Querying Event Sequences by Exact Match or Similarity Search: Design and Empirical Evaluation

Krist Wongsuphasawat[a,b,c,*], Catherine Plaisant[a,c], Meirav Taieb-Maimon[a,d], Ben Shneiderman[a,b,c]

[a] *Human-Computer Interaction Lab*
[b] *Department of Computer Science*
[c] *University of Maryland, College Park, MD, USA*
[d] *Ben-Gurion University of the Negev, Beer-Sheva, Israel*

Abstract

Specifying event sequence queries is challenging even for skilled computer professionals familiar with SQL. Most graphical user interfaces for database search use an exact match approach, which is often effective, but near misses may also be of interest. We describe a new similarity search interface, in which users specify a query by simply placing events on a blank timeline and retrieve a similarity-ranked list of results. Behind this user interface is a new similarity measure for event sequences which the users can customize by four decision criteria, enabling them to adjust the impact of missing, extra, or swapped events or the impact of time shifts. We describe a use case with Electronic Health Records based on our ongoing collaboration with hospital physicians. A controlled experiment with 18 participants compared exact match and similarity search interfaces. We report on the advantages and disadvantages of each interface and suggest a hybrid interface combining the best of both.

*Corresponding author

Email addresses: kristw@cs.umd.edu (Krist Wongsuphasawat), plaisant@cs.umd.edu (Catherine Plaisant), meiravta@bgu.ac.il (Meirav Taieb-Maimon), ben@cs.umd.edu (Ben Shneiderman)

URL: http://www.cs.umd.edu/~kristw (Krist Wongsuphasawat), http://www.cs.umd.edu/hcil/members/cplaisant (Catherine Plaisant), http://www.cs.umd.edu/~ben (Ben Shneiderman)

Keywords: Temporal Categorical Data, Event Sequence, Temporal Query Interface, Similarity Search, Similarity Measure, Similan

1. Introduction

Life can often be described as a series of time-stamped event sequences. If decision-makers have sufficiently powerful tools to query these event sequences, they can discover important patterns. Health organizations have Electronic Health Record (EHR) databases containing millions of records of patient histories, which document heart attacks, hospital admissions, medication orders, treatments, lab results, etc. Transportation management systems keep records of traffic incidents in which each record includes a sequence of incident management events, such as incident notification or arrival time of each unit on the scene. Academic institutions keep detailed records of the educational advancement of their students, such as completing classes, thesis defense or graduation. Three examples of event sequences are shown below.

Patient#01 – (6:11am, Arrive hospital), (6:15am, Emergency Room), (9:05am, ICU), ...
Incident#243 – (8:05pm, Incident Notification), (8:10pm, Police arrived), ...
Student#46311621 – (28 Aug'07, Enter PhD program), (30 Apr'10, Proposal), ...

Querying these event sequences to answer specific questions or look for patterns is an important activity, such as finding patients who were transferred from an "Emergency room" to the "ICU (Intensive Care Unit)" and "Die", incidents in which the police "arrived" 2 hours after "incident notification" or a PhD student who "proposed" a dissertation topic twice before "graduated". While this paper focuses on the medical domain because our case study was done with physicians, the techniques were designed for event sequences, and thus widely applicable to event sequences in other fields, such as incident management, academic records analysis, manufacturing process review, log analysis, or the study of human activities.

1.1. Example of Event Sequence Analysis

Our physician partners in the Emergency Department at the Washington Hospital Center are analyzing sequences of patient transfers for quality assurance. One of their interests are the monitoring of *bounce backs*, which occurred when a patient's level of care was decreased then increased back again urgently, such as: 1) Patients who were transferred from the ICU to

the Floor (normal bed) and then back to the ICU 2) Patients who arrived at the emergency room then were transferred to the Floor and then back to the ICU. Time constraints are also associated with these sequences (e.g. the bounce backs should occur within a certain number of hours).

The bounce back patients correspond to a quality metric for the hospital and are difficult to monitor. The physicians have been using MS Excel to find these bounce back patients. They exported data from the database and wrote formulas to express the queries. An interview with the physician who performed these tasks revealed frustration with the approach because of its complexity and time-consuming aspects (it took many hours to create the formulas). We also asked about the possibility of performing these queries using SQL. He explained that SQL was even harder for him and he was not quite sure how to start (even though he had earned a computer science undergraduate degree in addition to his medical degree.)

1.2. Motivation for Similarity Search for Event Sequences

Specifying temporal queries in SQL is difficult even for computer professionals specializing in such queries. We gave 6 computing students who had completed a database course, the schema of a simple dataset and asked them to write a SQL query to find patients who were admitted to the hospital, transferred to Floor, and then to ICU (no time constraints were to be specified). Even with this simplified query, only one participant succeeded after 30 minutes, adding evidence that SQL strategies are challenging to use for temporal event sequences.

Researchers have made progress in representing temporal abstractions and executing complex temporal queries (Snodgrass, 1987, 1995; Clifford and Croker, 1987), but there is little research that focuses on making it easy for end users such as medical researchers, traffic engineers, or educators to specify the queries and examine results interactively and visually.

To the best of our knowledge, existing visual temporal query tools have used an *exact match* approach, in which each query is interpreted as "every record in the result MUST follow these constraints". As a result, the tool returns only the records that strictly follow every constraint in the query. This approach works well when the users are fairly certain about their query (e.g. "find all patients admitted to the emergency room within a week after leaving the hospital.")

However, exploratory search (Tukey, 1977; White and Roth, 2009), in which users are uncertain about what they are looking for, is gaining more

attention. When using the exact match, broad queries return too many results that are not relevant. Narrow queries miss records that may be "just off" (e.g. 7.5 days instead of 7 days as specified in the query). A more flexible query method could help the exploratory searchers.

1.3. Similarity Search for Event Sequences

A *similarity search* approach has been used in other systems to query other types of data, such as images or text. In this approach, users can sketch an example of what they are seeking and get similar results. The users then receive a ranked list of results, sorted by similarity to the query. The key to this approach is the similarity measure, which is used to calculate the similarity score between the query and every record, so all records then can be sorted by similarity to the query.

Our preliminary work (Wongsuphasawat and Shneiderman, 2009) designed a similarity measure for event sequences called the *Match and Mismatch (M&M)* measure. We also introduced a simple query interface called *Similan*, that employed the M&M measure and allowed users to select an existing record in the database as a query and search for similar records. This prototype was seen as promising by medical researchers.

However, Similan's usefulness is limited for several reasons: it only allows the users to select an existing record from the database as a query (not to specify an example of their choice), the visualization can be misleading and frustrating to users in some situations, and the similarity measure is not flexible enough to support different definitions of similarity for different tasks.

Therefore, to address these limitations, we developed a new version of the similarity measure and the user interface, which are both presented in this paper. We present *Similan2*, a query interface which allows the users to draw an example of a time-stamped event sequence by placing events on a blank timeline and search for records that are similar to their example using the M&M measure v.2, a new version. The M&M measure v.2 is designed to be faster than the first version and customizable by four decision criteria, responding to users' need for richer and more flexible definitions of similarity. Similan2 allows the users to customize the parameters in the M&M measure v.2 via the user interface and also changes how events are visualized on the timeline.

1.4. Motivation for a Controlled Experiment

Using the Multi-dimensional In-depth Long-term Case Study methodology (Shneiderman and Plaisant, 2006), We worked with a physician by assisting him through the analysis of his data using two query tools: LifeLines2 (Wang et al., 2008, 2009) (an exact match user interface from our research group) and Similan2 (similarity search). The physician reported that he was able to specify his queries easily in much shorter time than with the spreadsheet, and that he discovered additional patients who he had missed using his earlier work with Excel. He clearly stated that visualizing the results gave him a better understanding of the data, which could not have been achieved from his spreadsheet or an SQL query.

He also hinted at advantages and disadvantages of both visual approaches. For example he felt that similarity search made it easier to specify the pattern but that looking at the results ranked by similarity was difficult and sometimes frustrating as he was not always confident that the similarity measure was adequately computed to fit his needs. (The computation is explained in details later in this paper.) Those contrasting benefits led us to design the controlled experiment to see if we could confirm those impressions and better understand which query method is better suited for different tasks.

In summary, this paper provides the following contributions.

1. Similan2, a similarity query interface for event sequences, which allows users to draw an example of what they are looking for directly.
2. the M&M measure v.2, a similarity measure for event sequences, which can be customized according to users' need.
3. a controlled experiment to compare the features of the exact match and similarity search interfaces using LifeLines2 and Similan2, respectively. We summarize the advantages and disadvantages of each interface and discuss possible directions for combining them.

The rest of this paper is organized as follows: Section 2 reviews the background and related work, Section 3 describes the user interfaces and similarity measure, Section 4 explains the experimental design, Section 5 reports the results from the experiment, and Section 6 concludes with a suggestion of a hybrid interface.

2. Background and Related Work

2.1. Query Languages

A traditional approach to query temporal data was to use database query languages. According to Chomicki (1994) and Tansel and Tin (1997), many research projects were conducted on designing temporal databases and extending standard query languages into temporal query languages. Some of the well-known languages were TQuel (Snodgrass, 1987), TSQL2 (Snodgrass, 1995) and Historical Relational Data Model (HRDM) (Clifford and Croker, 1987). However, these temporal query languages were built on top of their specific data models and users had difficulty in learning their unique syntaxes, concepts, and limitations. They also supported only exact match.

2.2. Query-by-Example Languages

To provide a high-level language that offered a more convenient way to query a relational database (RDB), the query-by-example languages were introduced. According to Ozsoyoglu and Wang (1993), the early idea of query-by-example was a language that users entered what they expected to see in a database result table into a form that looked like a result table instead of writing lengthy queries, making it simpler for the users to specify a query. The first was Zloof's *Query-by-Example* (Zloof, 1975), which was refined by others (Chang and Fu, 1980; Klug, 1981; Zloof, 1982; Jacobs and Walczak, 1983; Ozsoyoglu et al., 1989; Tansel et al., 1989).

