the {DestAirport}-node at depth 1 in Fig. 2(b) is the only trend at this depth with a steep descent. We can see that at depth 2, the two children nodes of the {DestAirport}-node both contain trends that resemble the outlier trend in their parent and that happen to also be global outlier trends for depth 2, as well as local outliers within their nodes. At the same time, the {OriginAirport,Airline}-node, which is not a child of the {DestAirport}-node, does not contain outlier trends for depth 2.

These observations give rise to the basic best-effort heuristic strategy of the procedure *TrendSurfer I*, see Algorithm 2. When exploring a node N at depth i in the search space A(R, AVG) (recall line 6 in Algorithm 1), *TrendSurfer* ***I*** calls procedure *TopChildTrends* (line 2 of Algorithm 2), which opens all the children $N_1, N_2, \ldots, N_m$ of N

in $\mathbf{A}(\boldsymbol{R}, AV\,G)$ and produces the trend $\boldsymbol{tr}_j$ for each child $\boldsymbol{N}_j$, $1 \leq \boldsymbol{j} \leq m$, such that the score of $\boldsymbol{tr}_j$ *within* N_j is the top-outlier score according to the function f. *TrendSurfer I* then selects from the outputs of *TopChildTrends* the index H' of the child node of N whose trend's f-score is the highest among all the trends returned by *TopChildTrends* (line 3). In line 4, *TrendSurfer I* returns the selected node index and the corresponding trend.

By design, for each node N that is examined by Algorithm 1 in the search space A(R, $AV\,G$), *TrendSurfer I* guides the navigation to the child of N whose trend's f-score is maximal among the top f-scores within all the children of N. Thus, the intuition of the strategy is that of local top-outlier trends also being global top-outlier trends. While being productive in many cases, as exemplified by the experimental results that we report in Section 5.2, this intuition is in some cases misleading. Consider Fig. 3, which shows the trends of several nodes at the same depth of a search space of trends. While the trend $P1$ in Fig. 3 is the top outlier within its node w.r.t. the Euclidean-distance outlier-scoring function, it is clearly not the top outlier globally at the given depth. In fact, we can see that $P1$ is close to the *center of mass* of all the trends at that depth (see Section 2.3).

Algorithm 2: Procedure TrendSurfer I.

Data: Search space $A(R, AV\,G)$ for temporal fact table R with n nontemporal dimensions;
H-node N at depth $i < n$ in $A(R, AV\,G)$; outlier-scoring function f.
Result: Index H' of a child N' of N in $A(R, AV\,G)$ and a trend tr' of N', such that the f-score of tr' is maximal among the top f-scores within all the children of N in $A(R, AV\,G)$.

```
1 begin
2     S ← TopChildTrends(A(R, AV G), N, f);
3     (H', tr') ← pair from S whose trend's f-score is the highest among all the trends in S;
4     return (H', tr');
```

The intuition here is that some trends that are top outliers locally within their nodes could still be close to the center of mass when considered globally at a given depth.

In view of this intuition, we modify the strategy of *TrendSurfer I* while still keeping it local. The pseudocode for the resulting procedure *TrendSurfer II* is shown in Algorithm 3. Similarly to *TrendSurfer I*, *TrendSurfer II* works with the output of the procedure *TopChildTrends* (line 2). However, unlike *TrendSurfer I*, *TrendSurfer II* selects from the nodes returned by *TopChildTrends* the node whose local top-outlier trend is likely to be the farthest from the center of mass at the depth of these nodes.

Clearly, calculating exactly the center of mass of all the trends at a given depth requires opening all the nodes at that depth. This operation is worst-case intractable, see Section 2.5. Thus, we resort to using in our approach a proxy for the exact center-of-mass trend. It turns out that the sole trend tr_0 at depth zero of the search space of trends is a convenient proxy center of mass for each depth in the space; our measurements show that tr_0 is very close to the center of mass at any depth of the search spaces of trends for the data sets used in our experiments of Section 5. Accordingly, we generate the trend tr_0 in line 3 of Algorithm 1 and pass it as an argument in calls to the procedure *TrendSurfer II* (line 10). Lines 3–8 of Algorithm 3 show how *TrendSurfer II* selects and returns the child of its input node N by the criterion of the child's top-outlier trend being the farthest from the trend tr_0. For example, in the setting of Fig. 3 *TrendSurfer II* would return the node whose top-outlier trend is $P2$, rather than the node whose top-outlier trend is $P1$.

Algorithm 3: Procedure TrendSurfer II.

