DataJewel: Tightly Integrating Visualization with Temporal Data Mining

Mihael Ankerst, David H. Jones, Anne Kao, Changzhou Wang

Boeing Phantom Works,
P.O. Box 3707 MC 7L-70, Seattle, WA 98124-2207
[mihael.ankerst | david.h.jones | anne.kao
| changzhou.wang]@boeing.com

Abstract. In this paper we describe DataJewel, a new architecture designed for temporal data mining. It tightly integrates a visualization component, an algorithmic component and a database component. We introduce a new visualization technique called CalendarView as an implementation of the visualization component. We show how algorithms can be tightly integrated with the visualization component and that most existing temporal data mining algorithms can be leveraged by embedding them into our architecture. This integration is achieved by an interface that is used by the user and the algorithm to assign colors to events. The user assigns colors to interactively incorporate domain knowledge or to formulate hypotheses. The algorithm assigns colors based on the discovered patterns. Using the same visualization technique for both data and patterns makes it more intuitive for the user to select useful patterns from those returned by the algorithm. We also present a data structure that supports temporal mining of very large databases. In the experiments, we apply our approach to several large datasets from the airplane maintenance domain and discuss its applicability to domains like homeland security, market basket analysis and web mining.

1 Introduction

In recent years, there has been a lot of interest in the KDD community in mining temporal data. Temporal datasets have a dedicated attribute storing a time stamp for each record. This time stamp usually refers to the date an event has happened or some kind of data has been measured and collected. Examples for temporal datasets include stock market data, manufacturing or production data, maintenance data, web mining and point-of-sale records. Due to the importance and complexity of the time attribute, a lot of different kind of patterns are of interest. An overview is provided in [2]. Typically, in different domains different kind of temporal patterns are of interest. This is one aspect motivating our architecture, which provides access to many temporal data mining algorithms and an easy way to add new ones.

When dealing with temporal databases a second but very substantial aspect becomes an important challenge. In large enterprises, databases evolve as a consequence of an organizational need. They are designed to serve a specific (e.g. operational) purpose. Often databases from different organizations can be linked together to serve a new purpose, e.g. to provide a platform for data mining. However, the task of linking databases together is far from trivial; the field of information integration deals with challenging and laborious problems of maintaining data integrity, schema mapping, and resolving duplication. Often, there is no common attribute at all except the timestamp. By linking tables together to explore a subset of the union of the attributes with respect to time, a powerful view upon the data is obtained. E.g. intelligence agencies can link tables together that correspond to news, credit card histories, travel itineraries to detect suspicious activities. An enterprise can link together helpdesk data about computer problems with a completely independent table from the procurement department and a labor database. The detected patterns might reveal insights into causes of computer problems and might form a new purchasing strategy.

In this paper, we address both aspects of temporal data mining. On the one hand, our approach is applicable to a variety of domains because it leverages existing algorithms. On the other hand, it offers a means of linking tables together that have no primary key – foreign key relationship. All they are required to have is an attribute with a timestamp.

In addition, our new architecture for temporal data mining also makes the following contribution. Traditionally, algorithmic approaches are introduced by one research community and some of them also focus on scalability aspects. However, for most of the papers, the visualization component is omitted, either because the authors do not feel comfortable in this area or because they think a graphical user interface alone should be sufficient and that is not a research task. On the other hand, the visualization community focuses most often just on the visualization aspect, i.e. how to represent data, but does not investigate algorithmic approaches [5]. With this paper, we would like to make a contribution towards a closer collaboration of these fields. We show that a system that is designed to tightly integrate components from various disciplines can substantially improve the functionality of loosely coupled components.

The rest of the paper is organized as follows. In Section 2, we summarize related work. Section 3 describes a user-centric data mining process and the DataJewel architecture. Section 4 presents the visual component of our architecture and describes in detail our new visualization technique called CalendarView. Section 5 outlines how temporal data mining algorithms can be tightly integrated into DataJewel. Section 6 reports how to handle large datasets. In section 7, we describe several experiments with large datasets. We conclude the paper with section 8 and discuss some future directions.

2 Related Work

Our main contribution is to tightly integrate a visual, an algorithmic and a database component for temporal data mining. To our knowledge no such architecture has been proposed so far. Most of the work in temporal data mining deals with either just an algorithmic approach, a way to visualize data over time or an approach to scale up to large datasets. We review these areas in the following paragraphs.

Many approaches for visualizing data over time have been proposed. Typically, visualization techniques represent temporal data either as a sequence along an axis, or as animations where data at different times is represented in different frames. A recent approach which treats data as a sequence is ThemeRiver [6]; it employs the metaphor of a current and maps histograms of document keywords to the height of a wave at a particular time. Mackinlay et. al. [11] uses a spiral for calendar visualization, however, calendar days are just used as reference points. Hierarchical pixel bar charts [8] are not aimed at visualizing temporal data but it can be used as an alternative pixel representation within a day.

Several algorithms for mining temporal datasets have been proposed. According to a recent overview [2], contributions have been made in the areas of how to model a temporal sequence, how to define a suitable similarity measure for sequences and what kind of mining operations can be performed. We will show in section 5 that many of the existing algorithms can be leveraged by our architecture.

Tightly integrated architectures have been proposed, but are only partially comparable to our approach. In [1], the authors describe an approach called cooperative classification, where the visualization and the algorithmic component are tightly integrated. This approach however, was specifically designed for decision tree classification and does not elaborate on scalability issues. Similarly, HD-eye [8] and n23Tool [15] integrate visualization with algorithms but are applicable just to clustering methods. [14] represents clusters of time series data which contain a pattern spanning one day and relating them to days with similar patterns. In contrast to our approach, it does not represent the data for each day nor does it cover scalability issues. Tightly integrating algorithms with databases or incorporating scalability considerations into data mining algorithms has been recognized and studied more extensively. A comprehensive survey is presented in [10]. Proposed ways to achieve scalability are falling in one of the three categories: design of a fast algorithm (e.g. by restricting the model space or parallelization), partitioning of the data (instance/feature selection methods) and relational representations (e.g. integration of data mining functionality in database systems). Recent approaches include the computation of sufficient statistics, like Rainforest [4] does for decision trees. [12] describes an in-depth analysis of different level of integration of an association mining algorithm into database systems. CONTROL [7] aims at a database-centric interactive analysis of large datasets focusing on online query processing. All these approaches, however, are not directly applicable to temporal data.

3 User-centric Data Mining

One design goal of our user-centric architecture is its intuitive use by a domain expert as opposed to data mining experts. As a result, the user can steer the exploration of temporal data, invoke algorithms to automatically discover patterns, incorporate his domain knowledge, hypothesize on the fly and use his perception to detect patterns of interest. In figure 1, we outline the mining process with DataJewel.

