ABSTRACT

Title of dissertation: INTERACTIVE EXPLORATION OF TEMPORAL EVENT SEQUENCES
Krist Wongsuphasawat
Doctor of Philosophy, 2012

Dissertation directed by: Professor Ben Shneiderman
Department of Computer Science

Life can often be described as a series of *events*. These events contain rich information that, when put together, can reveal history, expose facts, or lead to discoveries. Therefore, many leading organizations are increasingly collecting databases of *event sequences*: Electronic Medical Records (EMRs), transportation incident logs, student progress reports, web logs, sports logs, etc. Heavy investments were made in data collection and storage, but difficulties still arise when it comes to making use of the collected data. Analyzing millions of event sequences is a non-trivial task that is gaining more attention and requires better support due to its complex nature. Therefore, I aimed to use information visualization techniques to support *exploratory data analysis*—an approach to analyzing data to formulate hypotheses worth testing—for event sequences. By working with the domain experts who were analyzing event sequences, I identified two important scenarios that guided my dissertation:

First, I explored how to provide an overview of multiple event sequences? Lengthy reports often have an executive summary to provide an overview of the

report. Unfortunately, there was no executive summary to provide an overview for event sequences. Therefore, I designed *LifeFlow*, a compact overview visualization that summarizes multiple event sequences, and interaction techniques that supports users' exploration.

Second, I examined how to support users in querying for event sequences when they are uncertain about what they are looking for. To support this task, I developed similarity measures (*the M&M measure 1-2*) and user interfaces (*Similan 1-2*) for querying event sequences based on similarity, allowing users to search for event sequences that are similar to the query. After that, I ran a controlled experiment comparing exact match and similarity search interfaces, and learned the advantages and disadvantages of both interfaces. These lessons learned inspired me to develop *Flexible Temporal Search (FTS)* that combines the benefits of both interfaces. FTS gives confident and countable results, and also ranks results by similarity.

I continued to work with domain experts as partners, getting them involved in the iterative design, and constantly using their feedback to guide my research directions. As the research progressed, several short-term user studies were conducted to evaluate particular features of the user interfaces. Both quantitative and qualitative results were reported. To address the limitations of short-term evaluations, I included several multi-dimensional in-depth long-term case studies with domain experts in various fields to evaluate deeper benefits, validate generalizability of the ideas, and demonstrate practicability of this research in non-laboratory environments. The experience from these long-term studies was combined into a set of design guidelines for temporal event sequence exploration.

My contributions from this research are *LifeFlow*, a visualization that compactly displays summaries of multiple event sequences, along with interaction techniques for users' explorations; similarity measures (*the M&M measure 1-2*) and similarity search interfaces (*Similan 1-2*) for querying event sequences; *Flexible Temporal Search (FTS)*, a hybrid query approach that combines the benefits of exact match and similarity search; and case study evaluations that results in a process model and a set of design guidelines for temporal event sequence exploration. Finally, this research has revealed new directions for exploring event sequences.

INTERACTIVE EXPLORATION OF TEMPORAL EVENT SEQUENCES

by

Krist Wongsuphasawat

Dissertation submitted to the Faculty of the Graduate School of the
University of Maryland, College Park in partial fulfillment
of the requirements for the degree of
Doctor of Philosophy
2012

Advisory Committee:
Dr. Ben Shneiderman, Chair/Advisor
Dr. Catherine Plaisant
Dr. Amol Deshpande
Dr. Lise Getoor
Dr. David Gotz
Dr. Jeffrey Herrmann

Dedication

To the Wongsuphasawat Family

Acknowledgments

Doing a PhD is a long, complicated and devoted journey. Of course, there were good, fun, inspiring and exciting moments; otherwise, I would not have survived long enough to be sitting here writing this dissertation. However, there were also many challenges and obstacles, which I probably would not be able to go through without the advice, inspiration and support from these people, whom I deeply appreciate.

First and foremost, I would like to thank my advisor, Ben Shneiderman. Ben introduced me to the world of information visualization and inspired me to continue for a PhD after completing my Master's degree. He is a truly devoted professor who constantly inspires people around him with his passion and energy. His tremendous work in HCI is nothing short of amazing. His warmth, encouragement, creativity, depth and breadth of knowledge, and great vision had helped me greatly throughout my PhD training. He taught me how to be a good researcher and also showed how to be a good colleague and a good person. He always supported me when I was in doubt and celebrated my success with his signature "Bravo!" I am grateful and feel very fortunate to have a chance to work with and learn from him.