Time-by-Example (Tansel et al., 1989) followed the Query-by-Example idea and adopted subqueries concepts from *Aggregates-by-Example* (Klug, 1981) and *Summary-Table-by-Example* (Ozsoyoglu et al., 1989) to serve historical relational database (HRDB). HRDB was an extension of RDB that stored the changes of attribute values over time. For example, a `patient` entity had an attribute called `room`, which was changed every time each patient was moved to a new room. HRDB could keep track of the room values and Time-by-Example provided a way to query those values. For instance, the users could ask queries such as list all rooms a patient was in or which patient was in an ICU room between January and March?

However, Time-by-Example supported only exact match, operated only on top of HRDM and still required the users to learn their languages for specifying conditions in complex queries, e.g. "`($sal.T overlaps $dept.T overlaps $msal.T overlaps $m.T) and $sal.v > $msal.v`".

2.3. Query by Graphical User Interfaces (GUIs)

2.3.1. Exact Match Approach

As the graphical user interfaces (GUIs) were becoming more common, many GUIs were developed for temporal data (Aigner and Miksch, 2006; Shahar et al., 2006; Klimov et al., 2009, 2010). Several GUIs used the *exact match* approach, in which users specify exact constraints to construct the queries. These constraints are often specified via controls, such as sliders or drop-down lists. The tool then returns only the records that follow every constraint in the query. Karam (1994) presented a visualization called *xtg*, which allowed users to explore temporal data and do simple searches for events. Hibino and Rundensteiner (1995, 1997) proposed a visual query language and user interface for exploring temporal relationships using slider filters with results displayed in a graph-like visualization. *PatternFinder* (Fails et al., 2006) allowed users to specify the attributes of events and time spans to produce pattern queries that are difficult to express with other formalisms. *LifeLines2* (Wang et al., 2008, 2009) used an alignment, ranking and filtering (ARF) framework to query for temporal categorical records. *ActiviTree* (Vrotsou et al., 2009) provided a tree-like user interface with suggestions about interesting patterns to query for sequences of events. *QueryMarvel* (Jin and Szekely, 2009) utilized and extended the semantic elements and rules of comic strips to construct queries. Instead of following the exact match approach, Similan2 followed the similarity search approach (Section 2.3.2), and applied the concept for querying event sequences.

2.3.2. Similarity Search Approach

Many GUIs followed the *similarity search* approach, in which users could draw an example of what they expect to see as a result of a query. The result from a query was a list of records, sorted by similarity to the given example. Kato et al. (1992) presented *QVE* that accepted a sketch drawn by users to retrieve similar images or time series from the database. *IFQ (In Frame Query)* (Li et al., 1997) was a visual user interface that supported direct manipulation (Shneiderman, 1983) allowing users to combine semantic expressions, conceptual definitions, sketch, and image examples to pose queries. *Spatial-Query-by-Sketch* allowed users to formulate a spatial query by drawing on a touch screen and translated this sketch into a symbolic representation that can be processed against a geographic database. Bonhomme et al. (1999) and Bonhomme and Aufaure (2002) discussed the limitations of previous query-by-sketch approaches and extended the *Lvis* language, which

was developed for spatial data, to temporal data. The new language used visual metaphors, such as balloons and anchors, to express spatial and temporal criteria. *QuerySketch* (Wattenberg, 2001) allowed users to sketch a graph freehand, then view stocks whose price histories matched the sketch. Watai et al. (2007) proposed a web page retrieval system that enables a user to search web pages using the user's freehand sketch. *WireVis*(Chang et al., 2007) introduced techniques to extracted bank accounts that showed similar transaction patterns. To the best of our knowledge, existing event sequence query tools have used an exact match approach. These systems demonstrated the similarity search concept in other types of data and inspired us to develop a similarity search tool for event sequences.

Timesearcher (Hochheiser and Shneiderman, 2004) visualized multiple timelines as line charts on the same plane, using horizontal and vertical axis to represent time and value, respectively. Users drew timeboxes, rectangular widgets that could be used to specify query constraints, on the timeline to query for all time series that passed through those timeboxes. In Timesearcher, users could draw an example (timeboxes) to specify the query, but the timeboxes were converted into exact rules, e.g. January $<$ time $<$ March and $100 <$ value < 200 , when processing the query in the background. Similan2 allowed the users to draw an example, but did not convert the example into any exact rule. Instead, it compared the example with each record directly and sorted the result by similarity to the example.

2.4. Similarity Measure

Pattern matching computes a boolean result indicating whether an event sequence matches the specified pattern, or it does not. In contrast, *similarity measure* calculates a real number measurement that expresses how similar is an event sequence to the specified pattern.

2.4.1. Numerical Time Series

Many similarity measures had been proposed for comparison between series of numerical values measured over time, such as stock price. Event sequences, in contrast, are series of categorical values measured over time. Hence, these approaches were not directly applicable to event sequences because they were designed to capture the difference between numerical values, not categorical.

Nevertheless, there were some common concepts that worth mentioning here. The first concept was *lock-step* measures, which compared the i-th

point of one time series to the i-th point of another, such as the well-known *Euclidean distance*. However, since the mapping between the points of two time series was fixed, these measures were sensitive to noise and misalignments in time. The M&M measure was different from lock-step measures because it did not fix the mapping of i-th events together.

The second concept, *elastic* measures, allowed comparison of one-to-many points (e.g., Dynamic time warping (DTW) (Berndt and Clifford, 1994) and one-to-many / one-to-none points (e.g., Longest Common Substring (LCSS)). The sequences were *stretched* or *compressed* non-linearly in the time dimension to provide a better match with another time series. Unlike elastic measures, the M&M measure did not allow one-to-many mapping.

2.4.2. String and Biological Sequences

Edit distance is the number of operations required to transform one string into another string. The lower the number is, the more similar the strings are. *Hamming distance* (Hamming, 1950), *Levenshtein distance* (Levenshtein, 1966) or *Jaro-Winkler distance* (Winkler, 1999) are some examples. The best known such distance is the *LCSS* distance (André-Jönsson and Badal, 1997). A more completed survey can be seen from (Navarro, 2001).

One neighbor area is biological sequence searching. There exist many algorithms for comparing biological sequence information, such as the amino-acid sequences of different proteins or the nucleotides of DNA sequences. BLAST (Altschul et al., 1990), FASTA (Pearson and Lipman, 1988) and the TEIRESIAS algorithm (Rigoutsos and Floratos, 1998) were some examples.

Mongeau and Sankoff (1990) defined a similarity measure specifically for comparing musical pieces based on number of transformations required to transform one into another. They allowed one-to-many mapping called *consolidation/fragmentation*, which is similar to time warping. Gómez-Alonso and Valls (2008) proposed a similarity measure for sequences of categorical data (without time) based on edit distance.

These approaches considered the difference in ordering and existence, but did not consider the time that events occurred. Event sequences might occur at non-uniform intervals, which made the timing become important. Also, more than one events could occur at the same time while two characters or amino acids could not occur at the same position in the string or biological sequence.

2.4.3. Event Sequences

Mannila and Ronkainen (1997) introduced a similarity measure for event sequences based on three edit operations: *insert*, *delete* and *move*. The *move* operation was included to incorporate the occurrence time of the events. This approach allowed only monotonic mapping, which means that the matched events in the target and candidate sequences must be in similar order, and did not offer a user interface. Sherkat and Rafiei (2006) binned the timeline into intervals and compared events within each interval.

The Match & Mismatch (M&M) measure v.1 (Wongsuphasawat and Shneiderman, 2009) calculated a similarity score from two types of difference: time difference of matched events and number of mismatches. It supported matching that may not preserve the order of event sequence (non-monotonic). This paper continues the work on the M&M measure. A few projects were also developed in parallel with the work in this paper. *Timed String Edit Distance* (Dobrisek et al., 2009) inserted timed null symbols into event sequences before matching. It allowed matching between events with different event types and measured two types of difference: time difference and event type difference (symbol dissimilarity). Vrotsou (2010) and Vrotsou and Forsell (2011) identified nine measures to cover several aspects of similarity. This approach also considered multiple occurrences of the target sequence in the candidate sequence. Obweger et al. (2010) defined *single-event similarity* by comparing event attributes. Their event sequence similarity then combined single-event similarities, order of events and time that the events occurred with weights and more options. However, their computation time to find the best match is exponential while others are polynomial.

Some methods extracted "fingerprints" from event sequences and compared the fingerprints instead of comparing the event sequences directly. Mannila and Moen (1999) detected similar event types by comparing their context. They converted each context (event sequence around the selected event type) into feature vectors and developed methods for comparing these vectors. Mannila and Seppänen (2001) mapped event sequences into points in k-dimensional Euclidean space using a random function and searched for similar event sequences from their k-dimensional projections.

The growth of measures for event sequence similarity demonstrates the importance of these search capabilities in many domains beyond our medical interests, such as human activity event streams, business transactions, or legal actions.

3. Systems Description

This section describes the user interfaces in more detail. Section 3.1 and 3.2 explain the main features of the exact match interface (Lifelines2) and the similarity search interface (Similan2), respectively. LifeLines2 is a former work which is described here only for the purpose of the controlled experiment while Similan2 is presented in this paper for the first time. Similan2 allows the users to draw an example of event sequence on a blank timeline to query for similar event sequences using the new M&M measure v.2, which was designed to address the limitations of the M&M measure in the preliminary work. The M&M measure v.2 is explained in Section 3.4.

3.1. Exact Match Interface: LifeLines2

LifeLines2 (Figure 1) is a Java application, utilizing the Piccolo 2D graphics framework (Bederson et al., 2004). In LifeLines2, each record is vertically stacked on an alternating background color and identified by its ID on the left. Events appear as triangle icons on the timeline, colored by their type (e.g. Admission or Exit.) Placing the cursor over an event pops-up a tooltip providing more details. The control panel on the right side includes filters and other controls. The visibility and color of each event type (category) can be set in the control panel.

Users can select an event type to *align* all the records. For example, Figure 1 shows records aligned by the `Admit` event. When the alignment is performed, time is recomputed to be relative to the alignment event.

Users can apply the *sequence* filter to query records that contain a particular sequence, e.g. finding patients who were admitted, then transferred to a special room and exited. The first step is to select a sequence filter from the "filter by" drop-down list, then several drop-down lists that contain categories will appear. Users then set the values of the 1st, 2nd and 3rd drop-down lists to `Admit`, `Special` and `Exit`, respectively. The records that pass this filter will be selected and highlighted in yellow. A click on "Keep selected" removes the other records.