First, the user selects the tables and attributes for analysis. Then the data is loaded and

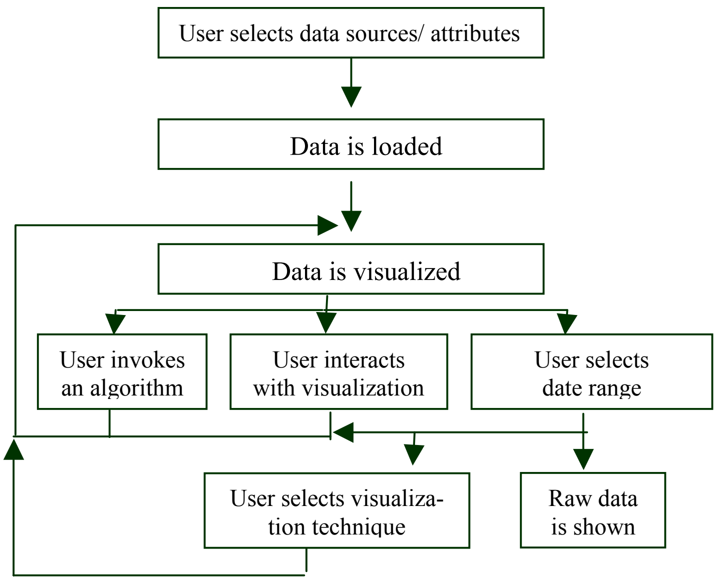

User selects data sources/ attributes
Data is loaded
Data is visualized
User invokes an algorithm
User interacts with visualization
User selects date range
User selects visualiza-tion technique
Raw data is shown

Figure 1. The mining process with DataJewel

visualized. The user has the option of invoking an algorithm and visualizing the resulting patterns using the current settings. Alternatively, the user can interact with the visualization to incorporate his domain knowledge or discover some patterns based on his perception. In either case, the user hopefully discovers some pattern of interest. Then he selects a date range of interest and visualizes it with the same or another visualization technique. Another visualization technique might be picked to represent the data in a different way or because it is more suitable due to the reduced size after the selection. After the user has iterated this loop several times, he might be interested in "drilling down" to the raw data to see all attributes. The corresponding tables are accessed, the data is retrieved and presented. Note that this approach facilitates extensions by incorporating new algorithms and visualizations.

In the following, we introduce some terminology and state assumptions for our architecture. Let us assume, the data sources consist of a set of tables. Each table contains r records, with each record consisting of d attributes $a_1,\ldots,a_d$. At least one attribute contains a timestamp for each record. We refer to the timestamp attribute as the *event date*, all categorical[1] attributes that should be incorporated in the analysis are *event attributes* and the attribute values of these event attributes are *events*. In this paper, we will focus on event attributes (categorical attributes only) for which the following assumptions hold:

a) The number of event attributes is low. (< 10)

b) The number of different events of one event attribute is moderate. (< 200)

c) The smallest time unit of interest in the event dates is one day

Assumption a) restricts the number of event attributes used during the analysis. As opposed to high-dimensional feature vectors for which some mining tasks are performed, event attributes usually have a clear meaning. Often, in one given analysis, the analyst selects a small number of event attributes, which can be associated with each other in the particular domain. Using domain knowledge, the remaining attributes are omitted from the analysis because they would just add noise.

Assumption b) limits the number of events of an event attribute to a moderate size. In case, where this is not true for the initial dataset, a concept hierarchy can be defined for the event attribute to reduce the total number of events.

With assumption c) we focus on the most common time unit of interest in business domains. Note that days are just the smallest unit of interest and the discovery of weekly or monthly patterns is also supported. Obviously, for intrusion detection systems, our proposed unit of time would have to be refined to reflect finer grained time units.

In figure 2, a simplified view of the DataJewel architecture is depicted consisting of three layer. Although the data flows from the data source to the visualization layer, we have designed our system from the opposite direction to better support a user-centric process. Corresponding to each one of these layers, we will describe the visualization, the algorithmic and the database component. We will present just one instance of the visualization and the algorithmic component, but new ones can be easily integrated.

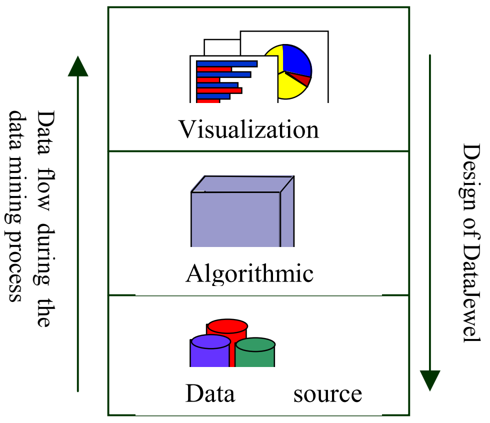


Figure 2. Data flow versus design of DataJewel

[1] Continuous attributes can be transformed into categorical attributes by discretization.

4 The Visualization Component

The visualization component contains visualization techniques suitable for representing temporal data. We present a new visualization technique, which represents temporal data on a daily basis.

4.1 CalendarView

Our architecture is primarily designed for domain experts not just for data mining experts. Thus the visualization component has to be intuitive as well as versatile. CalendarView, our new visualization technique, is motivated by what the human is already very familiar with. First, the representation of event dates is designed following the visual metaphor of a calendar. Second, the structure of the data that is represented along the event dates is the frequency of events. Its representation is based on the familiarity of humans with histograms.

In simpler linear representations, time is greatly simplified by modeling it as a sequence of dates. In contrast, we have selected the calendar metaphor because it reflects the rich temporal structure more effectively than typical simplified representations. From a calendar, the human preattentively extracts the notion of weekends, weekly repetitions, seasons, days with a special meaning in his domain, etc.

Whereas the calendar metaphor is used to represent the event dates on a daily basis, an extended version of histograms reflects the distribution of events for a single day. To enable the user to compare different event attributes with each other, each event attribute is represented by a separate calendar. In the final visualization all calendars are drawn one above the other.

Table 1. Example of a temporal dataset

Event Date	Event Attribute: Page hit	Event Attribute: Browser	Event Attribute: …
1/1/2002	Index.html	MS IE	…
1/1/2002	Dep1/contacts.htm	Netscape	…
…	…	…	…

Table 1 depicts an example of a temporal dataset. Each event has an associated event date, so we can count the frequency of this event occurring on a single day. If we do that for each event of one event attribute we can display the distribution by a histogram for this event attribute. We can initially assign a different color to each event of one event attribute. The default color map is the PBC color map [1] which has been developed to map distinct values to distinct colors. As illustrated[2] in figure 3, for each day the frequency distribution of the events is represented within the corresponding day in the calendar.