The second person that I would like to thank is Catherine Plaisant. By working with a great researcher like her, I have learned many design principles and valuable lessons about how to work with users. In addition, her enthusiasm and energy were like triple shot espressos that could cheer me up even on a depressing rainy day. I could not remember how many times I walked to her room (which was conveniently located right in front of my cubicle) and asked for help or suggestions. She always

helped me with smiles no matter how big or small the problem was. The work in this dissertation could not have been completed without her invaluable feedback and suggestions. Similar to Ben, it was my honor and great pleasure to work with and learn from Catherine.

I am thankful to Lise Getoor, Amol Deshpande, Jeffrey Herrmann and David Gotz for agreeing to serve on my dissertation committee, for sparing their invaluable time reviewing the manuscript, and for sharing their thoughtful suggestions that help me improve this dissertation. Additionally, I would like to thank David Gotz for offering me an internship at IBM, which was a great experience that led to the spin-off Outflow project. I am also thankful for the guidance from Samir Khuller when I started working on Similan.

I owe my thanks to Taowei David Wang who had helped me start my PhD student life and set many good examples for me to follow. Taking care of his LifeLines2 while he was away to a summer internship also led me to many ideas that later became LifeFlow. I appreciate his generous help and suggestions during the time I was finding my dissertation topic and preparing my proposal. I appreciate John Alexis Guerra Gómez's help in the development of LifeFlow and thank him for taking care of LifeFlow while I was away to a summer internship. Many new datasets were made available for analysis in the late stage of my dissertation thanks to Hsueh-Chien Cheng, who developed DataKitchen and made data preprocessing a much more pleasurable experience.

My colleagues in the Human-Computer Interaction Lab have made my PhD life such a great memory. I have enjoyed every moment of our discussion, collabora-

tion (and procrastination). I will always remember the brown bag lunches (including the free pizzas), normal lunches (which people often found me with a bag of Chick-fil-A), annual symposiums (and the green t-shirts), helpful practice talks, fruitful brainstorming sessions (aka: productive-procrastination), yummy international cuisine tasting, relaxing late-night hangouts in the lab, entertaining discussions in the cubicles, and all other memorable activities.

I appreciate the support from Mark Smith and his team from the Washington Hospital, MedStar Health and MedStar Institute for Innovation, especially Phuong Ho and A. Zach Hettinger. Without the inspirations from their medical problems, none of the projects discussed herein would have materialized. In addition, I would like to acknowledge financial support from the National Institutes of Health (NIH) Grant RC1CA147489 and Center for Integrated Transportation Systems Management (CITSM), a tier 1 transportation center at the University of Maryland.

Many thanks to the user study participants: Phuong Ho, Nikola Ivanov, Michael VanDaniker, Michael L. Pack, Sigfried Gold, A. Zach Hettinger, Jae-wook Ahn, Daniel Lertpratchya, Chanin Chanma, Sorawish Dhanaphanichakul and Anne Rose, as well as the anonymized participants in other studies. Their precious feedbacks have contributed greatly to my research.

Studying abroad and living away from the place that I had been living for my entire life was never easy. I owe my thanks to the Thai communities in Maryland and DC area for making this place feel like a home away from home. Thank you for inviting me to dinners and fun activities on weekends. I am also thankful to my friends, no matter where you are, who chatted with me when I was bored, frustrated,

or just sleepy on Monday morning. Thank you for keeping my sanity in check and reminding me every once in a while that the dissertation is only a part of my life, not my entire life.

Words cannot express the gratitude I owe my family. I would not be the person I am today without the nurture of my parents, grandparents and aunt. My family always stood by me when I questioned myself on my quest to earn a PhD. Their support and guidance gave me the strengths to overcome all challenges and obstacles. They celebrated my achievements, made fun of my photos on Facebook at certain times, and embraced me with love every time I went home. I also have to thank my lovely Ben for supporting me to follow my dreams, having faith in me and adding unforgettable memories every time I returned.

Remembering everybody is a challenging task. I have tried to acknowledge as many people as I could remember, but if I have inadvertently left anyone out, I sincerely apologize.

While I am about to finish writing this dissertation in the next paragraph, I would like to thank the person who created this LaTeX template for making my life easier, and PhDComics for making my life funnier.

Lastly, my gratitude is described in Algorithm 1.