Data: Search space $A(R, AVG)$ for temporal fact table R with n nontemporal dimensions; H-node N at depth $i < n$ in $A(R, AVG)$; outlier-scoring function f; sole trend tr_0 at depth zero of $A(R, AVG)$.
Result: Index H' of a child N' of N in $A(R, AVG)$ and a trend tr' of N', such that tr' is the farthest from tr_0 among the top f-scoring trends within all the children of N in $A(R, AVG)$.

```
1 begin
2     S ← TopChildTrends(A(R, AVG), N, f);
3     Ḣ ← H; ṫr ← tr0; maxScore ← −∞;
4     for each element (Ḧ, ẗr) of S do
5         d ← EuclideanDistance(ẗr, tr0);
6         if d ≥ maxScore then
7             maxScore ← d; Ḣ ← Ḧ; ṫr ← ẗr;
8     return (Ḣ, ṫr);
```

We have shown that each step through the search space of trends A(R, AV G) using the *TrendSurfer II* strategy takes us at least as far away from the center of mass at depth zero of the search space as the previous step. Further, Algorithm 1 with the *TrendSurfer II* strategy can return optimal top-unusual (global-outlier) trends: We have found that if there is a single global outlier trend, then this approach is likely to return it and, if there is a group of global outliers, then the same approach is likely to return one of them.

3.3. Retrieving Multiple Unusual Trends

We now discuss how to return to the user multiple unusual trends at the specified depth of the search space of trends. (The algorithm modifications that are required to accommodate the solutions are straightforward and are omitted due to the space limit.) Multiple trends can be returned either in several runs of Algorithm 1 or in a single run. In the former case, a randomizer is use to perturb the f-scores of trends in the procedure *TopChildTrends*, to achieve a degree of randomness in the scores. This way, different best-effort top-outlier trends are likely to be returned in different runs of Algorithm 1.

To return multiple top-outlier trends in a single run of Algorithm 1, the procedure *TopChildTrends* would rank by the f-score all the trends in all the children of its input node N in the search space A(R, AV G) and, if needed, also trends in nodes adjacent to the children of N. Then a desired number of top-ranked trends from this ranking can be returned when the target depth k is reached in the main loop of Algorithm 1.

3.4. Categorical Dependent Attributes

We now consider the case in which the dependent attribute of interest in the fact table R takes categorical values. Recall the example from Section 1, in which the attribute Ampicillin (AMP) R/S/I in the spreadsheet in Fig. 1, indicating degree of resistance of serotypes to antimicrobials, takes values “resistant” (R), “susceptible” (S), and “intermediate” (Int). In our proposed approach for this case, we convert categorical values into numbers using the common choice of one-hot vectors [1]; e.g., the value S in the top tuple t in Fig. 1 would be converted to the vector (0,1,0) indicating that S is the only value present from among (R,S,Int). The vectors in the tuples would then be flattened using a newly added numeric dependent attribute MNUM. For the tuple t in our example of Fig. 1, three tuples will be generated, with each tuple having the respective code in the original dependent attribute M: code S for the value 1 of MNUM, and R and Int for each value 0 of

MNUM in the vector (0,1,0). The approach of Section 3.1 would then apply to the new numeric dependent attribute MNUM. (The attribute M with its codes will serve as an additional nontemporal dimension alongside the temporal dimension T in the GROUP BY clauses of all the SQL queries that create nodes of the search space A(*R, AV G*) of trends for the table *R*.) Section 5.2 presents our experimental results for this modification of the approach of Section 3.1 for the case of categorical dependent attributes.

4. USER-DRIVEN TREND NAVIGATION

In Section 3 we detailed automated approaches to returning best-effort top-outlier tem- poral trends for a given fact table *R*. Recall that for these approaches to work, the target depth in the search space of trends for *R* needs to be prespecified. We are now addressing this limitation, by considering the problem of *user-driven* temporal-trend retrieval. In this version of the problem, the user would select for exploration any number of nodes in the space that they deem necessary. Accordingly, the nodes that contain the unusual temporal trends of interest to the user would be determined solely by the user.

In this section we restrict the discussion to the problem of users retrieving a single temporal trend of interest to them; the generalization to the case of multiple trends is straightforward. We also assume that users always start their search paths from depth zero of the search space, and that they do their navigation using the parent-child relationships between nodes at adjacent depths in the search space.

In looking for a trend of interest, users may or may not be assisted in their navigation with suggestions of the next node to explore. Consider first the case of unassisted user navigation. We define the *length of a user-navigation path* as the number of forward steps (i.e., from a node at depth m to a node at depth $m + 1$) in the path, with backward steps in backtracking not contributing to the calculation. Here, under natural assumptions, such as that all user paths in the search space are equally likely and that users do not go from a parent node to the same child node more than twice, we obtain the following result:

Theorem 3. *The expected path length in unassisted user navigation in the space of tem- poral trends is in* $O(n^2)$*, with n the number of nontemporal dimensions in the table R.*

Now consider the version of the problem in which users can involve in their navigation the *TrendSurfer* procedures of Section 3.2, see Problem 2 in Section 2.4. Here, *TrendSurfer I* or *TrendSurfer II* would make suggestions for the users at each node in the user's other- wise unrestricted path. The user would then be free to follow or decline each suggestion. In the extreme case of users following *TrendSurfer* suggestions at each step of the path, they would get the same effectiveness results as those reported in Section 5.2 for the automated approaches of Section 3, provided the users' unusualness criteria coincide with the criteria (A)–(B) of Section 2.4. Moreover, the length of the *user-perceived* path to the goal node equals the number of nontemporal dimensions in the node. Thus, this (optimal) length is linear in the number of nontemporal dimensions in the fact table *R*; contrast this with the result of Theorem 3. The opposite case, in which the users decline all the *TrendSurfer* suggestions, is the same as that of unassisted user navigation.