[2] Note that the color mapping in this paper is not the original color assignment. It has been changed to optimize for grayscale printing.

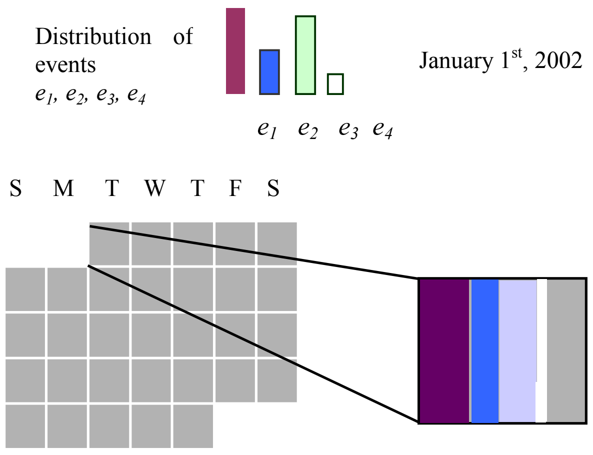


Figure 3. Illustration of CalendarView

Instead of using colored histograms where frequency is depicted by the height of the bins, the events are represented pixel by pixel to account for more categories than are usually depicted by a histogram. In particular each day is filled with pixels in the following way:

Each day is represented by a constant size square of $n \times n$ pixels. If the number of events of this event attribute on the corresponding day is less or equal to n^2, we can use one pixel per event. The pixel arrangement starts in the lower left corner of the day square. It goes up (*n-1*) times, goes one pixel to the right, then goes (*n-1*) pixel down, one pixel to the right, and so forth. Following the illustration in figure 3, let us assume we have four events e_1, e_2, e_3 and e_4. The frequency of the occurrence of e_1 on a particular day is denoted by $f(e_1, date)$. Then we draw the first $f(e_1,date)$ pixels with the color assigned to e_1. Following the described pixel arrangement, the next $f(e_2, date)$ pixels are drawn in the color of event e_2, and so forth. We can distinguish between the following three cases:

1) If $n^2 = \sum_e f(e,date)$ then we fill up the complete day square and each pixel represents one event by its color.

2) If $n^2 > \sum_e f(e,date)$ then each pixel represents one event by its color but all pixels do not fill up the entire space in the day square. The remaining pixels are drawn with a separate (background) color.

3) If $n^2 < \sum_e f(e,date)$ we fill up the complete day square by the algorithm above after substituting f(e, date) with $\left\lceil \frac{f(e,date)}{\sum_e f(e,date)} \cdot n^2 \right\rceil$. (formula 1)

The order of the events can greatly contribute to perception of their distribution. By reordering the events for each day preattentive processing can be improved. The rea-

son becomes clear with the following example. Let us assume the daily distribution of ten events over one year is very similar and even within each day the number of events does not differ largely. Let us further assume there is exactly one day where the least frequent event suddenly happens more often. At that day it is the second most frequent event. Then in addition to representing this event with more pixels than at other days, the reordering yields a better perception of this distribution change. Thus, the reordering improves the perception of distribution changes (cf. figure 4 where the 4th day is reordered). Note that the daily reordering is done in real time, since computation is negligible, due to our assumptions in section 3.
Our default setting which we use throughout the paper is *n = 10*. The size of the day square n^2 is a tradeoff between representing each event by one pixel and the size of the (virtual) screen.

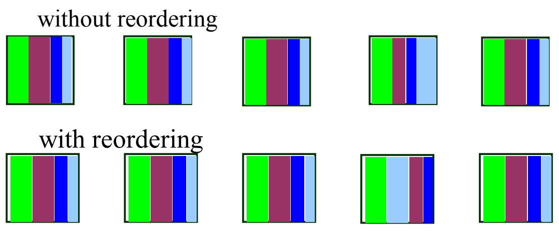


Figure 4. Ordering events daily by frequency

4.2 Interaction with CalendarView

In the following, we will describe the main interaction capabilities of CalendarView:

- Selection

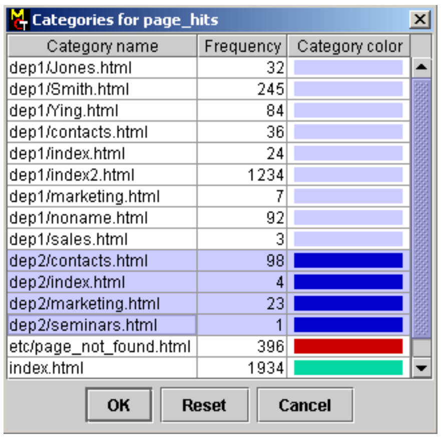
As described in section 3, one essential feature of the visualization component is to select a subset of dates. The user is enabled to interactively select a set of consecutive days. The subset corresponding to the selected event dates can again be visualized following the iterative process outlined in section 3.

- Ascending/descending order

The decision if the events should be ordered ascending or descending by frequency is just important for the case where we have less pixels in a day square than events. If the frequency distribution on a particular day is very skewed, some events might not be represented at all because the drawing algorithm with formula 1 might have already filled up the complete day square. We think, in most cases the user is either interested in outlier events which happen very rarely as opposed to others or he is interested in the overall distribution of the "main" events. Therefore we enable the user to switch between ascending or descending order in real time. In case the ascending order has been selected, the drawing of the pixels starts with the rarest events and thus uncovers them possible at the expense of cutting off the largest event at the end. If the descending order is selected, the most frequent events are drawn first.

- Interactive color assignment

Initially, colors are assigned to events based on the PBC color map. Thus missing values can be elegantly treated as a distinct event and are assigned to a certain color (background by default). A dialog window enables the user to interactively assign colors to events. With manual color assignment the user can specify his domain knowledge, can formulate and test an hypothesis on-the-fly or steer the exploration in a meaningful way. The notion of color assignment is implemented as follows: If the user changes several events to have the same color, he indicates a conceptual generalization of the events. As a result, all events which are assigned to the same color are referred to as the same event when the visualization is redrawn. Thus, for each day, the events with the same color are grouped together before the drawing algorithm is invoked. For example, following the web mining dataset from table 1, let us assume we are recording web page hits on a particular day. These events can be generalized by the user by assigning color c_1 to all pages of the main website which have been visited. Color c_2 refers to the dep1/ subdirectory, c_3 to the dep2/ subdirectory and color c_4 is used for all other web pages. The interface is depicted in figure 5, also enabling the user to sort by event name or frequency. Each event attribute has a separate color assignment.