Algorithm 1 Infinite Gratitude

Require: a lot of water

```
1: for int i=0; i<555; i=i+0 do
2:     say("Thank you.");
3: end for
```

Table of Contents

List of Figures

List of Abbreviations

ED	Emergency Department
EHR	Electronic Health Record
EMR	Electronic Medical Record
EPL	English Premier League
ER	Emergency Room
FTS	Flexible Temporal Search
HCIL	Human-Computer Interaction Lab
IBM	International Business Machine
ICU	Intensive Care Unit
ICDL	Internation Children's Digital Library
IMC	Intermediate Medical Care
Man U	Manchester United Football Club
M&M	Match and Mismatch
MILCs	Multi-dimensional In-depth Long-term Case Studies
MVC	Model-View-Controller
NIH	National Institute of Health
RPG	Role-Playing Game
TC	Temporal Categorical
UCL	UEFA Champions League
UM	User Model
WHC	Washington Hospital Center

Chapter 1

Introduction

"Every moment and every event of every man's life on earth plants something in his soul."

–Thomas Merton

Our lives are defined by *events*, those tiny yet powerful pieces of information. From the day that I started my graduate program `(August 2007, enter graduate school)` to the day that I received my doctoral degree `(May 2012, graduation)`, many events had occurred in-between. Everyday, I woke up `(8:00 a.m., wake up)`, had breakfast `(8:15 a.m., breakfast)`, took a shower `(9:00 a.m., shower)`, then I went to campus `(10:00 a.m., work)`. Some days I had a meeting `(1:00 p.m., meeting)` or an important soccer game that I needed to watch `(2:45 p.m., soccer)`. While I was working in the lab, some patients might visit the hospital `(August 10, 2011 6:11 a.m., arrival)`, get diagnosed by the physicians `(August 10, 2011 6:30 a.m., emergency room)` and have to stay in the ICU `(August 10, 2011 9:05 a.m., ICU)` for several days `(August 15, 2011 3:00 p.m., discharged)`.

Each *event*, or *temporal event*, contains a *timestamp* `(8:00 a.m.)`, its *event type* `(wake up)` and additional information that make each event unique and have a story of its own. However, that is only a small part of its potential. The greater potential of these events is revealed when they are grouped and connected into *event*

sequences, which connect the dots between the scattered events and transform them into a more complete story. Here are some examples of event sequences.

1) *Student record*:

`(August'07, enter PhD program)` → `(April'10, propose)` → `(May'12, graduate)`

2) *Human activity*:

`(8:00 a.m., wake up)` → `(8:15 a.m., breakfast)` → `(9:00 a.m., shower)` → ...

3) *Medical record*:

`(6:11 a.m., arrival)` → `(6:30 a.m., emergency room)` → `(9:05 a.m., ICU)` → ...

4) *Soccer match*:

`(1st min, team A goal)` → `(89th min, team B goal)` → `(90th min, team B goal)`

An increasing number of leading organizations are collecting event sequence data. Health organizations have Electronic Medical Record (EMR) databases containing millions of records of patient histories. Each patient history contains hospital admissions, patients' symptoms, treatments and other medical events. Transportation systems generate logs of incidents and the timing of their management (first report, notifications and arrivals of each unit on the scene). Academic institutions keep detailed records of the educational advancement of their students (classes, milestones reached, graduation, etc.). Web logs, financial histories, market baskets and data in many other domains can also be viewed as event sequences. (More examples of event sequences in various domains are listed in Appendix A.)

However, these vast collections of data are not being fully utilized due to the limited approaches available for analyzing event sequences. Much effort has been put into designing data storage and information retrieval, but the exploratory

data analysis [124]—an approach to analyzing data for the purpose of formulating hypotheses worth testing—is still insufficiently supported.

1.1 Overview of the Dissertation

To help the users harvest from the diligently collected, rich and informative but gigantic databases, my dissertation aims to use *information visualization* techniques to support exploratory data analysis of event sequences. The fundamental concept of information visualization is that "visual representations and interaction techniques take advantage of the human eye's broad bandwidth pathway into the mind to allow users to see, explore, and understand large amounts of information at once" [123]. Therefore, I believe that by designing the suitable visual representations and interaction techniques for exploring event sequences, I can help users understand and gain new insight from the data.

This dissertation work was initially inspired by working closely with the physicians at the Washington Hospital Center. While helping them analyzing patient transfers sequences from Electronic Medical Records, I came across two important tasks that are not supported by the current systems and remain open problems in the research community. These two tasks are significant not only for analyzing patient transfers, but also in other event sequences. Seeing the opportunities to make an impact both theoretically and practically, I defined these two research questions to drive my research directions.