To query for records that have events occurring at particular intervals, users have to first display the *distribution* of selected events (with the distribution control) then select intervals on the distribution display. For example, to find patients who were admitted, then transferred to the ICU room on the first day of their stay and transferred to the intermediate ICU room on the

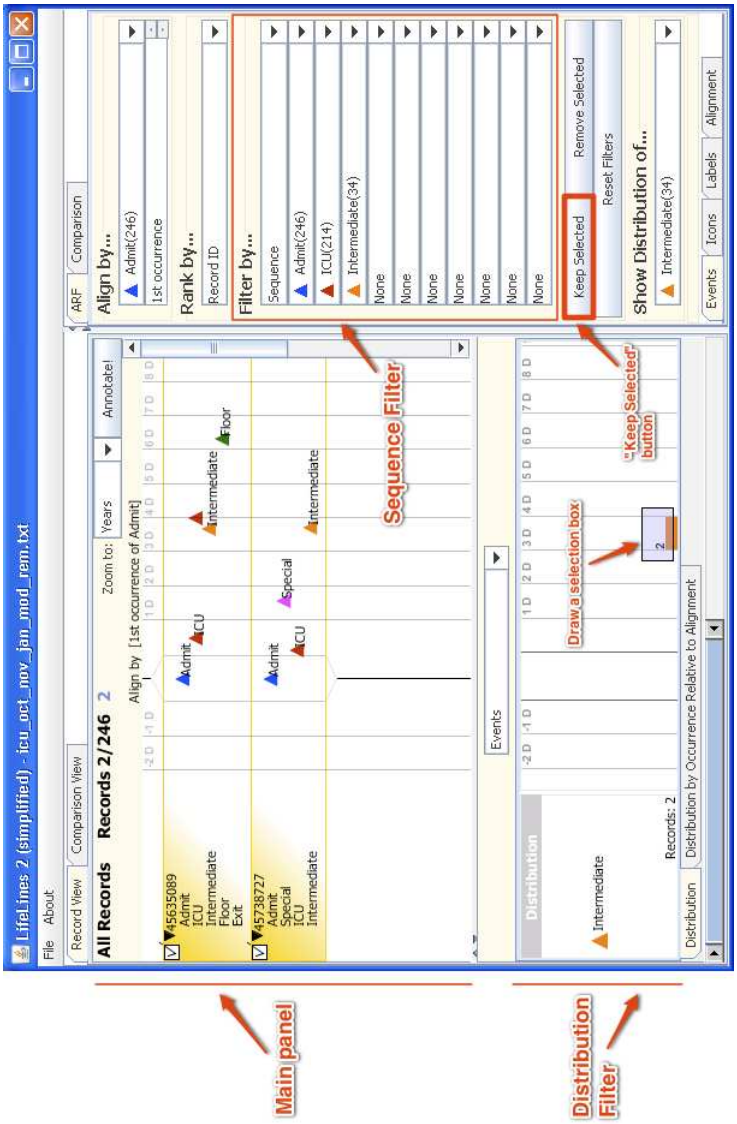

Figure 1: Exact match interface (LifeLines2) showing the results of a query for patients who were admitted to the hospital then transferred to the Intensive Care Unit (ICU) within a day, then to an Intermediate ICU room on the fourth day. The user has specified the sequence filter on the right selecting `Admit`, `ICU` and `Intermediate` in the menus, and aligned the results by the time of admission. The distribution panel in the bottom of the screen shows the distribution of `Intermediate`, which gives an overview of the distribution and has allowed users to select the time range of interest (e.g. on the fourth day) by drawing a selection box on the distribution bar chart.

fourth day, users have to align all the records by `Admit`, show the distribution of `ICU` using the "Show Distribution of" control, then select the 1st day on the distribution of `ICU` events at the bottom of the screen and click on "Keep selected" then show the distribution of `Intermediate` and draw a selection box from the 1st to the fourth day and "Keep selected". (See Figure 1.) A similar process can be used for consecutive interval specification using different alignments and filtering.

3.2. Similarity Search Interface: Similan2

Similan2 (Figure 3), is an Adobe Air Application using the Adobe Flex 3 Framework. The designs of LifeLines2 and Similan2 have evolved in parallel.

Its ancestor, Similan (Figure 2), developed in C#, only allows the users to select an existing record from the database as a query (not to specify an example of their choice). Also, according to the feedback from the usability study, the binned timeline visualization in Similan sometimes confused the users and could be misleading in some situations.

Therefore, Similan2 adopted the basic display of the records from Lifelines2: each record is stacked on the main panel, events are colored triangle icons, and users can customize the visibility and colors of each event type (category). Users can also align all the records by a selected event type (e.g. align by admission to the hospital in Figure 3). Similan2 also employed an improved similarity measure (M&M measure v.2), which will be explained in Section 3.4.

In Similan2, the panel on the top is called the query panel, where users can specify their queries. On the right side is the control panel, which provides controls for users to customize the search parameters. The largest area on the screen is the main panel, where all records in the data are listed.

To perform a query users first create or select an existing record. For example, to find patients who were admitted, transferred to the ICU room on the first day and then to the intermediate room on the fourth day, users can start by aligning all records by `Admit`. Then users click on the *edit* button on the query panel to open a popup window, and drag and drop events on the empty timeline (i.e. they can select `Admit` from the list of categories shown in the popup and click on *Add*. The cursor will change into a magic wand and they can drop the event on the line). Figure 2 shows the patterns they created `Admit`, `ICU` and `Intermediate` at time 0, on the first day and fourth day, respectively. (See Figure 3.) The only type of time constraint that is currently supported by Similan2 is specifying when each event occurred.

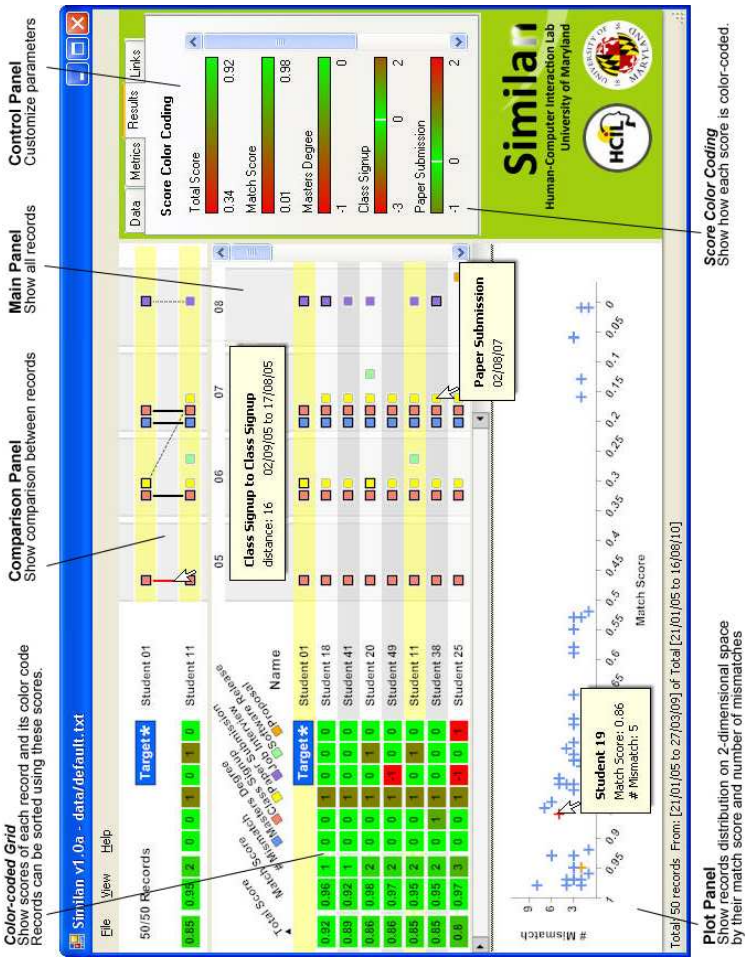

Figure 2: A screenshot of Similan, the predecessor of Similan2. Users can start by double-clicking to select a target record from the main panel. Similan will calculate a score that indicates how similar to the target record each record is and show scores in the color-coded grid on the left. The score color-coding bars on the right show how the scores are color-coded. The users then can sort the records according to these scores. The main panel also allows users to visually compare a target with a set of records. The timeline is binned (by year, in this screenshot). If the users want to make a more detailed comparison, they can click on a record to show the relationship between that record and the target record in the comparison panel on the top. The plot panel at the bottom shows the distribution of records. In this example, the user is searching for students who are similar to Student 01. The user sets Student 01 as the target and sorts all records by total score. Student 18 has the highest total score of 0.92 so this suggests that Student 18 is the most similar student. Student 41 and Student 18 both have one missing paper submission but Student 41 has a lower match score so Student 18 has higher total score.