In the intermediate case, in which users follow some *TrendSurfer* suggestions and de- cline others, the problem of reaching nodes that have the most unusual trends by our criteria of Section 2.4 becomes very hard. Indeed, at any point in the navigation the users can choose to follow *TrendSurfer* suggestions from a node whose children do not contain unusual trends; being

local, the procedures are then unlikely to give good suggestions. On the other hand, with the navigation being at least partially user driven, the users are free to apply their own unusualness criteria for trends, instead of having to work with those that are hardwired into the *TrendSurfer* procedures. In Section 5.3 we show that in this intermediate case, *TrendSurfer* suggestions have the potential to significantly shorten the length of the user-perceived path to the user's goal node.

5. IMPLEMENTATION AND EXPERIMENTS

In this section we report the results of experiments involving three real-life data sets that focused on the performance of the proposed approaches of Sections 3–4. We also report the results of a user study. The results showcase the effectiveness and efficiency of our approaches against the exhaustive baseline, as well as their potential to improve user experience in user-driven retrieval of unusual temporal trends of interest to the users.

5.1. Implementation and Experimental Settings

For brevity, we will be referring to the algorithm of Section 3.1 with the *TrendSurfer I* procedure (the *TrendSurfer II* procedure, respectively) of Section 3.2 as the *TrendSurfer I*

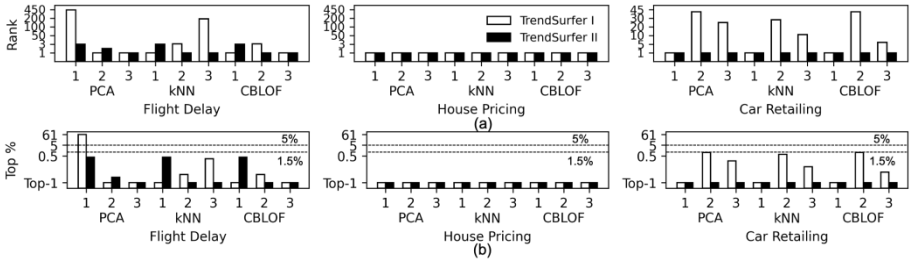

Fig. 4: Ranks of the outputs *tr* of the two *TrendSurfer* algorithms against the ranked ground-truth trends at the same depth, for numeric dependent attributes in the data. The results are reported for 27 test cases resulting from combining the three test data sets with values 1 through 3 of the target depth in the search space for each data set and with the selected outlier detectors. The X-axes specify the test cases; in (a), the (logarithmic) Y-axis shows the absolute ranks of the trends *tr* against the ground truth, while in (b) the (logarithmic) Y-axis shows the percentage ranks of *tr* against the ground truth.

algorithm (the *TrendSurfer II algorithm*, respectively). We have implemented the exhaustive approach of Section 2.5, the algorithms *TrendSurfer I* and *TrendSurfer II*, and all the programs used in the experiments in Python 3.8 using common data-analysis packages, including Pandas, Numpy, scikit-learn, and PyOD. The experiments were conducted on a MacBook Pro with a 64-bit MacOS, 2.70 GHz Quad-Core Intel Core i7 processors, and 16 GB of 2133 MHz LPDDR3 memory. For each experiment, the relevant data sets were preloaded into memory. We ran ten trials for each effectiveness and efficiency experiment.

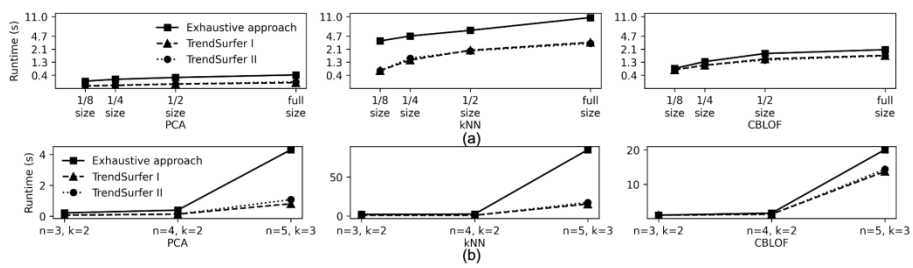

Fig. 5: Runtime efficiency of the two *TrendSurfer* algorithms against the exhaustive baseline for *Flight Delay* data [33], shown in (a) for fractions 1/8, 1/4, 1/2, and 1 of all the tuples in the data, and in (b) for three data sets with numbers $n = 3$, 4, and 5 of nontem- poral dimensions formed from the original data and normalized for size, see the text for the details. The target depth k is $\lceil n/2 \rceil$ in all the cases. (In (a), the Y-axis is logarithmic.)