1. **How to provide an overview of multiple event sequences?**

   Although many systems were developed to visualize event sequences. [98, 8, 28, 97, 37, 131, 129], how to visualize an overview of multiple event sequences still remains a challenging problem. Previous systems can answer questions regarding the number of records that include a specific event sequence (e.g., "How many patients went from the Emergency Room to the Intensive Care Unit (ICU)?") but questions requiring an overview are not adequately supported. For example, a question such as "What are the most common transfer patterns between services within the hospital?" requires examination of all records one by one. Being unable to see all records on the screen at once makes it difficult to spot any pattern. Providing an overview is significant because it gives users a big picture of the underlying data, in the same way that an abstract does for a scholarly paper and an executive summary does for a marketing report. Squeezing a billion records into a million pixels [113] is also a great challenge in information visualization.

   Therefore, I have developed a novel interactive visual overview of event sequences called *LifeFlow*. This approach aggregates and compresses information from multiple event sequences into a data structure called *tree of sequences* whose size depends on length and total number patterns, thus reducing amount of information greatly. This data structure is later converted into a *LifeFlow* visualization that can display all possible sequences of events and the temporal spacing of the events within sequences.

2. **How to support users in querying for event sequences when they are uncertain about what they are looking for?**

   Many traditional tools use an *exact match* approach. Once the query is submitted, the queries are transformed into constraints and the tools return only the records that match the constraints. For example, users may want to find the records that have event "A" followed by event "B" within exactly 2 days, so they specify the query as `A`→≤`2 days`→`B`. This approach is effective when the users know exactly what they want.

   However, sometimes the users are uncertain about what they are looking for. For example, the physician wants to find the records that have event `Surgery` followed by event `Die` within approximately 2 days. The value "2 days" is just an approximation. Specifying narrow queries (e.g., `Surgery`→≤`2 days`→`Die`) could miss the records that might be "just off" (e.g., a patient who died 2 days and 1 minute after surgery). Using broad queries (e.g., `Surgery`→`Die`) could return too many results that are not relevant. Query methods that search for similar records rather than finding exact matches could better support the exploratory searchers.

   Therefore, I have developed methods for searching event sequences based on similarity. Since then, I have learned from the user studies that using similarity alone also has its limitations, so I have revised and developed *Flexible Temporal Search*, a hybrid approach between exact and similarity search for event sequences.

This dissertation describes how I approached these two problems in detail and include several user studies, ranging from a usability study, controlled experiments to multi-dimensional in-depth long-term case studies (MILCs) [114] that prove the benefits of my approaches. Although this research was initially inspired by applications in the medical domain, I have designed generalizable solutions that are not domain-specific, and therefore make them applicable to other event sequences outside of the medical domain, as shown in the case studies.

1.2 Contributions

My contributions from this research are:

1. A *tree of sequences*, a data structure that aggregates multiple event sequences while preserving temporal and sequential aspects of the data, and a visual representation, called *LifeFlow*, which compactly displays summaries of multiple event sequences, along with interaction techniques that facilitate the users' explorations. (Figure 1.1)

2. Similarity measures (the M&M measure 1-2) and similarity search interfaces (Similan 1-2) for querying event sequences. (Figure 1.2)

3. A similarity measure and user interface for *Flexible Temporal Search (FTS)*, a hybrid query approach for querying event sequences that combines the benefits of exact match and similarity search. (Figure 1.3)

4. Case study evaluations to refine the concept and user interfaces resulting in

Concept	Language	Lines of Code	Evaluation
Similan 1	C#	8,138	Usability Study
Similan 2	Adobe Flex	11,629	Controlled Experiment
LifeFlow	Java	61,188	Usability Study, MILCs
FTS	Java		MILCs

Table 1.1: Summary of all software developed in this dissertation

a process model and a set of design recommendations for temporal event sequence exploration.

A summary of all software developed in this dissertation and evaluation results are shown in Table 1.1 and 1.2, respectively.

1.3 Dissertation Organization

The remainder of this dissertation is organized as follows: Chapter 2 provides a literature review of background and related work; Chapter 3 describes *LifeFlow*, an overview visualization for event sequences; Chapter 4 explains *Similan* and the *M&M measure*, a user interface and similarity measure for querying event sequences by similarity; Chapter 5 then discusses *Flexible Temporal Search*, a hybrid interface for querying event sequences that combines the benefits of exact match and similarity search interfaces; Chapter 6 reports on several multi-dimensional in-depth long-term case studies that demonstrate the applications of my research in several domains; Finally, I give concluding remarks and discuss future work in Chapter 7.