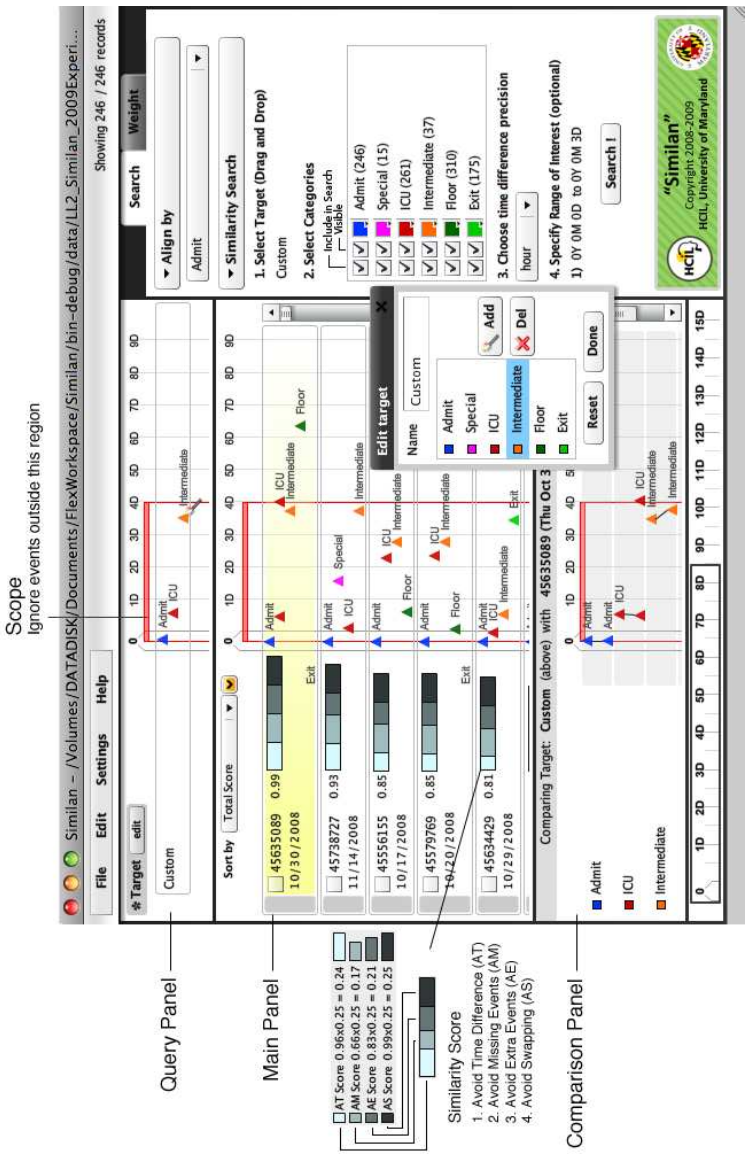

Figure 3: Similarity search interface (Similan2) with the same query as in Figure 1. Users specify the query by placing events on the query panel. To set the time range of interest and focus on events within this range, users draw a red box. After clicking on "Search", all records are sorted by their similarity to the query. The similarity score is represented by a number that is the total score and a bar with four sections. A longer bar means a higher similarity score. Each section of the rectangle corresponds to one decision criterion, e.g. the top two records has longer leftmost section than the third record because it has lower time difference so the Avoid Time Difference Score (AT) is high, resulting in longer bars. Figure 4 shows how users can adjust the weight.

Users can also select any existing record as a query by dragging that record from the main panel and dropping it into the query panel. This is useful for finding patients who exhibit a pattern of events similar to a particular known patient. A time scope can be drawn on the top of the timeline (See red line in Figure 3). In our example query, drawing a scope from the time zero to the end of the fourth day will exclude all other events outside of the scope from the search. If no scope is specified, the entire timeline will selected by default. The unit for time differences (e.g. hours or days) can be selected from a drop-down list. Event categories that should be excluded from the search can be unchecked in the control panel.

After clicking on Search, the records are sorted by their similarity score (with records with the highest scores on the top). Each record has a *score indicator*, a rectangle with four sections of different color (See Figure 3.), inspired by *ValueCharts* (Carenini and Loyd, 2004), a visualization to support decision-makers in inspecting linear models. The length of a score indicator represents total score. It is divided into four colored parts which represent the four decision criteria. The length of each part corresponds to the $weight * score$. Placing a cursor over the score indicator brings up an explanation tooltip.

Users can see a detailed comparison of the query and any other record by dragging that record into the comparison panel in the bottom. Lines are drawn between pairs of events matched by the M&M measure v.2. Moving the cursor over a link displays a tooltip showing the event type, time of both events and time difference.

By default, the search uses default weights, which means that all criteria are equally important. However, users may have different meanings for similarity in mind. Similan2 allows users to adjust the weight of all criteria in the "Weight" tab in the control panel. (See Figure 4.) The weight for each decision criterion can be adjusted with the slider controls, as well as the weight of each event type for each decision criteria, A click on "Apply Weight" refreshes the similarity measures and the order of the records on the display. For example, if the value of time intervals is not important in this task (e.g. finding patients who were admitted, transferred to the special room and exited) the user can set a low weight for "Avoid Time Difference" to reduce its importance. Because the definition of weights can be complex, Similan2 includes sets of preset weight combinations for users to choose from. For instance, one preset is called "Sequence", which uses a low weight for "Avoid Time Difference" and a high weight for "Avoid Missing Events".

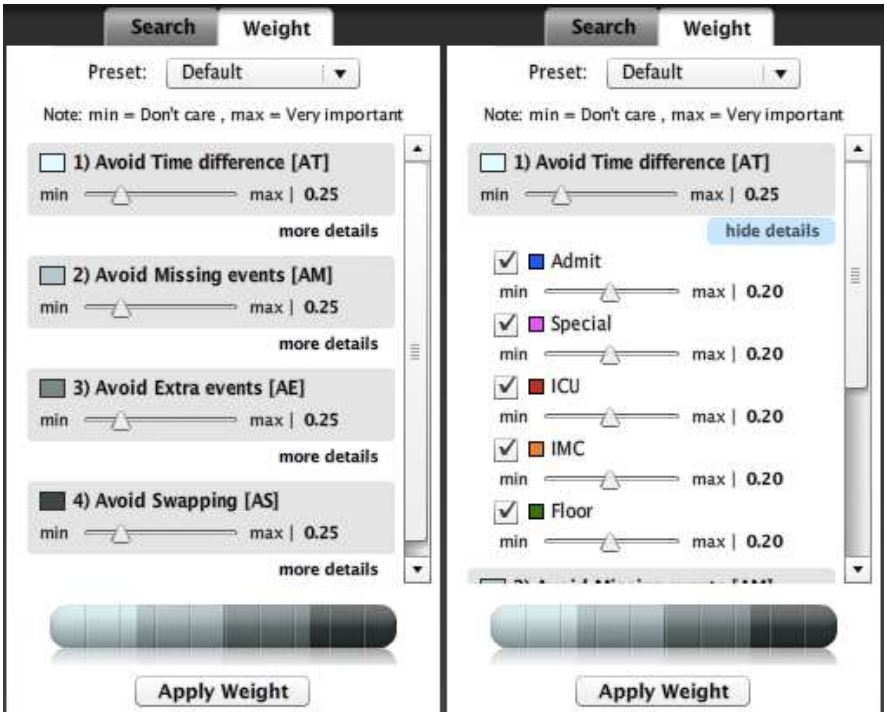

Figure 4: Similan2's control panel has 2 tabs. The first tab is "search" as shown in Figure 3. Another tab is weight and detailed weight – The users can adjust the weight of the four decision criteria using the four sliders in the left figure. For more advanced customization, they can even set the weight for each event type within each decision criterion by clicking on "more details" (right figure).

3.3. The Match and Mismatch (M&M) measure v.1

Many methods for computing a similarity measure between time series have been proposed. However, modifying them to suit event sequences remains an open problem. We presented a similarity measure for event sequences called the *Match and Mismatch (M&M)* measure (Wongsuphasawat and Shneiderman, 2009) and used it in Similan. Based on the idea that similar records should have the same events and the same events should occur almost at the same time, the M&M measure uses the time difference and number of missing and extra events as the definition of similarity. The original M&M measure consists of two steps: *matching* and *scoring*.

1) Matching : The first step is to match the events in the query with events in the compared records. The simplest case is when the events are identical between records. The problem becomes more complex when the set of events in the query does not exactly match those in another record. Since there can be many possible ways to match the events between the two records, the matching has to be carefully chosen. The matching problem can be reduced to a problem called the *assignment problem* (Kuhn, 1955), which is described as follows:

"There are a number of agents and a number of tasks. Any agent can be assigned to perform any task, incurring some cost that may vary depending on the agent-task assignment. It is required to perform all tasks by assigning exactly one agent to each task in such a way that the total cost of the assignment is minimized."

Let the events from one record become agents and the events from another records become tasks, The matching problem now becomes the assignment problem. The original M&M measure then uses the Hungarian Algorithm (Kuhn, 1955; Munkres, 1957) to solve the assignment problem.

2) Scoring : After the matching is completed, the scores can be computed. Scoring is based on a combination of the number of mismatches and time difference. In the original M&M measure, the time difference is converted into a *match score* while the number of mismatches (number of events which occur in the query but do not occur in the compared record, or vice versa) is converted into a *mismatch score*. Match and mismatch scores are combined into *total score*, ranging from 0.01 to 1.00 using a weighted sum. A higher score represents higher similarity. The weight is a customizable parameter that can be adjusted by the users.

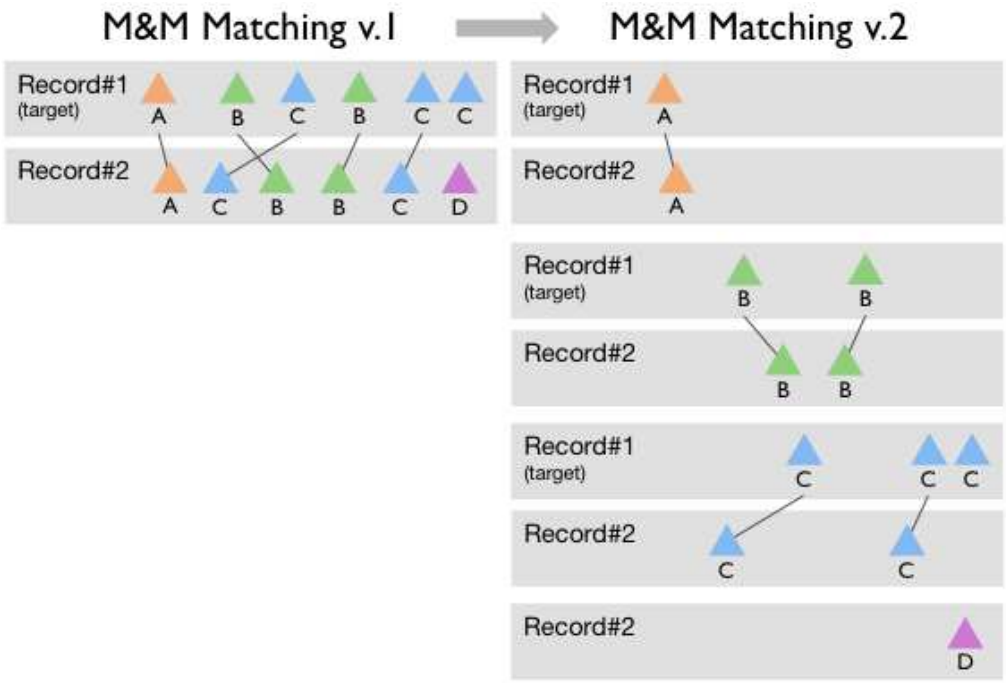

Figure 5: (left) M&M Matching v.1 (right) M&M Matching v.2 – Events in each event type are matched separately.