Category name	Frequency	Category color
dep1/Jones.html	32	🟪
dep1/Smith.html	245	🟪
dep1/Ying.html	84	🟪
dep1/contacts.html	36	🟪
dep1/index.html	24	🟪
dep1/index2.html	1234	🟪
dep1/marketing.html	7	🟪
dep1/noname.html	92	🟪
dep1/sales.html	3	🟪
dep2/contacts.html	98	🟦
dep2/index.html	4	🟦
dep2/marketing.html	23	🟦
dep2/seminars.html	1	🟦
etc/page_not_found.html	396	🟥
index.html	1934	🟩

Figure 5.
Interactive color assignment

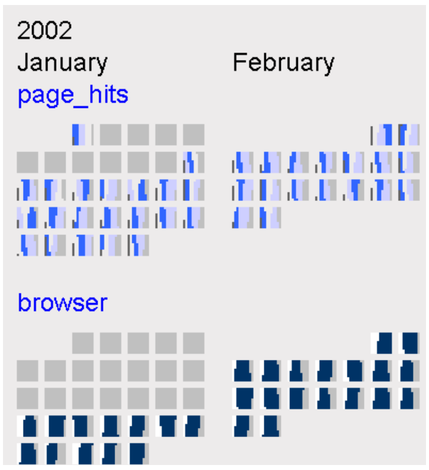


Figure 6.
Calendarview with web mining dataset

In figure 6, two event attributes from our example dataset are visualized from January 1st, 2002 to February 18th, 2002. For the "page hits" event attribute, the user has assigned colors to four different groups of web pages as described above. We see page hits on January 1st but no more until Saturday, January 12th. Maybe the web server was down for 11 days? Also the event attribute "browser" has its first event on Sunday, January 20th. Maybe the web server did not recognize the browser type until that day? Just two different browsers have been recognized. One browser has being used more often throughout the whole time period.

-Zooming

The user can zoom in or zoom out.

- Detail on Demand
The event corresponding to the pixel of the current mouse pointer position is displayed.

5 The Temporal Mining Component

Building the visualization component, we have introduced a visualization technique called CalendarView, which maps different events to distinct colors. In the case there are just a few events the visualization itself is very powerful since human's preattentive perception is very efficient in looking for variety of patterns. If the number of different events is larger, the usefulness of the default color assignment decreases because colors are not perceived as being distinct any more.

Nevertheless the visualization technique might reveal patterns since changes in the event distribution might still be perceived. With the interface for interactive color assignment, we have introduced one concept for handling a larger number of events. However, if focus is not on a certain known event, manually changing a random sequence of colors can quickly become tedious. This realization has led us to consider a tight integration of the temporal mining algorithms to the visualization. Coming from this perspective, we would like to have algorithms that discover patterns, determine the events involved in the patterns and use this information to automatically select colors based on the patterns that will be revealed. This automatic color selection can be used to compute a reasonable default color assignment or it can be invoked at any time during the exploration.

In summary, two aspects of our architecture contribute to the intuitive cooperative exploration of the data by the user and the algorithms. First, CalendarView visualizes not just the data but also the patterns. Second, the same color assignment interface is used by both the user and the algorithm. We now will focus on how the following three classes of algorithms use color assignment:

- Discover one single event of one event attribute that shows an interesting pattern
- Discover multiple events of one event attribute that show an interesting pattern
- Discover one event for each event attribute such that these events together show an interesting pattern (an extension is that the user selects one event and lets the algorithm detects events of other event attributes which show some relation to the selected event, e.g. similarity, correlation, etc.)

Discover one event of one event attribute
Many existing algorithms calculate one single event based on some measure of interest [2]. These measures can range from basic statistical methods like highest variance to more computationally expensive ones like "most interesting trend". No matter how the algorithms compute the single event of interest, our approach encapsulates it and changes the colors of the events accordingly. This means all colors but one are changed to one light color, whereas the event for which the pattern was found is

changed to have a unique dark color. Thus the user can focus on the distribution of this single event in relation to the overall frequency of all events.

We have included the following implementation of such an algorithm called *LongestStreak,* which is based upon the idea of stabilized p charts from the statistical field control charting [13]:

1. For each event *e*, compute a sequence of relative frequencies as follows: For each day, compute the percentage of occurrences of event *e* based on all events occurring on the same day.
2. Compute the weighted mean and standard deviation of each sequence. Consider just the days that are event dates.
3. Label each day where event *e* is significantly below or above its mean as *significant day* with respect to event *e*.
4. Return the event with the longest streak of consecutive significant days. Break ties by returning the first one found.

Alternatively, we could also modify step 4 to return the event with the most significant days. After the visualization is updated based on the discovered event the user can continue the exploration process.

Discover multiple events of one event attribute

Again, many algorithms have been proposed which compute this class of patterns [2], e.g. discovery of similar events. The algorithm returns a set of events which together represent a pattern. Our architecture changes the color assignment such that each event that is part of the pattern is assigned a distinct color, and all other events are assigned to one color.

Our implemented instance of this class of algorithms called *MatchingEvents* extends *LongestStreak* described above:

1. For each event, compute significant days and record a bit sequence having a ‘1’ for each a significant day and a ‘0’ otherwise
2. Take *LongestStreak* as the baseline event
3. Compare the bit sequence of the *LongestStreak* event with all others to find the closest match. This is determined by a bit-wise comparison and each match of a ‘1’ in both sequences increments the match counter by one. The event whose bit sequence has the highest match counter is the *correlated event.*
4. Return the LongestStreak event and the correlated event.

Discover one event for each event attribute

The two previous algorithms have looked for patterns in one single event attribute. In contrast, this class of algorithms looks for patterns relating event attributes to each other, instead of analyzing them separately. Many proposed algorithms fall into this class, e.g. finding similar events across different event attributes. The resulting pattern is visualized by updating the color assignments of each event attribute accordingly.

We implemented an instance of this class very similar to *MatchingEvents*. But instead of comparing the *LongestStreak* of the first event attribute to other events of the same attribute, it is compared to all events of the other event attributes. The algorithm returns the *LongestStreak* of the first event attribute and for each other event attribute the event that is correlated. In the experimental section, we refer to this algorithm as *MatchingEvents2*.

6 The Database Component

In this paper, we assume the datasets reside in tables from one or more relational databases. The integration of a database component should provide access to the data, a mechanism to scale up to large datasets and the capability to access the raw data of all attributes associated with the patterns found.