Concept	Evaluation	Results
Similan 1	Usability Study	A study with eight participants was conducted. The participants believed that Similan could help them find students who were similar to the target student. (Section 4.3)
Similan 2	Controlled Experiment	A controlled experiment that compared exact match and similarity search interfaces showed that an exact match interface had advantages in finding exact results and also gave more confidence to the users in tasks that involve counting. On the other hand, similarity search interface 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. (Section 4.5)
LifeFlow	Usability Study	A study with ten participants confirmed that even novice users with 15 minutes of training were able to learn to use LifeFlow and rapidly answer questions about the prevalence of interesting sequences, find anomalies, and gain significant insight from the data. (Section 3.6)
LifeFlow	MILCs	Eight long-term case studies in six application domains demonstrated the benefits of LifeFlow in exploring event sequences and identifying interesting patterns. Many interesting use cases and findings were reported. (Chapter 6)
FTS	MILCs	Two long-term case studies in two application domains were conducted. FTS was used to search for particular situations when needed. Having the similar results give users more confidence when the exact match results is empty. (Section 6.5 and 6.9)

Table 1.2: Summary of all evaluation conducted in this dissertation

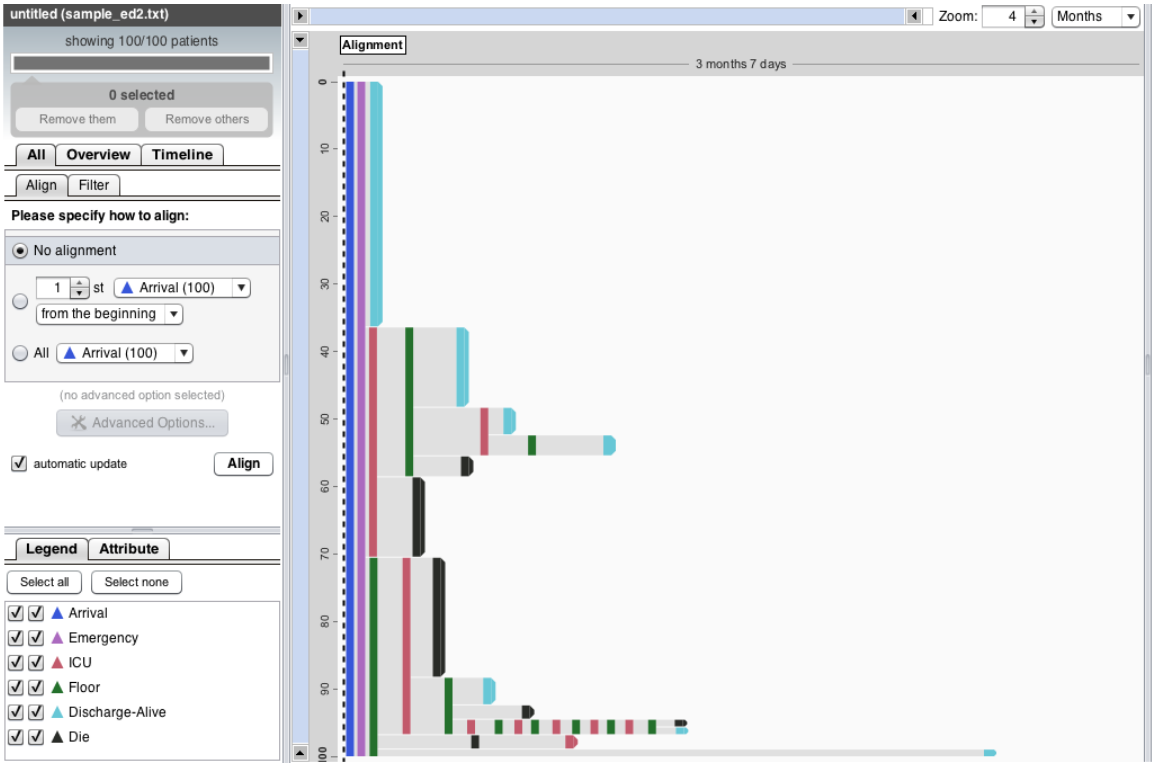

Figure 1.1: *LifeFlow*: An overview visualization for event sequences

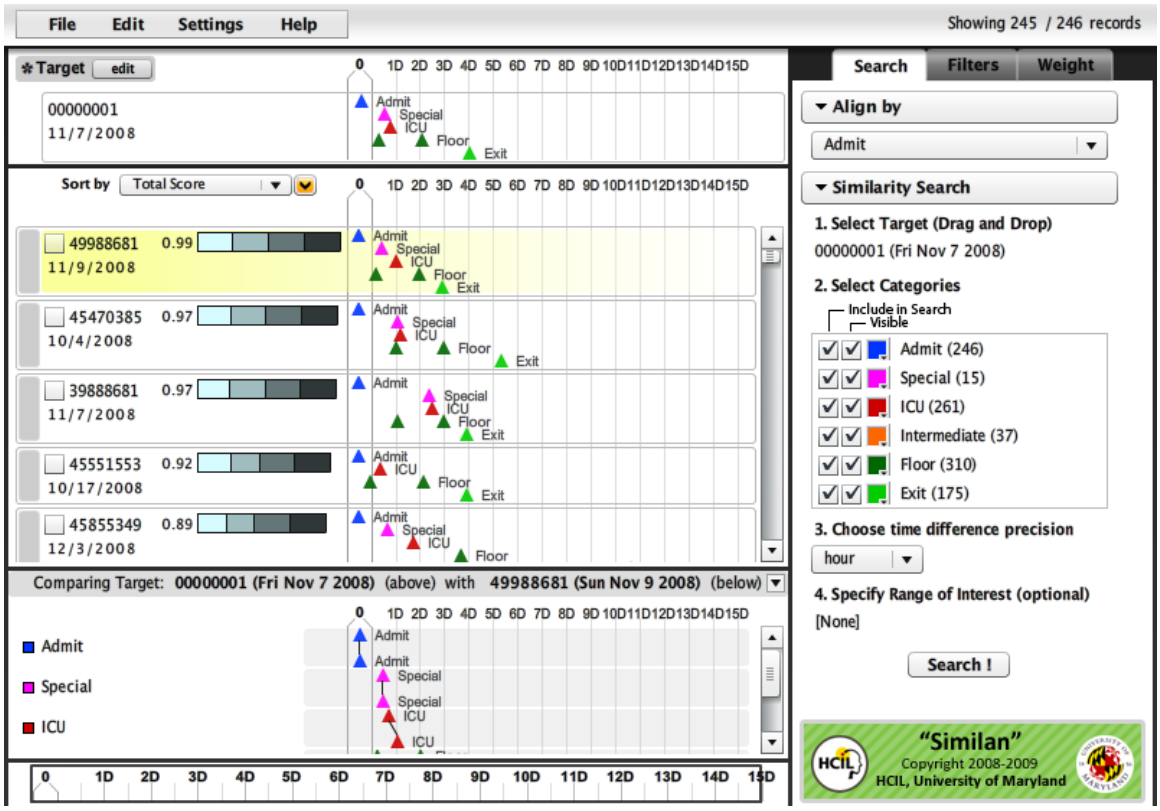