3.4. The Match and Mismatch (M&M) measure v.2

In this paper, we propose the M&M measure v.2, which improves on the original version in two ways: First, the matching problem is reduced to a simpler problem than the assignment problem. Therefore, the matching algorithm can be improved by using dynamic programming instead of the Hungarian Algorithm. Second, the M&M measure v.2 considers more types of differences. It splits the number of mismatches into number of missing events and number of extra events and also includes number of swaps. Moreover, it increases the flexibility by adding more customizable parameters. The M&M measure v.2 still consists of two steps: *matching* and *scoring*.

1) Matching : The M&M measure does not allow matching between events in different categories and allows only one-to-one matching. For example, event `A` can only match with event `A` and cannot match with event `B` or `C`. (See Figure 5.) Therefore, the matching can be reduced into a simpler problem by separating the matching for each event type.

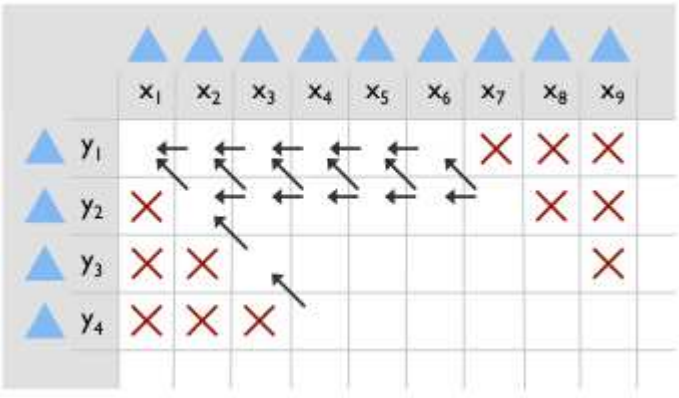

Figure 6: M&M Matching v.2 – Dynamic programming table

The notation below is used to describe an event sequence record, which is a list of timestamped events (t, c). The i-th event in the record is denoted by x_i or (t_i, c_i).

$$X = \{(t, c) \mid t \in Time \text{ and } c \in Categories\} \tag{1}$$

The M&M measure v.2 splits each record in to several lists, one list for each event type. For example, these two records X and Y

$$X = \{(t_1, \text{“}A\text{”}), (t_2, \text{“}A\text{”}), (t_3, \text{“}B\text{”})\}$$

$$Y = \{(u_1, \text{“}A\text{”}), (u_2, \text{“}B\text{”}), (u_3, \text{“}B\text{”})\}$$

$$\text{“}A\text{”}, \text{“}B\text{”} \in Categories$$

are split into X_A, X_B and Y_A, Y_B, respectively.

$$\begin{array}{ccc} X_A = \{(t_1, \text{“}A\text{”}), (t_2, \text{“}A\text{”})\} & , & X_B = \{(t_3, \text{“}B\text{”})\} \\ Y_A = \{(u_1, \text{“}A\text{”})\} & , & Y_B = \{(u_2, \text{“}B\text{”}), (u_3, \text{“}B\text{”})\} \end{array} \tag{2}$$

The problem of “matching events between two records” is then reduced to “matching events between two lists that contain only events in the same event type” multiple times, which is simpler. (See Figure 5.) For example, matching X and Y is reduced to matching X_A with Y_A, and X_B with Y_B. A faster algorithm based on dynamic programming can be used instead of the Hungarian algorithm to find the matching between X_A and Y_A that produces the minimum time difference.

Dynamic Programming Matching Figure 6 shows a dynamic programming table. The value in each cell ($cell(i, j)$) is the minimum cost of matching subsequences $X[1..i]$ and $Y[1..j]$. X must be longer or has equal length with Y. Cross symbols mark the cells that cannot be used because the matches would yield non-perfect matchings for Y. For example, matching y_2 with x_1 will cause y_1 to have no match.

The M&M matching v.2 algorithm (Algorithm 1) starts from the top-left cell and fills the cells from left to right, row by row. For each cell, the cell value is equal to the minimum between:

1. Cost of matching x_i to y_j ($d(x_i, y_j) = |x_i.time - y_j.time|$) plus minimum cost of matching the prefixes (upper-left cell: $cell(i-1, j-1)$)
2. Minimum cost of matching y_j to some x before x_i (left cell: $cell(i-1, j)$)

which can be represented by this formula:

$$cell(i,j) = min \begin{cases} d(x_i, y_j) + cell(i-1, j-1) \\ cell(i-1, j) \end{cases} \quad (3)$$

If choice 1 is selected, that cell maintains a link to its upper-left cell. If choice 2 is selected, that cell maintains a link to its left cell. After filling the entire table, the minimum matching cost is the value of the bottom-right-cell. The matching that produces the minimum cost can be retrieved by backtracking the link, beginning from the bottom-right cell.

Time Complexity If the number of events in X_A and Y_A are n_A and m_A, and $n_A > m_A$, the time to match the events between X_A and Y_A with dynamic programming is

$$O((n_A - m_A) * m_A) \quad (4)$$

Using the matching v.1 based on the Hungarian algorithm, the time complexity of matching events between X and Y is

$$O((max(n_A, m_A) + max(n_B, m_B) + max(n_C, m_C) + ...)^3) \quad (5)$$

Using the matching v.2, the time complexity is reduced to:

$$O((n_A - m_A) * m_A + (n_B - m_B) * m_B + (n_C - m_C) * m_C + ...) \quad (6)$$

2) Scoring : Once the matching is completed. The scores can be derived from the matching. The first version of the M&M measure considers only

Algorithm 1 M&M Matching v.2

```
1: n ← length(X)
2: m ← length(Y)
3: diff ← n − m
4: c ← array[diff+1][m]
5: for j := 0 to m − 1 do
6:    for i := 0 to diff do
7:       cost ← d(x_{j+i}, y_j)
8:       if j > 0 then
9:          cost ← cost + c[i][j − 1]
10:      end if
11:      if i > 0 then
12:         c[i][j] ← min(cost , c[i − 1][j])
13:      else
14:         c[i][j] ← cost
15:      end if
16:   end for
17: end for
```

two types of difference: time difference and number of mismatches (missing or extra events). In this second version, we decided to split the number of mismatches into number of missing and extra events because these two numbers can have different significance. For example, users may not care about extra events but want to avoid missing events, or vice versa. We also included the number of swaps because sometimes the users want the events in order but sometime the order is not significant. Therefore, the M&M measure v.2 considers four types of difference and allows users to customize each type of difference in more details for each event type. The four types of differences are listed as follows:

1. A *match* event is an event that occurs in both the query and the compared record. The *time difference (TD)* is a sum of time differences within each pair of matched events. The time difference is kept separately for each event type. Users also can specify what time unit they want to use for the time difference.
2. A *missing* event is an event that occurs in a query record but does not occur in a compared record. The *number of missing events (NM)* is counted for each event type.

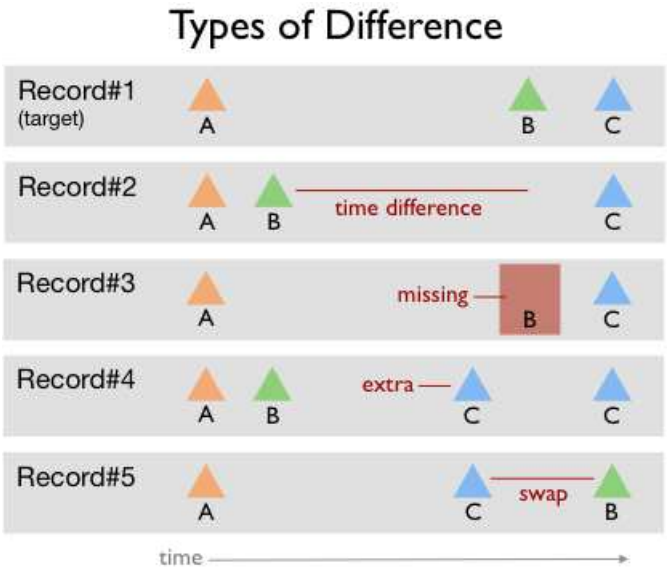

Figure 7: Four types of difference: time difference, missing events, extra events and swaps

3. An *extra* event is an event that does not occur in a query record but occurs in a compared record. The *number of extra events (NE)* is counted for each event type.
4. A *swapping* event occurs when the order of the events is reversed. The *number of swapping events (NS)* is counted for each pair of event categories. For example, in Figure 7, the query has `A` followed by `B` then `C` but record#5 has `A` followed by `A` then `C` then `B`. If you draw a line from query's `C` to record#5's `C` and do the same for `B`, it will create one crossing. So, the number of swaps between `B` and `C` ($NS_{B,C}$) is 1 while $NS_{A,B}$ and $NS_{A,C}$ are both 0.

Since the time difference may be not equally important for all categories, the *total time difference* ($\sum TD$) is a weighted sum of time difference from each event type. Users can adjust what is important by setting these weights ($\sum w^{TD} = 1$).

$$\sum TD = w_A^{TD} * TD_A + w_B^{TD} * TD_B + ... \tag{7}$$

Likewise, the *total number of missing events* ($\sum NM$), *total number of extra events* ($\sum NE$) and *total number of swapping* ($\sum NS$) are calculated

from weighted sums.

$$\sum NE = w_A^{NE} * NE_A + w_B^{NE} * NE_B + ... \quad (8)$$

$$\sum NM = w_A^{NM} * NM_A + w_B^{NM} * NM_B + ... \quad (9)$$

$$\sum NS = w_{A,C}^{NS} * NS_{A,C} + w_{B,C}^{NS} * NS_{B,C} + ... \quad (10)$$

Four Decision Criteria The 4 types of differences are normalized into a value ranging from $0.01 - 0.99$ and called *penalties*. The total time difference ($\sum TD$), total number of missing events ($\sum NM$), number of extra events ($\sum NE$) and total number of swappings ($\sum NS$) are normalized into *TD penalty*, *NM penalty*, *NE penalty* and *NS penalty*, respectively. The 4 penalties are converted into these 4 decision criteria:

1. *Avoid Time Difference (AT)* $= 1 - TD$ penalty
2. *Avoid Missing Events (AM)* $= 1 - NM$ penalty
3. *Avoid Extra Events (AE)* $= 1 - NE$ penalty
4. *Avoid Swapping Events (AS)* $= 1 - NS$ penalty

Total Score The total score is a weighted sum of the four decision criteria. The users can adjust the weights ($w_{AT}, w_{AM}, w_{AE}, w_{AS}$) to set the significance of each decision criteria ($\sum w = 1$).

$$T = w_{AT} * AT + w_{AM} * AM + w_{AE} * AE + w_{AS} * AS \quad (11)$$

The total score (T) is from 0.01 to 0.99. The higher score represents higher similarity. We selected the weighted sum model to combine the score because of its simplicity and ease of presentation to the users.