Our real-life data sets come from three domains: flight delays, house pricing, and car retailing. (We selected data sets with limited schema and data sizes, due to having to use the exhaustive approach of Section 2.5 to obtain the ground-truth temporal trends on each data set.) For the *Flight Delay* data set, we used the data from [33] with flight- delay values in minutes for USA in January 2019, using a subset of the data with four nontemporal dimensions, dependent attributes *Arrival Delay* (numeric) and *ARR DEL15* (categorical), and a temporal dimension for flight dates. The data set has over 565K tuples. The *House¯ Pricing* data set [34] has over 678K tuples with monthly house-price values in the USA between 2015–19. It has four nontemporal dimensions, a numeric dependent attribute *Price*, and a temporal dimension for the months in which the data were collected. Finally, the *Car Retailing* data set [35] has over 221K tuples with sales information for second-hand cars in Germany for several decades. It has four nontemporal dimensions, a numeric dependent attribute *Price*, and a temporal dimension for the sales dates.

Three popular outlier detectors for temporal data, *Principal Component Analysis (PCA)* [29], *Nearest Neighbors (kNN)* [30], and the *Clustering-Based Detector (CBLOF)* [31], all implemented in PyOD [36], provided the outlier-scoring functions in the experi- ments. As these detectors interpret outliers in different ways, this choice lends generality to our experimental results [1, 3]. The data sets were preprocessed via normalization and imputation of missing values, to enable correct application of the outlier detectors.

5.2. Effectiveness and Efficiency for Problem 1

We first report performance results for the problem of retrieving a single unusual temporal trend at a given depth in the search space of trends for a given fact table, see Section 2.4. In summary, our results suggest that *TrendSurfer II* outperforms *TrendSurfer I* in almost all cases, and is able to efficiently identify temporal trends at a given depth in the search space, with its outputs almost always satisfying the unusualness criteria of Section 2.4.

For the experiments with numeric dependent attributes in the test data sets we selected three values, 1 through 3, of the target depth k of the search space of trends for each test data set. We then combined each value of k with each selected outlier detector and with each test data set, focusing on their numeric dependent attributes. Finally, we ran both algorithms of Sections 3.1–3.2 on the 27 test cases. The outputs of the 54 runs were evaluated against the ground-truth results obtained by using the exhaustive approach of Section 2.5. In each of the 27 test cases, the ground-truth trends had at least one significantly unusual trend according to the outlier detector for that test case.

Fig. 4(a)–(b) shows the **effectiveness results for numeric dependent attributes** for the 54 test runs with respect to the unusualness criterion (A) (top 5% of the ground truth) of Section 2.4. The figure reports the absolute rank of each trend tr output by our algorithms against the ground-truth trends ranked by the applicable outlier-detector score (a), as well as the percentage score for tr against the ranked ground-truth trends (b). While the absolute ranks of the algorithm outputs can vary, in Fig. 4(b) we can see that the outputs of

TrendSurfer I are within the top 5% – and even within the top 1.5% – of the ranked ground-truth trends in 26 test cases out of the 27. Moreover, the outputs of *TrendSurfer II* are within the top 1.5% of the ranked ground-truth trends in all 27 cases. In summary, the *TrendSurfer II* algorithm outperformed *TrendSurfer I* in almost all the cases, while satisfying the unusualness criterion (A) of Section 2.4 in all the cases.

We also achieved good results for the same 54 test runs with respect to the unusualness criterion (B) (top cluster for the ground truth) of Section 2.4. We used *K-Means* to cluster each of the 27 ground-truth results, with the numbers of clusters varying from 2 to 15 for each test case. *TrendSurfer II* outperformed *TrendSurfer I* in most of the cases. Further, the outputs for *TrendSurfer II* were in the top-scoring outlier cluster in all the cases for the data sets *House Pricing* and *Car Retailing*. Table 1 shows the results for the *Flight Delay* data set; we can see that the output of *TrendSurfer II* is not always in the top ground- truth cluster. We believe that this performance can be explained by *K-Means* being the most closely related to *kNN*, as they are both based solely on Euclidean distance. In contrast, PCA is based on variance, and CBLOF incorporates other factors. We posit that with criterion (B) of Section 2.4, it is important to keep in mind that the chosen clustering algorithm should fit the users' subjective definition of outliers with respect to their problem and data set, just as the chosen outlier detector should.

Table 1: Effectiveness of the two *TrendSurfer* algorithms with respect to the unusualness criterion (B) of Section 2.4 for the *Flight Delay* data set, with focus on its numeric dependent attribute. We used *K-Means* to cluster each of the ground-truth results, with the numbers *c* of clusters varying from 2 to 15 in each case. The *N* column shows the values of *c* for which the algorithm outputs are found in the top ground-truth cluster of trends; the % column shows the proportion of the *N* values among the total 14 values of *c* considered.