Figure 1.2: *Similan 2*: Query event sequences by similarity

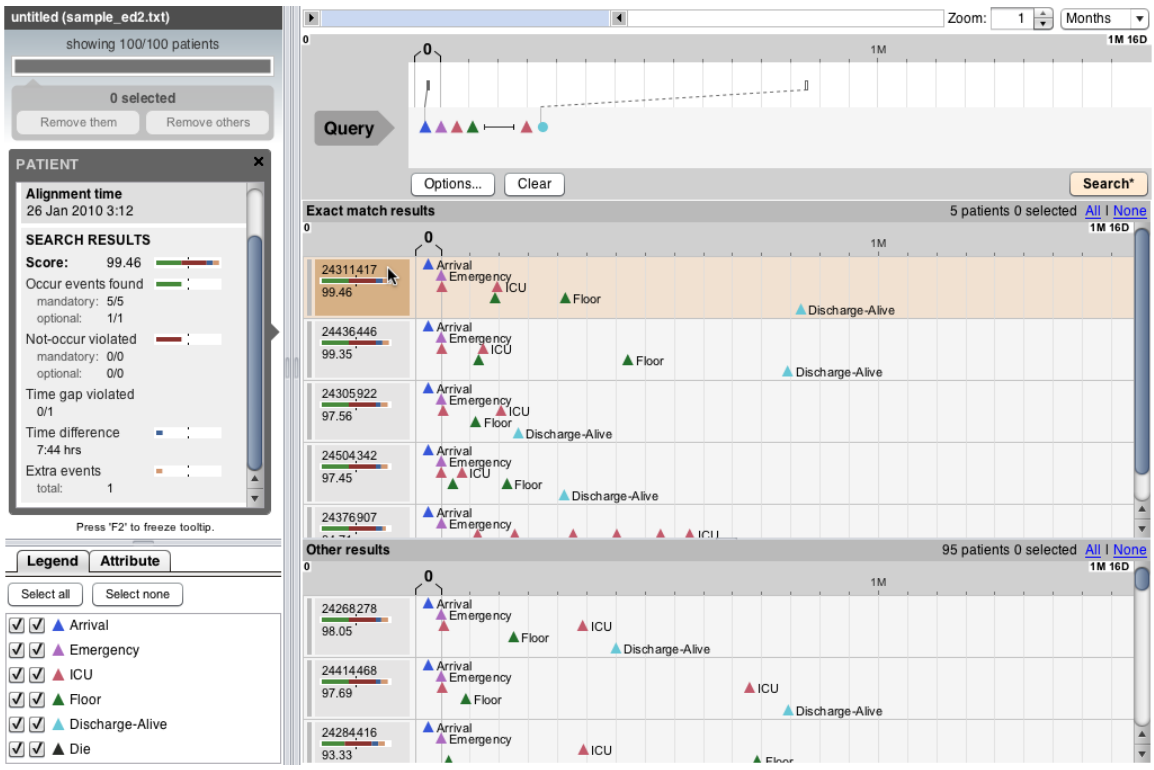

Figure 1.3: *Flexible Temporal Search (FTS)*: A hybrid approach that combines the benefits of exact match and similarity search for event sequences

Chapter 2

Background and Related Work

Time is one of the oldest data types that mankind has ever collected. Artifacts from the Palaeolithic suggest that the moon was used to reckon time as early as 6,000 years ago [105]. It would be interesting to review the literature of temporal data back that far, but for the sake of brevity, I select only a number of topics that are relevant to my research. In this chapter, I review the important techniques used for visualizing and querying temporal data, especially event sequences, and some other related areas to provide readers with the context of my dissertation work.

2.1 Information Visualization

2.1.1 Temporal Data Visualization

There has been a long history of visualizing temporal data [6]. I first review tools that support analysis of a single record, then tools designed to handle collections of records.

2.1.1.1 Single-Record Visualization

Many systems were designed for analyzing a single record [32, 46, 59, 97, 13, 7]. The most common approach is to use a timeline-based representation. Events are placed on a horizontal timeline according to their time. One record consists of

multiple rows, one for each category of events.