The critical part of the database component is to scale up to large databases. For our architecture, scalability entails a visualization and a memory aspect. The first aspect, namely how to visualize large datasets is addressed by visualizing the relative frequency of events on a single day as described in section 4. In this section, we describe how large datasets are processed. The fundamental idea is to compute an aggregated version of the dataset such that it fits in main memory. The aggregated dataset contains sufficient statistics similar to e.g. [4] for decision trees, and we show the upper bound of the main memory requirements based on our assumptions stated in section 3.

Let us pick up our example dataset from table 1. This dataset might consist of millions of rows since each occurrence of an event is typically stored as one record. If we use the aggregation capabilities of the database, the number of records that are loaded can be significantly reduced. Instead of storing each occurrence of an event, we count for each day the number of occurrences for each event. E.g. the sufficient statistics for event attribute “page hits” can be computed by submitting the following SQL query:

```
SELECT Event_date, page_hits, count(*) as Frequency
FROM example_table
GROUP BY Event_date, page_hits
ORDER BY Event_date, page_hits;
```

The resulting table is sketched in table 2. The amount of compression achieved by aggregation depends on the number of distinct event dates, the number of distinct events and how distinct events are distributed across the dates.

Table 2. Sufficient statistics for event attribute "page_hits"

Event date	Event attribute (page_hits)	Frequency
1/1/2002	Index.html	1934
1/1/2002	Dep1/contacts.html	36
…	…	…

The memory requirement for our initial dataset is proportional to the number of entries in a relational table. For one event attribute, event dates and events of this attribute have the memory requirements mem_{init}, with

$$mem_{init} \propto \textit{number of days} \cdot \textit{average number of events per day}$$

In contrast, the memory requirements mem_{new} for the computed sufficient statistics table (table 2) is

$mem_{new} \propto number\ of\ days \cdot average\ of\ the\ number\ of\ distinct\ events\ per\ day$

The difference in memory usage is the ratio between the average number of events per day and the average number of distinct events per day. This ratio will vary with the domain and the event attribute. For example, in the aircraft maintenance domain for one airline we had:

Average number of events per day: 402

Average number of distinct events per day: 32

The ratio in this example is 12.5:1. Whereas the number of records grows linearly for the initial dataset with every new event, our new table typically just increments a counter. This is most useful in domains where the number of events per day is very high, like web page accesses, items in market baskets across departments, phone calls, etc.

Given our assumptions from section 3, the worst case memory requirements mem_{worst} for the sufficient statistics table of one event attribute can be computed for e.g. 15 years:

$mem_{worst} \propto 15 \cdot 365\ days \cdot 200\ distinct\ events = 1{,}095{,}000$

In this case, every event happens every day at least once during a period of fifteen years. We can store each event with one byte (next to a small lookup table) and the days and frequency as integers with 4 bytes. The sufficient statistics table would require: 1,095,000 $\cdot$ (1 + 4 + 4) = about 9.8 Megabytes. Together with our assumption that the number of event attributes is low, we can conclude the sufficient statistics tables fit in main memory for many domains.

To summarize, the database component is integrated in two ways: First, the relevant event attributes of the original tables are compressed by computing the summary statistics offline. Second, database access is provided in a straight forward way: Since the user basically selects subsets of the initial time period during the exploration process, he can decide to retrieve the records with all attributes corresponding to the selected time period. Then a range query over the time period returns the raw data of interest. In our experiments, the computed summary statistics always fit in main memory and the computation of the proposed algorithms is efficient. Both, we believe is true for most datasets which fulfill our assumptions in section 3.

However, if more attributes are involved in an algorithmic run, or the integrated algorithms are more complex, then a tighter integration with the database component might be necessary. E.g. algorithms might be decomposed and leveraged by SQL extensions or user-defined functions could be used. If the algorithmic run is pushed back to the database, the user can continue to explore the data and get notified after the computation is finished.

7 Experiments

In our experiments, we investigate several real-world datasets from the airplane maintenance domain. We think the scenario we describe in this section is similarly applicable to many other domains like homeland security, web mining, market basket analysis or intrusion detection. The datasets are tables from a database containing maintenance events of different airlines for different airplane models[3]. Maintenance events range from negligible ones like coffee spills on the seat to major ones like problems with a landing gear. Each record has information about the date a maintenance problem has occurred, the airport where it was recorded, who discovered it, the written complaint, the maintenance action taken, the system and subsystems affected by the problem, etc… We will focus on the affected systems, which will be our event attribute. A system is a set of related parts that work together to perform a function such as communication, engine, flight control, doors, etc.

Table 3. Datasets

Dataset	Event dates	Nr. of events	Nr of records (originally)	Nr. of records (suff stat)
A	3/6/89-12/31/02	37	350,772	87,030
B	5/12/90-12/31/02	39	1,165,881	117,441
C	1/30/89-12/31/02	41	1,405,582	133,116
D	3/6/89-12/31/02	28	350,772	78,802
E	11/12/89 - 12/31/02	41	2,051,269	162,918
F	1/12/89-12/31/02	182	2,051,269	574,071
G	12/27/89 - 12/31/02	40	17,499	11,547

Table 4. Runtime (in seconds) of algorithms

Dataset	LongestStreak	MatchingEvents	MatchingEvents2
A	0.27	0.31	0.53 (with B)
B	0.31	0.30	0.62 (with C)
C	0.35	0.36	0.54 (with D)
D	0.28	0.26	0.63 (with E)
E	0.37	0.36	0.9 (with F)
F	0.71	0.68	0.87 (with G)
G	0.23	0.22	0.47 (with A)

[3] An airplane model is e.g. 747, 767, etc.

Metadata about the various datasets we explored is depicted in table 3. The recorded maintenance datasets span time periods between twelve and fourteen years. Table 4 shows the runtime of our implemented algorithms on the datasets. For the algorithm MatchingEvents2, we also indicate in brackets which other dataset has been the second event attribute. We ran all experiments on a PC with a Pentium III/ 800 Mhz processor and 1 GB main memory. For all datasets, we achieve an acceptable runtime.

7.1 Mining Airplane Maintenance Datasets

We describe a typical scenario which shows how our approach can be used. We start our investigation by selecting a dataset from one airline and one model. The chosen event attribute which we analyze over time is the system of the airplanes. Since there are a lot of different systems, we select the algorithm LongestStreak to compute one interesting system (it found engine fuel) which updates the color assignment. Figure 7 top row shows a small range of the resulting visualization. Especially during the last five days of July 2000, we perceive many events, indicating problems with engine fuel. Next, we add several datasets to compare this finding with patterns for different airlines. For each airline and the same model, we manually change the color assignment of the systems. We color every system except engine fuel with one light color and assign a dark color to all engine fuel related events. When we compare these airlines (two more airlines are shown in figure 7), we see the other airlines do not show a specific pattern. Even though just a small time range is shown, it is the case for all event dates. So we might decide to further investigate the first airline. Now we add to the first dataset another dataset which aggregates individual airplane id's of the same airline and model over time. The event attribute of the newly added dataset is the airplane id and we would like to find a correlation between the events we identified concerning engine fuel and maintenance events of individual airplanes. We run the algorithm *MatchingEvents2* to single out one airplane. This airplane is shown in figure 8 and we see e.g. that a lot of maintenance events for this single airplane have occurred on December 3rd, 1997. Note that for brevity we have omitted a screenshot of the corresponding time range of figure 7.