Outlier detector	Target depth	TrendSurfer I		TrendSurfer II	
		N	%	*N*	%
PCA	1	[]	0.0	[2,11]	71.4
	2	[2,15]	100.0	[2,8]	50.0
	3	[2,15]	100.0	[2,15]	100.0
kNN	1	[2,15]	100.0	[2,14]	92.9
	2	[2,15]	100.0	[2,15]	100.0
	3	[2,5]	28.6	[2,15]	100.0
CBLOF	1	[2,15]	100.0	[2,13]	85.7
	2	[2,10]	64.3	[2,15]	100.0
	3	[2,15]	100.0	[2,15]	100.0
Average			77.0		88.9

Next we discuss the results of Theorems 1–2 in practice, see Fig. 5. These **efficiency results for numeric dependent attributes** were obtained for the following experimen- tal settings. We chose the *Flight Delay* data set, since it has more temporal trends than the other data sets. For the results of Fig. 5(a), we varied the size of the data set, by looking at the fractions 1/8, 1/4, 1/2, and 1 of the number of tuples in *Flight Delay*. We ran both *TrendSurfer* algorithms and the exhaustive baseline on each fractional data set, in combination with each outlier detector, for the target depth $n/2 = 2$ in the search space of trends, where n is the number of nontemporal dimensions.

For the results of Fig. 5(b), we varied the number of nontemporal dimensions in the *Flight Delay* data set, by going back to the online source [33] and importing from it the version $FlightDelay^{+}$ of *Flight Delay* with five (rather than the originally chosen four) non-temporal dimensions. We then produced out of $FlightDelay^{+}$ a projection $FlightDelay^{-}$

with three nontemporal dimensions; finally, we randomly sampled each of $FlightDelay^{+}$, *Flight Delay*, and $FlightDelay^{-}$ to obtain three test data sets F^{+}, F, and F^{-}, each with 30K tuples. On the data sets F^{+}, F, and F^{-} we ran the two *TrendSurfer* algorithms and the exhaustive baseline in combination with each of the three outlier detectors, for the target depth $\lceil n/2 \rceil$ in the search space of trends for each of the data sets.

For both settings, we report in Fig. 5 the total runtime of each algorithm in each test case, including the time for creating all the needed nodes in the search space of trends and all the runtimes for each outlier detector. We can see that both *TrendSurfer* algorithms tend to reduce the runtimes in comparison with the baseline approach more as the data size grows, e.g., 1.62 sec vs 2.06 sec for full-size data set with CBLOF in Fig. 5(a). Since PCA and CBLOF were working faster for us than kNN, our algorithms improved the efficiency more for kNN. Indeed, running on fewer nodes is the key advantage that *TrendSurfers I–II* have over the exhaustive baseline. Note that the runtime of *TrendSurfer II* is on par with that of *TrendSurfer I*, even though *TrendSurfer II* significantly outperforms *TrendSurfer I* in terms of effectiveness, see the discussion of Fig. 4.

Finally, we discuss our **effectiveness results for categorical dependent attributes**, which are based on the binary attribute *ARR_DEL15* of the *Flight Delay* data set. We used the approach of Section 3.4 to preprocess the data set, and obtained from the result- ing data the ground truth and the outcomes of 18 runs of our algorithms for 9 test cases.Table 2: Effectiveness of the two *TrendSurfer* algorithms with respect to the unusualness criteria (A)–(B) of Section 2.4 for the *Flight Delay* data set, with focus on its categorical dependent attribute. We used *K-Means* to cluster each of the ground-truth results, with the numbers of clusters varying from 2 to 15 for each test case.

Outlier detector	Target depth	TrendSurfer I			TrendSurfer II		
		Rank	Top %	Top-cluster %	Rank	Top %	Top-cluster %
PCA	1	1	Top 1	100.0%	1	Top 1	100.0%
	2	2	0.01%	100.0%	2	0.01%	100.0%
	3	1	Top 1	100.0%	1	Top 1	100.0%
kNN	1	1	Top 1	100.0%	1	Top 1	100.0%
	2	1	Top 1	100.0%	1	Top 1	100.0%
	3	126	0.19%	100.0%	126	0.19%	100.0%
CBLOF	1	1	Top 1	100.0%	1	Top 1	100.0%
	2	3	0.02%	100.0%	3	0.02%	100.0%
	3	1	Top 1	100.0%	1	Top 1	100.0%
Average			0.02%	100.0%		0.02%	100.0%

The test cases were built in the same way as for our effectiveness experiments for numeric dependent attributes. Table 2 reports the results; we can see that the outputs of both our algorithms fall into the top 0.02% range on average, and are within the top-scoring out- lier clusters in 100% of the cases. In summary, *TrendSurfer I* and *TrendSurfer II* appear promising with respect to both criteria of Section 2.4 when applied to categorical data.