4. Evaluation

We conducted a controlled experiment comparing 2 interfaces: LifeLines2, an exact match interface, and Similan2, a similarity search interface. Our goal was not to determine which tool was superior (as they are clearly at different stages of refinement and represent different design concepts), but to understand which query method was best suited for different tasks. Another goal was to observe the difficulties that users encountered while using the interfaces to perform given tasks. Both interfaces were simplified by hiding certain controls to focus on the query features we wanted to compare.

4.1. Research questions

The evaluation was conducted to answer these research questions:

1) Are there statistically significant differences in performance time and performance accuracy between the two interfaces while performing different tasks?

2) Are there statistically significant differences in performance time and performance accuracy between the different tasks while using each interface?

3) Is there a statistically significant difference between the subjective ratings given by the users to the two interfaces?

4.2. Participants

Eighteen graduate and senior undergraduate students participated in the study. We recruited computer science students who are assumed to have high level of comfort with computers but no knowledge of either interface. The participants included 13 men and 5 women, 20 to 30 years of age. Participant received $20 for their 90-minute participation. To provide the motivation to perform the tasks quickly and accurately, an additional nominal sum was promised to the fastest user with the fewest errors of each interface.

4.3. Apparatus

The participants were required to perform the given tasks with the two interfaces: LifeLines2 and Similan. The two software interfaces were running on an Apple Macbook Pro 15" with Windows XP operating system. The participants controlled the computer using a standard mouse.

4.3.1. Tasks

The tasks were designed based on real scenarios provided by physicians and simplified to make them suitable for the time limit and participants who had never used the interfaces before. Participants were requested to find patients in the database who satisfied the given description. To avoid the effect of alignment choice, all tasks contained an obvious sentinel event (e.g. Admit). We considered these factors when designing the tasks:

1. *Query type*: Either a sequence description was provided or an existing record was used as a query.
2. *Time constraint*: Present or not
3. *Uncertainty*: Yes or No, e.g. the number of events may be precise or not, the time constraint may be flexible or not.

The tasks that were used in the experiment are listed below:

Task type 1 – Description without time constraint, no uncertainty

1: "Find at least one patient who was admitted, transferred to Floor then to ICU."

1.2: "Count all patients who fit task 1 description"

Task 1 was designed to observe how quickly the participants can use the interface to specify the query while task 1.2 focused on result interpretation and counting.

Task type 2 – Description with time constraints, no uncertainty

2: "Find at least one patient who was admitted and transferred to Intermediate on the second day then to ICU on the third day."

2.2: "Count all patients who passed task 2 description."

Task type 3 – Description with uncertainty, without time constraint

3: "Find a patient who best matches the following conditions: Admitted and then transferred to special room approximately 2 times and transferred to ICU room after that. If you cannot find any patient with exactly 2 transfers to the special room, 1-3 transfers are acceptable."

3.2: "Count all patients who passed task 3 description."

Task type 4 – Description with uncertainty and time constraint:

"Find a patient who best matches the following conditions: Admitted, transferred to Floor on the first day, ICU approximately at the end of the third day. The best answer is the patient who was transferred to ICU closest to the given time as possible."

Task type 5 – Existing record provided as query:

"Find a patient who was transferred with the most similar pattern with patient no. xx during the first 72 hours after being admitted. Having everything the same is the best but extra events are acceptable."

4.3.2. Data

We used a modified version of the deidentified patient transfer data provided by our partners. The data contained information about when patients were admitted (Admit), transferred to Intensive Care Unit (ICU), transferred to Intermediate Care Unit (Intermediate), transferred to a normal room (Floor), and exited (Exit).

4.3.3. Questionnaire

A 7-item, 7-point Likert-scale questionnaire was devised by the experimenter to measure the learnability and ease or difficulty of using the interfaces while performing the different tasks, and the level of confidence of the answers they provided for the different tasks. The highest (positive, such as "very easy" or "very confident") score that could be attained on the measure was 7; the lowest (negative, such as "very hard" or "not confident") score was 1. Thus, higher scores reflected more positive attitudes toward the interfaces.

Q1: Is it easy or hard to learn how to use?
Q2: Is it easy or hard to specify the query with sequence only?
Q3: Is it easy or hard to specify the query with time constraint?
Q4: Is it easy or hard to specify the query with uncertainty?
Q5: Is it easy or hard to specify the query in the last task?
Q6: How confident is your answer for finding at least one, best answer tasks?
Q7: How confident is your answer for counting tasks?

4.4. Design

The independent variables were: Interface type (2 treatments): exact match and similarity search, Task (8 treatments)

The dependent variables were: The time to complete each task, error rate for each task, and subjective ratings on a 7-point Likert scale.

The controlled variables were: Computer, mouse and window size. We used equivalent datasets for each interface.

To control learning effects, the presentation order of the LifeLines2 and Similan2 interfaces was counterbalanced. To avoid the situations that the users would always find repeating the tasks on the second system easier, since they already know the answer, we also used two sets of questions and datasets, one for each interface. The questions in the two sets are different but have same difficulty level, for example: "Find at least one patient who was admitted, transferred to Floor then to ICU." and "Find at least one patient who was admitted, transferred to Floor then to IMC." Half of the participants started with LifeLines2 while another half started with Similan2.

4.5. Procedure

Participants were given training which included a brief description of the data and ten-minutes tutorials of how to the first interface. Then, the participants had to complete two training tasks. When the participants

could answer the training questions correctly, they were considered ready to perform the study tasks. Next, the participants were asked to perform eight tasks using the first interface. After that, the experimenter followed the same procedure (tutorial, training tasks, study tasks) for the second interface.

Upon completion of the tasks, the participants were asked to complete the 7-point Likert scale questionnaire.

At the end of the experiment, we debriefed the participants to learn about their experience while using the interfaces for the different tasks and their suggestions for improving the interfaces.

5. Results

5.1. Performance Time

To examine the effects of the type of interface and the task on time to perform tasks 1-5, we conducted a two-way ANOVA with repeated measures. The time to perform the task was the dependent variable and the type of interface and the task were within participants independent variables. The results of the analysis showed that the main effect of the task was significant ($F(4, 68) = 15.15$, $p<.001$). The two-way interaction (interface $\times$ task) was also significant ($F(4, 68) = 6.63$, $p<.001$). The main effect of the interface was not found to be significant ($F(1, 17) = 1.60$, $p=.22$).

Figure 8 shows the performance time as a function of the interface and the task. It can be seen that for tasks 1-3, the performance times using the two interfaces are very similar and increase for the tasks with time constraint (2) and uncertainty (3) (M$\pm$SD of 26.83$\pm$10.90 s, 39.58$\pm$23.92 s and 58.67$\pm$33.60 s, respectively). However, the average performance times of tasks 4 and 5 are shorter using the similarity search interface (M$\pm$SD of 51.73$\pm$13.21 s and 37.74$\pm$18.63 s, respectively) than while using the exact match interface (M$\pm$SD of 68.33$\pm$31.18 s and 72.05$\pm$34.41 s, respectively). It can also be observed that the variances in the performance time of tasks 2-5 are larger while using the exact match.

A post-hoc Duncan test showed that the performance times of tasks 4 and 5 are significantly shorter while using the similarity search interface ($p<.05$). When using the exact match, there were significant differences in performance time between two homogenous groups: tasks 1-2 versus tasks 3-5 ($p<.001$). When using the similarity search, the main significant differences in performance time were between task 3 to tasks 1 and 5 ($p<.05$).

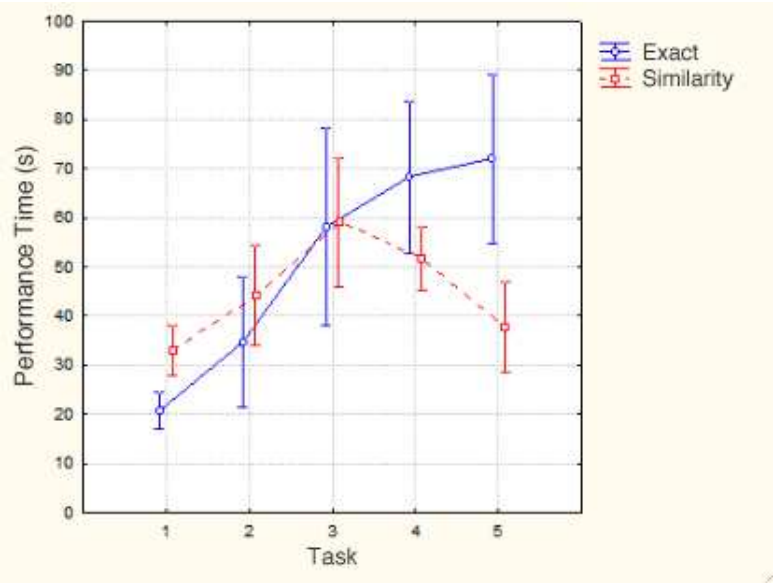

Figure 8: Performance time a function of the interface type and the tasks (1-5). Vertical bars denote 0.95 confidence intervals.

Similar analytic procedures were followed in the analysis of the effects of the interface type and the task on time to perform the counting tasks (1.2, 2.2 and 3.2). The results of the analysis showed that the only effect that was found to be significant was the main effect of the interface ($F(1, 17) = 23.65$, $p<.001$). The average performance time while interacting with the exact match was significantly shorter than with the similarity search (M±SD of 2.32±3.75 s and 15.20±25.10 s, respectively) The main effect of the task ($F(2, 34) = 2.05$, $p=.14$) and the interaction effect ($F(2, 34) = 2.03$, $p=.15$) were not found to be significant.