Finally, we select a dataset with maintenance events of just this airplane. The event attribute is again airplane systems. We run the algorithm *MatchingEvents* to see if two events frequently co-occur. A part of the resulting visualization is shown in figure 9. The two correlated events returned are fuel and communications indicated by the black and light gray color. E.g. on Monday 18th November, both events co-occur. With this knowledge we drill down to the raw data to further investigate the findings.

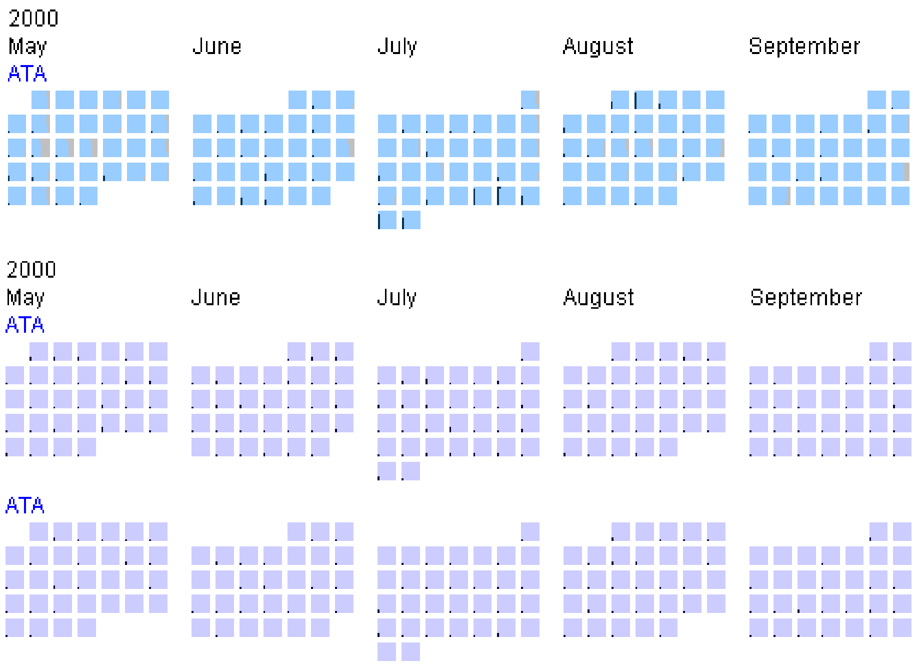


Figure 7. Focusing on maintenance events with the same subsystem for three different airlines

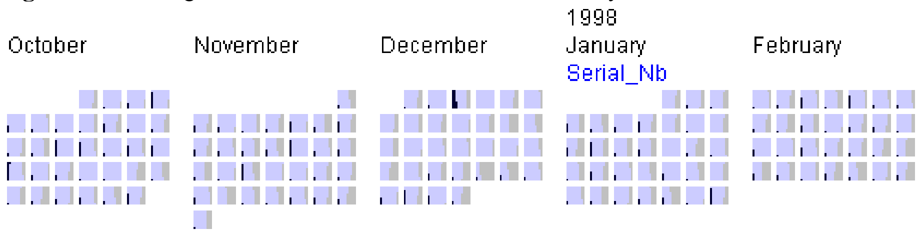


Figure 8. CalendarView focusing on maintenance events for one airplane

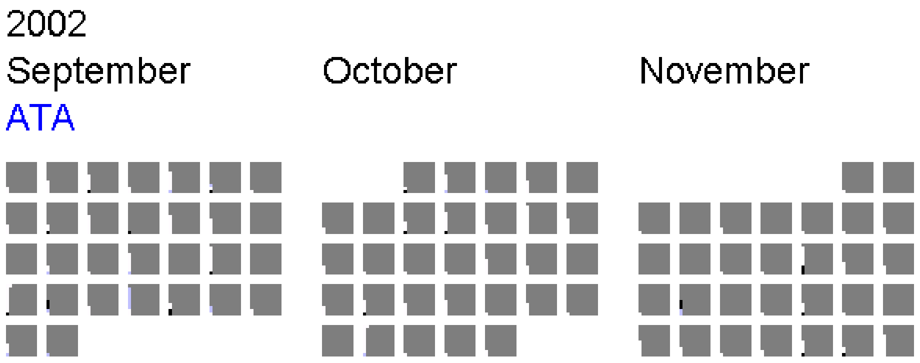
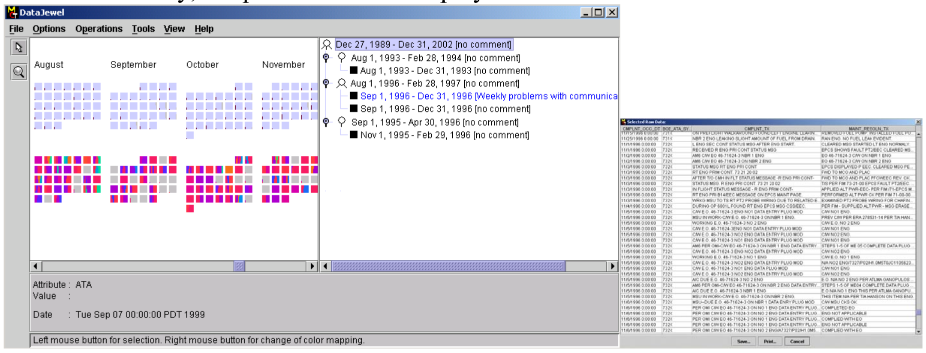


Figure 9. CalendarView focusing on maintenance events with two subsystems for one airplane

7.2 The DataJewel System

We have implemented the DataJewel system based on the architecture proposed in this paper. It can quickly be adapted to new domains since it is designed to be extensible for new visualization techniques and new algorithms. In the figure 10 two useful features are shown. First, the raw data can be accessed, displayed, saved or printed. Whereas the temporal analysis is based on just a few event attributes from possibly different tables, the user typically is interested in other attributes of the records corresponding the pattern found. As the data has been distilled and narrowed down during the exploration, the current range of event dates represents the dataset of interest. Thus just one range query is submitted against the database(s) to retrieve the attributes of interest. Second, an optional tree on the right side depicts the exploration process. The simplified temporal mining process presented in section 3, focuses on the iterative process of reducing the dataset. However, at some points the user may like to return to a previous stage, either because he found something of interest or not. Therefore, the tree on the right side shows a node for each subset of data explored so far. The user can either return to a node or annotate a node.