5.3. User-Study Results for Problem 2

We now report the results of a user study that we conducted for the problem of user-driven retrieval of a single unusual temporal trend from the search space of temporal trends for a given fact table, see Section 2.4. In summary, our results suggest that users tend to be interested in unusual temporal trends more than in typical trends, and that using our *TrendSurfer II* procedure for recommendations to users has potential to improve user experience in user-driven retrieval of unusual temporal trends.

In the study we worked with a total of 12 users with diverse backgrounds. We first looked at the potential for **correlation between a trend's unusualness and its inter- estingness**, by inviting four of the users to rate the given temporal trends based on the degree to which the trends look interesting to them. We preselected ten trends from each of the *Flight Delay* and *House Pricing* data sets. In each set, the trends were subdivided into two groups: Group A (unusual trends) contained two top-scored ground-truth trends from depth 1 in the search space of trends for the data set, and Group B (typical trends) contained eight typical ground-truth trends from the same depth; we used *kNN* for the scoring. The trends within each Group B were similar to each other, and there was a clear difference in trend shapes across the groups for each data set. The users were invited to rate each of the ten trends for each data set, based on whether the trend looked interesting to the user. (The group memberships of the trends were not disclosed to the users.)

We postprocessed the outcomes both per user and across the users (using AVG in the latter case) for each data set, using the proportion of trends of interest to users that belong to Group A as precision, and the proportion of Group A trends marked as interesting by the users as recall. Fig. 6 shows the results, including the F1 scores over the averaged precision and recall values. With the F1 scores being much higher than 50% (79.3% for *Flight Delay* and 88.4% for *Housing Prices*), we see that the users considered Group A (unusual) trends more interesting than Group B (typical) trends. We view these results as a confirmation that locating unusual trends is a valuable part of the data-analysis process.

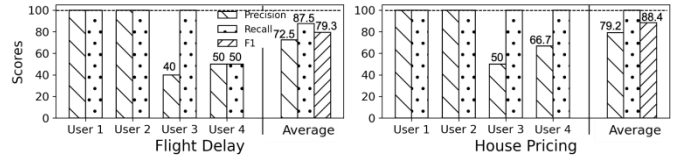

Fig. 6: Precision, recall, and F1-score results for the degree to which the unusualness of temporal trends correlates with their interestingness from the perspective of the users.

Observe that users 1, 2, and 3 all achieved 100% recall for both data sets in this study. This indicates that they were interested in all the unusual trends in the data sets, and thus reinforces the significance to users of discovering unusual trends in temporal data.

David C. Wyld et al. (Eds): MLNLP, NWCOM, DTMN, ASOFT, SIGPRO, AIFZ, CSITY, CLSB - 2023
pp. 103-123, 2023. CS & IT - CSCP 2023 DOI: 10.5121/csit.2023.131608

We then looked at the **capability of TrendSurfer II to reduce the length of user paths in search for unusual trends**. To that end, we divided the remaining eight users into four groups of two users each, and partitioned each of the *Flight Delay* and *House Pricing* data sets into two halves. The users' goal was to locate the top-scored ground-truth trend at at depth $k = 2$ for all the test cases. Each user was instructed to do an unassisted navigation in search for the goal trend in the first half of the data set assigned to them (stage 1). They then were to perform a navigation in the second half of the same data set with the same objective, this time starting from depth 1 after the *TrendSurfer II* procedure had selected for them the node at depth 1 from which to start the search (stage 2). There was no collaboration between the users within or across the groups.

Two of the user groups worked with the *Flight Delay* data set, with one group using the *k* NN outlier detector, and the other using *C* BLOF; the two remaining groups worked with the *House Pricing* data set, with the same outlier detectors allocated in the same way across the groups. Dividing each data set into halves was done to prevent the nontem- poral dimensions associated with the users' goal node in stage 1 of the study from being necessarily associated with the users' goal node in stage 2 of the study.

Table 3: Effectiveness of the *TrendSurfer II* procedure in reducing the length of search paths in user-driven retrieval of unusual temporal trends. The "assisted length" column lists those cases in which *TrendSurfer II* would choose for the user a node at depth 1 in the search space, with the user then choosing the goal node at depth 2 in the space.

Domain	Outlier detector	User-directed path length	*TrendSurfer II* assisted length	User rating of *TrendSurfer II*
Flight Delay	kNN	4	3	10
		7	2	5
	CBLOF	9	4	8.5
		2	2	5
	Average	**5.5**	**2.75**	**7.125**
House Pricing	kNN	2	2	10
		2	2	9
	CBLOF	2	2	7
		2	2	10
	Average	**2**	**2**	**9**

In each stage of the study, we measured the length of each user's path in the relevant search space of trends, using the definition of length of user-navigation paths introduced in Section 4. At the end of the study, we solicited the users' feedback on how well the *TrendSurfer II* suggestions had helped them in finding their target trends, in comparison with their unassisted search in stage 1 of the study. We also asked each user to provide a rating of the importance of the *TrendSurfer II* recommendations, ranging from 1 (not important at all) to 10 (critical to the success of the search).