5.2. Error Rates

To compare the error rates between the two interfaces while performing the different tasks, we performed a McNemar's test, which is a non-parametric test that is used to compare two population proportions that are related or correlated to each other. Since the error rates of tasks 1-3 were zero for both interfaces, we conducted this analysis only for tasks 4 and 5 (4 and 2 incorrect answers using the exact match, respectively and no error while using the similarity search). The results of the analysis showed that there was no significant difference between the two interfaces in the error

Question	Average Rating ± SD		t(17)	p-value
	X	*S*		
Q1: Easy to learn	5.67 ± 1.37	5.44 ± 1.29	0.61	p=.55
Q2: Query for sequence only	6.89 ± 0.32*	5.50 ± 1.20	4.74	p<.001
Q3: Query with time constraint	4.94 ± 1.51	6.00 ± 1.19*	-2.82	p<.05
Q4: Query with uncertainty	4.11 ± 1.37	5.78 ± 1.06*	-5.15	p<.001
Q5: Query similar records	3.94 ± 0.43	6.78 ± 0.43*	-7.99	p<.001
Q6: Confidence-Find most similar	5.83 ± 0.99	5.78 ± 1.17	0.15	p=.88
Q7: Confidence-Count	6.72 ± 0.75*	4.83 ± 1.10	6.78	p<.001

Table 1: Results of the analysis of subjective ratings given by the participants to the two interfaces while performed the different tasks. "X" denotes exact match while "S" denotes similarity search. "*" indicates preferred interface.

rates of task 4 ($\chi^2(1)$=3.06,p=.08) and 5 ($\chi^2(1)$=1.13,p=.29).

5.3. Subjective Ratings

To compare the difference between the subjective ratings given by the participants to the two interfaces, we conducted a paired-sample t-test for each question. The results of the analysis are presented in Table 1. The results showed that there was no significant difference for the ease of learning how to use the two interfaces (Q1). The participants reported the exact match to be significantly easier to use than the similarity search for the task with sequence only (task 1) (Q2). However, for the tasks with only time constraint (task 2) (Q3) or only uncertainty constraint (task 3) (Q4), they reported the similarity search to be significantly easier to use than the exact match. They also reported the similarity search to be significantly easier to use than the exact match in the task that required them to find a patient which is the most similar to the given patient (Q5). There was no significant difference between the confidence levels of the answers for the tasks which required finding at least one, best answer (tasks 1-5) (Q6). However, the participants were significantly more confident while using the exact match than the similarity search to find the answers for the counting tasks (tasks 1.2, 2.2 and 3.2) (Q7).

5.4. Debriefing

When asked about what they liked in LifeLines2, the participants said that it is easy for finding a sequence ("Easy to find sequence", "Very easy

to query with sequence" "Very intuitive to specify sequence") and counting ("Show only matched records make it easy to count", "It gives confidence.")

However, when asked about what they did not like in LifeLines2, they explained that it is difficult for uncertain and more complex tasks because they had to change the query and sometimes, more than one filter is needed. ("It doesn't find the similar case when it can't find the case I want", "Difficult for complex tasks or tasks with uncertainty", "Hard to find approximate best match", "Harder in LifeLines2 because I had to change the query [for the uncertain task]", "In order to find a patient, sometimes more than one filter is needed.")

When asked about what they liked in Similan2, the participants said that it is more flexible and easier to find similar patients. ("Very easy to find the similar pattern.", "Similan is more flexible.", "The similarity measure makes it FAR easier to find the best matches.", "Excellent in finding 'at least one' type results when formulating the query is harder [in LifeLines2] due to ambiguity.") They also said that it is easier to specify the time constraints in a query and that specifying how the answers should look like makes the search process more transparent. ("Query with time constraint is very easy.", "Time constraint searches are easier to input.", "the search process is more transparent.", "Drag and drop triangles gave me better control of how the specific sequences should look like.")

However, when asked about what they did not like in Similan2, the participants expressed difficulty in using it for the counting tasks because it is difficult to separate between the exact match results and the similar results. ("No easy way to count" "not sure [whether] the top rows are the only answers") Also, sometimes it is unclear where to place the events on the timeline. ("In Similan2, it is not immediately obvious where to place the icon for the 'second day'.") Two participants also mentioned that similarity search responded slightly slower than exact match.

Common suggestions for improvement included: "LifeLines2 should have a list of previous actions and undo button", "A counter in Similan for all patients that had a match of X score or better could be helpful.", "Have individual weight for events in the query in Similan, so the users can specify if some events are more important than others.", "Have more weight presets to choose from."

6. Conclusions and Future Work

Event sequence data are continuously being gathered by various organizations. Querying for time-stamped event sequences in those data sets to answer questions or look for patterns is a very common and important activity. Most existing temporal query GUIs use an exact match approach, which returns only records that match the query. However, in exploratory search, users are often uncertain about what they are looking for. Too narrow queries may eliminate results which are on the borderline of the constraints. On the other hand, the similarity search approach allows users to sketch an example of what they are seeking and find similar results, which provides more flexibility.

This paper makes these contributions:

1. We presented Similan2, an interface to specify an event sequence example and search for records that are similar to the example, following the similarity search approach.
2. We introduced the M&M measure v.2 which is faster and can be customized by four decision criteria, increasing its performance and flexibility from the first version.
3. We conducted a controlled experiment that assessed the benefits of exact match and similarity search interfaces for five tasks, leading to future directions for improving event sequences query interfaces that combine the benefits of both interfaces.

Our experiment showed that exact match had advantages in finding exact results. Users preferred to use it to find a simple sequence without time constraint or uncertainty more than the similarity search. The exact match also gave more confidence to the users in tasks that involve counting. However, users felt that it was more complex to use for tasks with time constraints or uncertainty (probably because it required many steps to add each constraint to construct a query).

On the other hand, similarity search had advantages in the flexibility and intuitiveness of specifying the query for tasks with time constraints or uncertainty, or tasks that ask for records that are similar to a given record. Users felt that it is easier to specify the time constraints in a query and that specifying how the answers should look like makes the search process more transparent because they could see the big picture of their query. However,

similarity search was more difficult for tasks that involve counting. The participants requested a better way to support counting tasks.

Further work is needed to address these two points:

First, the exact match and similarity search interfaces each have their advantages. How can we combine the best features from these two interfaces to create a new interface that can support queries with uncertainty and time constraints as well as simpler and counting tasks? Based on the results of the experiment and our observations during the longitudinal study with our partners, we list several ideas for hybrid query interfaces that should be explored in the future:

1) Draw an example. Specifying the query by placing event glyphs on a timeline seems closer to users' natural problem-solving strategy and the visual representation of the query also helps users compare results with the query to notice and correct errors.

2) Sort results by similarity to the query but do not return all records and allow users to see more if necessary. Showing all records, even those that do not fit the query, confuses users and reduces confidence in the results. However, users may want to see more results at certain times. One possible strategy is to show only exact results first (i.e. like exact match) and have "more" button to show the rest or the next n records. Another strategy is to add a borderline that separates the exact results from the near matches. This may increase confidence and still be useful for exploratory search.

3) Allow users to specify what is flexible and what is not. Even in a query with uncertainty, users may have some parts of the query that they are certain about, e.g. patients must be admitted to the ICU (i.e. do not even bother showing me records with no ICU event). These certain rules can be applied strictly to narrow down the result set without sacrificing the flexibility of the other parts of the query.

4) Weights. Whether users would be able to translate more complex data analysis goals into proper weight settings remains an open issue. One idea to prevent manual weight adjustment is to provide presets of weights that capture common definitions of similarity.

5) Avoid too many alternative ways to perform the same task. This can lead to confusion. In the experiment, we found many users used more filters than necessary.

Second, while this paper focuses on medical examples, all design principles are based on event sequences, which are not specific to the medical domain. Therefore, we believe that these concepts are applicable to event

sequences in other domains, such as traffic incidents logs, student records, researchers' publication list, U.S. bill status, web logs, usability logs, criminal investigations, and many more topics. Part of our future work will include more user studies that show applications of these concepts to other domains.

We believe that querying event sequences will become an increasingly common and important task. The growing effort on query languages and similarity measures is helpful, but this paper advances research by developing interfaces and evaluating them in a rigorous way with five tasks.

7. Acknowledgement

We appreciate support from the National Institutes of Health (NIH) grant CA147489 and Washington Hospital Center, and collaboration from our physician partners at the Washington Hospital Center, especially Dr. Phuong Ho, Dr. Mark Smith and David Roseman, and would like to thank Dr. Vibha Sazawal, Dr. Jen Golbeck, Dr. Taowei David Wang and Sureyya Tarkan for their thoughtful comments, and all participants in the studies for their participations.

References

Aigner, W., Miksch, S., 2006. CareVis: integrated visualization of computerized protocols and temporal patient data. Artificial Intelligence in Medicine 37, 203–18.

Altschul, S., Gish, W., Miller, W., Myers, E., Lipman, D., 1990. Basic local alignment search tool. Journal of Molecular Biology 215, 403–410.

André-Jönsson, H., Badal, D.Z., 1997. Using signature files for querying time-series data.

Bederson, B., Grosjean, J., Meyer, J., 2004. Toolkit design for interactive structured graphics. IEEE Trans. Software Engineering 30, 535–546.

Berndt, D.J., Clifford, J., 1994. Using dynamic time warping to find patterns in time series, in: AAAI-94 Workshop on Knowledge Discovery in Databases, pp. 229–248.

Bonhomme, C., Aufaure, M.A., 2002. Mixing icons, geometric shapes and temporal axis to propose a visual tool for querying spatio-temporal databases, in: Proc. Working Conf. on Advanced Visual Interfaces (AVI), ACM. pp. 282–289.

Bonhomme, C., Trépied, C., Aufaure, M.A., Laurini, R., 1999. A visual language for querying spatio-temporal databases, in: Proc. ACM International Symp. on Advances in Geographic Information Systems (GIS), ACM. pp. 34–39.