The algorithmic component can be used in three ways. It can be used to determine the default color mapping, it can be invoked at any time during the exploration process or it can run as a background process in parallel to the user's exploration and notify him upon discovery of some patterns. In addition to updating the color assignment after patterns have been found, the event dates not covering the patterns can be grayed out. Alternatively, the patterns can be displayed in a textual form.



Dec 27, 1989 - Dec 31, 2002 [no comment]
- Aug 1, 1993 - Feb 28, 1994 [no comment]
  - Aug 1, 1993 - Dec 31, 1993 [no comment]
- Aug 1, 1996 - Feb 28, 1997 [no comment]
  - Sep 1, 1996 - Dec 31, 1996 [Weekly problems with communica
  - Sep 1, 1996 - Dec 31, 1996 [no comment]
- Sep 1, 1995 - Apr 30, 1996 [no comment]
  - Nov 1, 1995 - Feb 29, 1996 [no comment]

Attribute : ATA
Value :

Date : Tue Sep 07 00:00:00 PDT 1999

Left mouse button for selection. Right mouse button for change of color mapping.

Selected Raw Data:

CMPLNT_OCC_DT	BOE_ATA_SY	CMPLNT_TX	MAINT_RESOLN_TX
11/15/1996 0:00:00	731<	ON PREFLIGHT WALKAROUND FOUND LEFT ENGINE LEAKIN...	REMOVED FUEL PUMP. INSTALLED FUEL PU...
11/25/1996 0:00:00	731<	NBR 2 ENG LEAKING SLIGHT AMOUNT OF FUEL FROM DRAIN.	RAN ENG. NO FUEL LEAK EVIDENT.
11/1/1996 0:00:00	732<	L ENG SEC CONT STATUS MSG AFTER ENG START.	CLEARED MSG STARTED LT ENG NORMALY.
11/1/1996 0:00:00	732<	RECEIVED R ENG PRI CONT STATUS MSG	EPCS SHOWS FAULT PT2/EEC CLEARED MS...
11/2/1996 0:00:00	732<	AM6 C/W EO 46-71624-3 NBR 1 ENG	EO 46-71624-3 C/W ON NBR 1 ENG
11/2/1996 0:00:00	732<	AM6 C/W EO 46-71624-3 ON NBR 2 ENG	EO 46-71624-3 C/W ON NBR 2 ENG
11/3/1996 0:00:00	732<	STATUS MSG RT ENG PRI CONT	EPCS DISPLAYED P EEC. CLEARED MSG PE...
11/3/1996 0:00:00	732<	RT ENG PRIM CONT. 73 21 20 02	FWD TO MCO AND PLAC
11/3/1996 0:00:00	732<	AFTER T/O CMH IN FLT STATUS MESSAGE -R ENG PRI CONT-	FWD TO MCO AND PLAC PFCWEEC REV. CK...
11/3/1996 0:00:00	732<	STATUS MSG. R ENG PRI CONT. 73 21 20 02	T/S PER FIM 73-21-00 EPCS FAULT PT2/EEC...
11/3/1996 0:00:00	732<	IN FLIGHT STATUS MESSAGE - R ENG PRIM CONT-	APPLIED ALT PWR-EEC- PER FIM /71-EPCS M...
11/3/1996 0:00:00	732<	RT ENG PRI B14/EEC MESSAGE ON EPCS MAINT PAGE.	PERFORMED ALT PWR CK PER FIM 71-00-00...
11/3/1996 0:00:00	732<	WRKG MSU TO TS RT PT2 PROBE WIRING DUE TO RELATED E...	EXAMINED PT2 PROBE WIRING FOR CHAFIN...
11/4/1996 0:00:00	732<	DURING OP 6801L FOUND RT ENG EPCS MSG CSS/EEC.	PER FIM - SUPPLIED ALT PWR - MSG ERASE...
11/5/1996 0:00:00	732<	C/W E.O. 46-71624-3 ENG NO1 DATA ENTRY PLUG MOD	C/W NO1 ENG.
11/5/1996 0:00:00	732<	MSU IN WORK-C/W E.O. 46-71624-3 ON NBR 1 ENG.	PREV C/W PER ERA 278531-14 PER T/A HAN...
11/5/1996 0:00:00	732<	WORKING E.O. 46-71624-3 NO 2 ENG	C/W E.O. NO 2 ENG
11/5/1996 0:00:00	732<	C/W E.O. 46-71624-3ENG NO1 DATA ENTRY PLUG MOD	C/W NO1 ENG
11/5/1996 0:00:00	732<	C/W E.O. 46-71624-3 NO2 ENG DATA ENTRY PLUG MOD	C/W NO2 ENG.
11/5/1996 0:00:00	732<	C/W E.O. 46-71624-3 NO1 ENG DATA ENTRY PLUG MOD	C/W NO1 ENG
11/5/1996 0:00:00	732<	AM6 PER OMI-C/W EO 46-71624-3 ON NBR 1 ENG DATA ENTRY...	STEPS 1-5 OF ME 05 COMPLETE DATA PLUG...
11/5/1996 0:00:00	732<	C/W E.O. 46-71624-3 ENG NO2 DATA ENTRY PLUG MOD	C/W NO2 ENG
11/5/1996 0:00:00	732<	WORKING E.O. 46-71624-3 NO 1 ENG	C/W E.O. NO 1 ENG
11/5/1996 0:00:00	732<	C/W E.O. 46-71624-3 NO2 ENG DATA ENTRY PLUG MOD	N/A NO2 ENG/7327/P02/H1.0M5T0JC1105623...
11/5/1996 0:00:00	732<	C/W E.O. 46-71624-3 NO1 ENG DATA PLUG MOD	C/W NO1 ENG
11/5/1996 0:00:00	732<	C/W E.O. 46-71624-3 NO2 ENG DATA ENTRY PLUG MOD	C/W NO2 ENG
11/5/1996 0:00:00	732<	A/C DUE E.O. 46-71624-3 NO 2 ENG	E.O. N/A NO 2 ENG PER ATLMA GANOPULOS...
11/5/1996 0:00:00	732<	AM6 PER OMI-C/W EO 46-71624-3 ON NBR 2 ENG DATA ENTRY...	STEPS 1-5 OF ME04 COMPLETE DATA PLUG...
11/5/1996 0:00:00	732<	A/C DUE E.O. 46-71624-3 NBR 1 ENG	E.O N/A NO 1 ENG THIS PER ATLMA GANOPU...
11/5/1996 0:00:00	732<	MSU IN WORK-C/W E.O. 46-71624-3 ON NBR 2 ENG.	THIS ITEM N/A PER T/A HANSON ON THIS ENG.
11/6/1996 0:00:00	732<	MSU--DUE E.O. 46-71624-3 ON NBR 1 DATA ENRY PLUG MOD	C/W MSU CKS OK
11/6/1996 0:00:00	732<	PER OMI C/W EO 46-71624-3 ON NO 1 ENG DATA ENTRY PLUG...	COMPLETED EO
11/6/1996 0:00:00	732<	PER OMI C/W EO 46-71624-3 ON NO 2 ENG DATA ENTRY PLUG...	ENG NOT APPLICABLE
11/6/1996 0:00:00	732<	PER OMI C/W EO 46-71624-3 ON NO 1 ENG DATA ENTRY PLUG...	COMPLIED WITH EO
11/6/1996 0:00:00	732<	PER OMI C/W EO 46-71624-3 ON NO 1 ENG DATA ENTRY PLUG...	ENG NOT APPLICABLE
11/6/1996 0:00:00	732<	PER OMI C/W EO 46-71624-3 ON NO 2 ENG/A7327/P02/H1.0M5...	COMPLIED WITH EO