The results are presented in Table 3. In the case of the *Flight Delay* data set, we can see that the *TrendSurfer II* suggestions have shortened the length of the user-navigation path by 50% on average, from 5.5 to 2.75. Recall from Section 4 that the optimal length of the user-perceived path when using our *TrendSurfer* procedures equals the number of nontemporal

dimensions in the goal node, which is 2 in the case of our test data sets. For the *Housing Pricing* data set, the users achieve this optimal path length with or without *TrendSurfer II* recommendations. This appears to be due to the users being more familiar with the house-pricing domain than with the airline-flights domain, which involves more complex and specialized expertise. Indeed, the users could have deduced from their life experience that the dimensions *City* and *Metro* would be more indicative of house prices than the *County* and *State* dimensions of the data set.

For both test data sets, the users provided high ratings of the *TrendSurfer II* recommendations on average. In those cases where individual users submitted lower ratings, we hypothesize that the users could have placed a high emphasis on the quality of the search paths that they had devised without assistance from *TrendSurfer II*.

In summary, we observe that our proposed *TrendSurfer II* procedure was rated highly by the users on average. The procedure did help significantly in the case of the *Flight Delay* data set, for which the users may not have had appropriate domain expertise and thus might have had trouble with their unassisted navigation of the search space. We conclude that using our proposed trend-surfing approach for recommendations to users has potential to improve user experience in user-driven retrieval of unusual temporal trends.

REFERENCES

[1] C. C. Aggarwal, Outlier Analysis. Springer International Publishing, 2nd ed. 2017 ed., 2017.
[2] A. Bl´azquez-Garc´ıa, A. Conde, U. Mori, and J. A. Lozano, "A review on outlier/anomaly detection intime series data," ACM Computing Surveys (CSUR), vol. 54, no. 3, pp. 1–33, 2021.
[3] M. Gupta, J. Gao, C. C. Aggarwal, and J. Han, "Outlier detection for temporal data: A survey," IEEETKDE, vol. 26, no. 9, pp. 2250–2267, 2013.
[4] CDC, "National Antimicrobial Resistance Monitoring System for Enteric Bacteria (NARMS): HumanIsolates Surveillance Report for 2015," tech. rep., Atlanta, Georgia: CDC, 2018.
[5] B. Borck Høg, F. Bager, H. B. Korsgaard, J. Ellis-Iversen, K. Pedersen, L. B. Jensen, et al., "DANMAP2017 - use of antimicrobial agents and occurrence of antimicrobial resistance in bacteria from foodanimals, food and humans in Denmark," tech. rep., 2018. https://danmap.org.
[6] J. Gray, S. Chaudhuri, and et al, "Data cube: A relational aggregation operator generalizing group-by,cross-tab, and sub-totals," Data Min. and Knowl. Discov., vol. 1, no. 1, pp. 29–53, 1997.
[7] C. S. Jensen, T. B. Pedersen, and C. Thomsen, "Multidimensional databases and data warehousing,"Synth. Lect. Data Manag., vol. 2, no. 1, pp. 1–111, 2010.
[8] P. Eichmann, F. Solleza, N. Tatbul, and S. Zdonik, "Visual exploration of time series anomalies withMetro-Viz," in SIGMOD, pp. 1901–1904, ACM, 2019.
[9] R. Neamtu, R. Ahsan, E. A. Rundensteiner, and G. N. S´ark¨ozy, "Interactive time series explorationpowered by the marriage of similarity distances," VLDB Endow., vol. 10, no. 3, pp. 169–180, 2016.
[10] M. Mannino and A. Abouzied, "Expressive time series querying with hand-drawn scale-free sketches,"in Proceedings of the CHI Conference on Human Factors in Computing Systems, p. 388, ACM, 2018.
[11] T. Siddiqui, P. Luh, Z. Wang, K. Karahalios, and A. G. Parameswaran, "ShapeSearch: A flexible andefficient system for shape-based exploration of trendlines," in SIGMOD, pp. 51–65, ACM, 2020.
[12] K. Zoumpatianos, S. Idreos, and T. Palpanas, "RINSE: Interactive data series exploration withADS+," VLDB Endow., vol. 8, no. 12, pp. 1912–1915, 2015.
[13] K. Zoumpatianos, S. Idreos, and T. Palpanas, "Indexing for interactive exploration of big data series,"in SIGMOD, pp. 1555–1566, ACM, 2014.
[14] F. Chirigati, H. Doraiswamy, T. Damoulas, and J. Freire, "Data Polygamy: The many-many relationships among urban spatio-temporal data sets," in SIGMOD, pp. 1011–1025, ACM, 2016.
[15] A. Bessa, J. Freire, T. Dasu, and D. Srivastava, "Effective discovery of meaningful outlier relationships," Trans. Data Sci., vol. 1, no. 2, pp. 12:1–12:33, 2020.