Carenini, G., Loyd, J., 2004. ValueCharts: analyzing linear models expressing preferences and evaluations, in: Proc. Working Conf. on Advanced Visual Interfaces (AVI), ACM. pp. 150–157.

Chang, N., Fu, K., 1980. Query-by-Pictorial-Example. IEEE Trans. Software Engineering 6, 519–524.

Chang, R., Ghoniem, M., Kosara, R., Ribarsky, W., Yang, J., Suma, E., Ziemkiewicz, C., Kern, D., Sudjianto, A., 2007. WireVis: Visualization of Categorical, Time-Varying Data From Financial Transactions, in: Proc. IEEE Symp. on Visual Analytics Science and Technology (VAST), IEEE. pp. 155–162.

Chomicki, J., 1994. Temporal query languages: A survey, in: Proc. International Conf. on Temporal Logic, Springer. pp. 506–534.

Clifford, J., Croker, A., 1987. The Historical Relational Data Model (HRDM) and Algebra Based on Lifespans, in: Proc. IEEE International Conf. on Data Engineering (ICDE), IEEE. pp. 528–537.

Dobrisek, S., Zibert, J., Pavesić, N., Mihelic, F., 2009. An edit-distance model for the approximate matching of timed strings. IEEE Trans. pattern analysis and machine intelligence 31, 736–41.

Fails, J., Karlson, A., Shahamat, L., Shneiderman, B., 2006. A Visual Interface for Multivariate Temporal Data: Finding Patterns of Events across Multiple Histories, in: Proc. IEEE Symp. on Visual Analytics Science and Technology (VAST), IEEE. pp. 167–174.

Gómez-Alonso, C., Valls, A., 2008. A Similarity Measure for Sequences of Categorical Data Based on the Ordering of Common Elements, in: Torra,

V., Narukawa, Y. (Eds.), Modeling Decisions for Artificial Intelligence. Springer. 1. chapter 13, pp. 134–145.

Hamming, R.W., 1950. Error Detecting and Error Correcting Codes. The Bell System Technical Journal 29, 147–160.

Hibino, S., Rundensteiner, E., 1995. A visual query language for identifying temporal trends in video data, in: Proc. International Workshop on Multi-Media Database Management Systems, IEEE. pp. 74–81.

Hibino, S., Rundensteiner, E.A., 1997. User interface evaluation of a direct manipulation temporal visual query language, in: Proc. ACM International Conf. on Multimedia (MULTIMEDIA), ACM. pp. 99–107.

Hochheiser, H., Shneiderman, B., 2004. Dynamic query tools for time series data sets: Timebox widgets for interactive exploration. Information Visualization 3, 1–18.

Jacobs, B., Walczak, C., 1983. A Generalized Query-by-Example Data Manipulation Language Based on Database Logic. IEEE Trans. Software Engineering SE-9, 40–57.

Jin, J., Szekely, P., 2009. QueryMarvel: A visual query language for temporal patterns using comic strips, in: Proc. IEEE Symp. on Visual Languages and Human-Centric Computing (VL/HCC). IEEE. pp. 207–214.

Karam, G., 1994. Visualization using timelines, in: Proc. ACM SIGSOFT International Symp. on Software Testing and Analysis, ACM. pp. 125–137.

Kato, T., Kurita, T., Otsu, N., Hirata, K., 1992. A sketch retrieval method for full color image database-query by visual example, in: Proc. IAPR International Conf. on Pattern Recognition, IEEE. pp. 530–533.

Klimov, D., Shahar, Y., Taieb-Maimon, M., 2009. Intelligent selection and retrieval of multiple time-oriented records. Journal of Intelligent Information Systems 35, 261–300.

Klimov, D., Shahar, Y., Taieb-Maimon, M., 2010. Intelligent visualization and exploration of time-oriented data of multiple patients. Artificial Intelligence in Medicine 49, 11–31.

Klug, A.C., 1981. Abe: a query language for constructing aggregates-by-example, in: Proc. LBL Workshop on Statistical Database Management (SSDBM), Lawrence Berkeley Lab. pp. 190–205.

Kuhn, H.W., 1955. The Hungarian method for the assignment problem. Naval Research Logistics Quarterly 2, 83–97.

Levenshtein, V.I., 1966. Binary codes capable of correcting deletions, insertions and reversals. Soviet Physics Doklady 10, 707–710.

Li, W.S., Candan, K.S., Hirata, K., Hara, Y., 1997. IFQ: a visual query interface and query generator for object-based media retrieval, in: Proc. IEEE International Conf. on Multimedia Computing and Systems, IEEE. pp. 353–361.

Mannila, H., Moen, P., 1999. Similarity between Event Types in Sequences, in: Proc. International Conf. on Data Warehousing and Knowledge Discovery (DaWaK), Springer. pp. 271–280.

Mannila, H., Ronkainen, P., 1997. Similarity of Event Sequence, in: Proc. International Workshop on Temporal Representation and Reasoning (TIME), pp. 136–139.

Mannila, H., Seppänen, J., 2001. Finding similar situations in sequences of events via random projections, in: Proc. SIAM International Conf. on Data Mining, Citeseer. pp. 1–16.

Mongeau, M., Sankoff, D., 1990. Comparison of Musical Sequences. Computer and the Humanities 24, 161–175.

Munkres, J., 1957. Algorithms for the assignment and transportation problems. Journal of the Society for Industrial and Applied Mathematics 5, 32–38.

Navarro, G., 2001. A guided tour to approximate string matching. ACM Computing Surveys 33, 31–88.

Obweger, H., Suntinger, M., Schiefer, J., Raidl, G., 2010. Similarity searching in sequences of complex events, in: Proc. International Conf. on Research Challenges in Information Science (RCIS), IEEE. pp. 631–640.

Ozsoyoglu, G., Matos, V., Ozsoyoglu, M., 1989. Query processing techniques in the summary-table-by-example database query language. ACM Trans. Database Systems 14, 526–573.

Ozsoyoglu, G., Wang, H., 1993. Example-based graphical database query languages. Computer 26, 25–38.

Pearson, W.R., Lipman, D.J., 1988. Improved tools for biological sequence comparison., in: Proc. National Academy of Sciences of the United States of America, pp. 2444–2448.

Rigoutsos, I., Floratos, a., 1998. Combinatorial pattern discovery in biological sequences: The TEIRESIAS algorithm. Bioinformatics 14, 55–67.

Shahar, Y., Goren-Bar, D., Boaz, D., Tahan, G., 2006. Distributed, intelligent, interactive visualization and exploration of time-oriented clinical data and their abstractions. Artificial Intelligence in Medicine 38, 115–35.

Sherkat, R., Rafiei, D., 2006. Efficiently evaluating order preserving similarity queries over historical market-basket data, in: Proc. International Conf. on Data Engineering (ICDE), pp. 19–30.

Shneiderman, B., 1983. Direct Manipulation: A Step Beyond Programming Languages. Computer 16, 57–69.

Shneiderman, B., Plaisant, C., 2006. Strategies for evaluating information visualization tools, in: Proc. AVI workshop on BEyond time and errors novel evaluation methods for information visualization (BELIV), ACM. pp. 1–7.

Snodgrass, R., 1987. The temporal query language TQuel. ACM Trans. Database Systems 12, 247–298.

Snodgrass, R.T., 1995. The TSQL2 temporal query language. Kluwer Academic Publishers.

Tansel, A., Arkun, M., Ozsoyoglu, G., 1989. Time-by-example query language for historical databases. IEEE Trans. Software Engineering 15, 464–478.

Tansel, A., Tin, E., 1997. The expressive power of temporal relational query languages. IEEE Trans. Knowledge and Data Engineering 9, 120–134.

Tukey, J.W., 1977. Exploratory Data Analysis. Addison-Wesley.

Vrotsou, K., 2010. Everyday mining Exploring sequences in event-based data. Ph.D. thesis. Linkoping University.

Vrotsou, K., Forsell, C., 2011. A Qualitative Study of Similarity Measures in Event-Based Data, in: Proc. Human Interface and the Management of Information. Interacting with Information Symp. on Human Interface, Springer. pp. 170–179.

Vrotsou, K., Johansson, J., Cooper, M., 2009. ActiviTree: interactive visual exploration of sequences in event-based data using graph similarity. IEEE Trans. Visualization and Computer Graphics 15, 945–52.

Wang, T.D., Plaisant, C., Quinn, A.J., Stanchak, R., Murphy, S., Shneiderman, B., 2008. Aligning temporal data by sentinel events: discovering patterns in electronic health records, in: Proc. Annual SIGCHI Conf. on Human Factors in Computing Systems (CHI), ACM. pp. 457–466.

Wang, T.D., Plaisant, C., Shneiderman, B., Spring, N., Roseman, D., Marchand, G., Mukherjee, V., Smith, M., 2009. Temporal Summaries: Supporting Temporal Categorical Searching, Aggregation and Comparison. IEEE Trans. Visualization and Computer Graphics 15, 1049–1056.

Watai, Y., Yamasaki, T., Aizawa, K., 2007. View-Based Web Page Retrieval using Interactive Sketch Query, in: Proc. IEEE International Conf. on Image Processing, IEEE. pp. 357–360.

Wattenberg, M., 2001. Sketching a graph to query a time-series database, in: Proc. Annual SIGCHI Conf. on Human Factors in Computing Systems (CHI) - Extended Abstracts, ACM. pp. 381–382.

White, R.W., Roth, R.A., 2009. Exploratory Search: Beyond the Query-Response Paradigm, in: Synthesis Lectures on Information Concepts, Retrieval, and Services, pp. 1–98.

Winkler, W.E., 1999. The state of record linkage and current research problems. Technical Report. Statistical Research Division, U.S. Census Bureau.

Wongsuphasawat, K., Shneiderman, B., 2009. Finding comparable temporal categorical records: A similarity measure with an interactive visualization, in: Proc. IEEE Symp. on Visual Analytics Science and Technology (VAST), IEEE. pp. 27–34.

Zloof, M., 1982. Office-by-Example: A business language that unifies data and word processing and electronic mail. IBM Systems Journal 21, 272–304.

Zloof, M.M., 1975. Query by example, in: Proc. National Computer Conf. and Exposition (AFIPS), ACM. pp. 431–438.