Figure 10. Screenshots of DataJewel

7.3 Discussion

We think the DataJewel architecture is also well adapted to areas like homeland security, market basket analysis or intrusion detection. Homeland security tasks like identifying suspicious behavior can be supported by our architecture in several powerful ways. For example, different event attributes can be associated with each other even though their events take place at different dates, months or possibly years. For intrusion detection, data may be aggregated hourly instead of daily, therefore an additional visualization technique would need to be added to the visualization component.

In the context of market basket analysis, many algorithms have already been proposed and successfully used to find patterns. The discovered rules look like: *If a customer buys bread and sugar then she is likely to buy beer as well.*

These algorithms look for items that are frequently bought together, however, they do not make use of the time that is associated with each transaction. Analyzing market basket databases over time can reveal a new set of patterns like: *Customers are likely to buy cereal and fruits in the beginning of the week and alcohol and candies at the end of the week.*

Note that our approach would be suitable for these datasets even though the dimensionality of market basket databases is typically very high (hundreds or thousands of items). Each item is usually modeled as one attribute and a record corresponds to all items purchased by one customer. Instead, for our approach we would map all items to different events of <u>one</u> attribute and store the frequency of the corresponding items bought per day. If the number of items is very large, a concept hierarchy could be used to generalize to fewer items, as outlined in section 3.

8 Conclusions

Visualization, mining algorithms and databases are main areas in the field of KDD. Most research concentrates in just one of these areas. Our work is based on an integrated approach that we believe can significantly improve the discovery of useful and understandable patterns. We present a novel user-centric architecture for temporal data mining, tightly integrating a visualization, an algorithmic and a database component. We introduce a new visualization technique called CalendarView for representing temporal data. One main contribution is the use of the same visualization for the data and for the computed patterns. In addition, we designed an interface of assigning colors to categories, which is used by both the user and the algorithms. On the one hand, the user can steer the exploration or incorporate his domain knowledge, on the other hand, the algorithm can suggest meaningful color mappings based on the pattern discovered. By precomputing sufficient statistics from the initial datasets, our approach scales up to very large databases.

In our future work, we will apply DataJewel to different areas, using the extensible architecture to add new visualization and algorithmic components. We will investigate how our approach can be extended to fit different data types like text or multimedia data.

9 References

[1] Ankerst M., Ester M., Kriegel H.-P.: Towards an Effective Cooperation of the Computer and the User for Classification, SIGKDD 2000, Boston, MA.

[2] Antunes, C.M., Oliviera A.L.: Temporal Data Mining: an overview, SIGKDD 2001 Workshop on Temporal Data Mining, San Francisco, CA.

[3] C. Daassi, M. Dumas, M-C. Fauvet, L. Nigay, P-C. Scholl : Visual Exploration of Temporal Object Databases, Proc. of BDA'00, 24-27 October 2000, Blois, France, pp. 159-178.

[4] Gehrke J., Ramakrishnan R., Ganti V.: RainForest – A Framework for Fast Decision Tree Construction of Large Datasets, Data Mining and Knowledge Discovery journal, Vol. 4, p.122-162, Kluwer, 2000.

[5] Grinstein G., Ankerst M., Keim D.A.: Visual Data Mining: Background, Techniques and Drug Discovery Applications, SIGKDD 2002, Tutorial, Edmonton, Canada.

[6] Havre S., Hetzler E., Whitney P., Nowell L.: ThemeRiver: Visualizing Thematic Changes in Large Document Collections", IEEE Transactions on Visualization and Computer Graphics, Vol.8, No. 1, January-March 2002.

[7] Hellerstein J.M., Avnur R., Raman V., Informix under CONTROL: Online Query Processing, Data Mining and Knowledge Discovery journal, Vol. 12, p.281-314, Kluwer, 2000.

[8] Hinneburg A., Keim D.A. Wawryniuk M.: HD-Eye: Visual Mining of High-Dimensional Data, IEEE Computer Graphics and Applications, Vol. 19, No. 5, 1999.

[9] Keim D.A., Hao M.C., Dayal U.: Hierarchical Pixel Bar Charts, IEEE Trans. On Visualization and Computer Graphics, Vol. 8, No. 3, pp 255-269, 2002.

[10] Kolluri V., Provost F.:A Survey for Scaling Up Inductive Algorithms, Data Mining and Knowledge Discovery journal, Vol. 2, p.131-169, Kluwer, 1999.

[11] Mackinlay J.D., Robertson G.G., deLine R.: Developing Calendar Visualizers for the Information Visualizer. Proc. UIST '94, 1994.

[12] Sarawagi S, Thomas S., Agrawal R.: Integrating Mining with Relational Database Systems: Alternatives and Implications. SIGMOD Conference 1998, 343-354.

[13] Trueblood R.P., Lovett J.N. Jr.: Data Mining and Statistical Analysis using SQL, Apress, ISBN 1893115542, 2001.

[14] Van Wijk J.J., Van Selow, E.R.: Cluster and Calendar based Visualization of Time Series Data, IEEE InfoVis '99, San Francisco, October.

[15] Yang L.: Interactive Exploration of Very Large Relational Datasets through 3D Dynamic Projections, SIGKDD 2000, pp.236-243, Boston, MA.