[16] M. Steiger, J. Bernard, S. Mittelst¨adt, and et al, “Visual analysis of time-series similarities for anomalydetection in sensor networks,” Comput. Graph. Forum, vol. 33, no. 3, pp. 401–410, 2014.
[17] M. Cavallo and C¸ . Demiralp, “Track Xplorer: A system for visual analysis of sensor-based motoractivity predictions,” Comput. Graph. Forum, vol. 37, no. 3, pp. 339–349, 2018.
[18] S. Idreos, O. Papaemmanouil, and S. Chaudhuri, “Overview of data exploration techniques,” in SIGMOD, pp. 277–281, ACM, 2015.
[19] M. Vartak, S. Huang, T. Siddiqui, S. Madden, and A. G. Parameswaran, “Towards visualizationrecommendation systems,” SIGMOD Rec., vol. 45, no. 4, pp. 34–39, 2016.
[20] M. Vartak, S. Rahman, and et al, “SeeDB: Efficient data-driven visualization recommendations tosupport visual analytics,” VLDB Endow., vol. 8, no. 13, pp. 2182–2193, 2015.
[21] H. Ehsan, M. A. Sharaf, and G. Demartini, “QuRVe: Query refinement for view recommendation invisual data exploration,” in ADBIS, vol. 1259, pp. 154–165, Springer, 2020.
[22] H. Ehsan, M. A. Sharaf, and P. K. Chrysanthis, “Efficient recommendation of aggregate data visualizations,” IEEE Trans. Knowl. Data Eng., vol. 30, no. 2, pp. 263–277, 2018.
[23] B. Tang, S. Han, M. L. Yiu, R. Ding, and D. Zhang, “Extracting top-k insights from multi-dimensionaldata,” in SIGMOD, pp. 1509–1524, ACM, 2017.
[24] R. Ding, S. Han, Y. Xu, H. Zhang, and D. Zhang, “QuickInsights: Quick and automatic discovery ofinsights from multi-dimensional data,” in SIGMOD, pp. 317–332, ACM, 2019.
[25] Y. Luo, X. Qin, N. Tang, and G. Li, “DeepEye: Towards automatic data visualization,” in ICDE,pp. 101–112, IEEE Computer Society, 2018.
[26] X. Zhang, X. Ge, P. K. Chrysanthis, and M. A. Sharaf, “ViewSeeker: An interactive view recommendation tool,” in EDBT/ICDT Joint Conference, vol. 2322, CEUR-WS.org, 2019.
[27] Y. Sismanis, A. Deligiannakis, N. Roussopoulos, and Y. Kotidis, “Dwarf: Shrinking the petacube,” inSIGMOD, pp. 464–475, 2002.
[28] R. T. Snodgrass and I. Ahn, “Temporal databases,” Computer, vol. 19, no. 9, pp. 35–42, 1986.
[29] M.-L. Shyu, S.-C. Chen, K. Sarinnapakorn, and L. Chang, “A novel anomaly detection scheme basedon principal component classifier,” tech. rep., Miami Univ. ECE Department, 2003.
[30] S. Ramaswamy, R. Rastogi, and K. Shim, “Efficient algorithms for mining outliers from large datasets,” SIGMOD Rec., vol. 29, p. 427–438, May 2000.
[31] Z. He, X. Xu, and S. Deng, “Discovering cluster-based local outliers,” Pattern Recognition Letters,vol. 24, no. 9, pp. 1641–1650, 2003.
[32] F. T. Liu, K. M. Ting, and Z. Zhou, “Isolation-based anomaly detection,” ACM Trans. Knowl. Discov.Data, vol. 6, no. 1, pp. 3:1–3:39, 2012.
[33] “Flight delay data set,” 2019. https://www.transtats.bts.gov/.
[34] “House pricing data set,” 2020. https://www.zillow.com/research/data/.
[35] “Car retailing data set,” 2020. https://data.world/data-society/used-cars-data/.
[36] Y. Zhao, Z. Nasrullah, and Z. Li, “PyOD: A python toolbox for scalable outlier detection,” Journalof Machine Learning Research, vol. 20, no. 96, pp. 1–7, 2019.

AUTHORS

J. Ao earned her Bachelor’s and Master’s degrees from the Beijing University of Technol- ogy in China, and her Ph.D. degree in Computer Science from NCSU in the United States. Her research interests include knowledge graphs, data management, and data science.

Kara Schatz earned a Master’s degree from NCSU and a Bachelor’s degree from Xavier University. Currently, she is pursuing her Ph.D. in Computer Science at NCSU.

Rada Chirkova received her Ph.D. from Stanford University, an is a Professor of Com- puter Science at NCSU. Her research lies in the area of data management.