Cousins *et al.* [32, 33] developed the *Timeline Browser* for visualizing diabetes data of a single patient. The Timeline Browser has one special row in which the vertical position is used to indicate the value of blood glucose concentration readings. Other information, such as insulin doses, meals, and clinical studies are placed on different rows. It also provides zooming, filtering and details-on-demand. Harrison *et al.*'s *Timelines* [46] is an interactive system for collection and visualization of video annotation data. The visualization plots the categories (events and intervals) on the y-axis and time along the x-axis. Zooming, scrolling and reordering events are supported. Karam [59] introduced *xtg (Timeline Display Generator for X-windows)* for visualizing activities in client-server application—a video conference system. Xtg supports multiple views and zooming with details-on-demand. It also allows annotation and searching for events in the timeline. *LifeLines* (Figure 2.1) was developed by Plaisant *et al.* [97] to provide a general visualization environment for personal histories, which was applied to clinical patient records. Problems, diagnoses, test results or medications are presented as dots or horizontal lines. Colors can be used to indicate severity or type. Bade *et al.* [13] presented *MIDGAARD*, a timeline-based visualization, which enables the users to reveal the data at several levels of detail and abstraction, ranging from a broad overview to the fine structure. Resizing the height of the visualization adjusts the abstraction levels. *PlanningLines* [7], a visualization used for representing temporal uncertainties, was developed to support project management. The glyph consists of two encapsulated bars, representing minimum and maximum duration, that are bounded by two caps

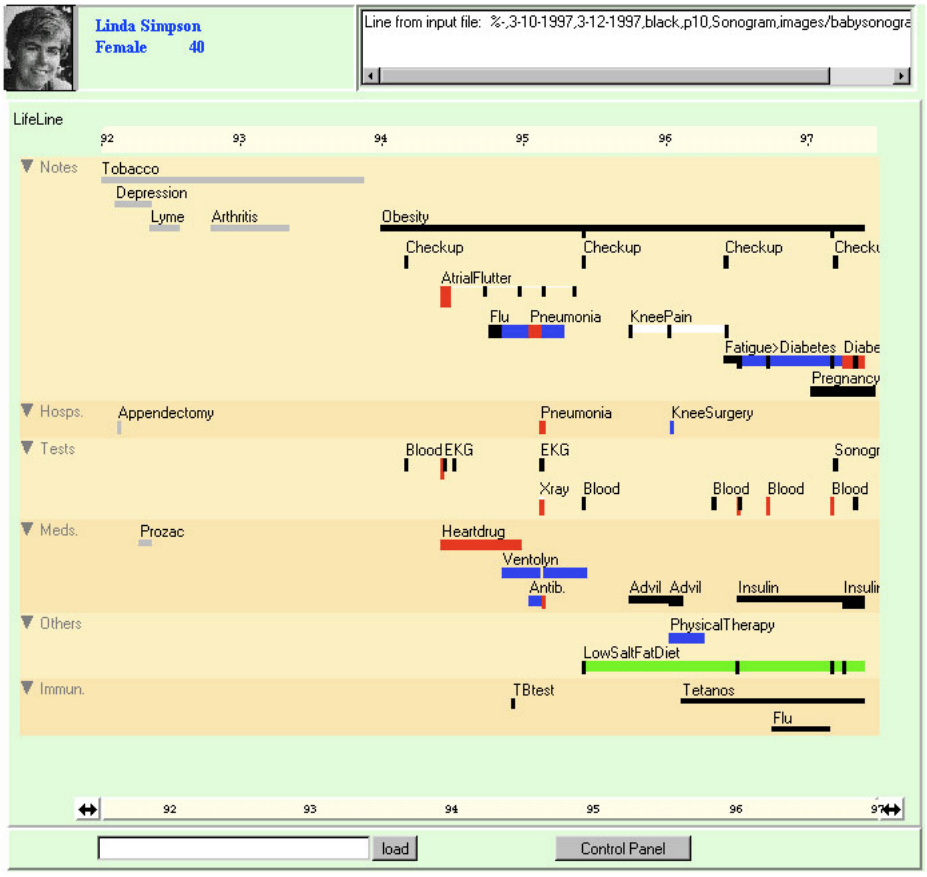

Figure 2.1: *LifeLines* [97] displays a medical history of a single patient on a timeline.

that represent start and end intervals. *TimeZoom* [34] allows zooming into regions of the timeline, permitting arbitrary granularity of time units. It supports multiple focus regions with various levels of detail, which allows comparison between multiple regions while preserving the overall context.

Spiral timelines—in which angle represents time interval (time of day, days of week, months, or years)—were inspired by the cyclic nature of how we organize time and used to reveal periodic patterns in time series [25, 49, 137]. Carlis *et al.* [25] proposed a visualization on a spiral timeline in both 2D and 3D. Visualizations, such as bar charts or size-coded circles, are placed along the spiral to represent the values of the events. Hewagamage *et al.* [49] used multiple 3D spirals on a geographical

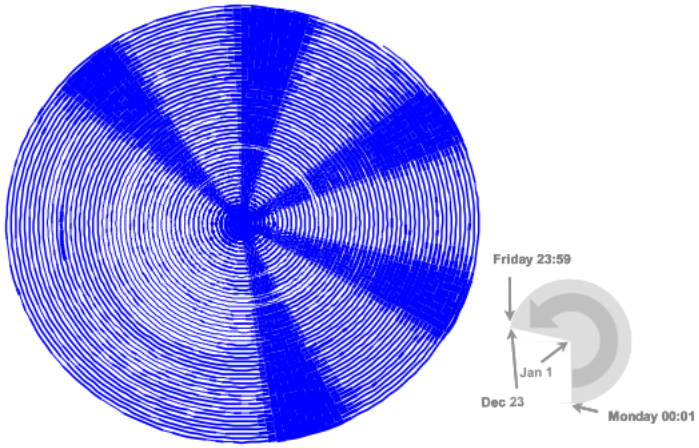

Figure 2.2: The Spiral visualization approach of Weber *et al.* [137] applied to the power usage dataset

map to visualize spatio-temporal patterns. Weber *et al.*'s work [137], as shown in Figure 2.2, allowed the users to adjust the length of the cycle to find periodic patterns. Suntinger *et al.* [120] visualized event sequence streams by plotting circles on a radial layout. Outer ring represents more recent time.

Tree-based representation was used by *VizTree* [72, 73, 74] to detect patterns in numerical time series. The numerical values are binned into categorical values. VizTree then displays a prefix tree of patterns and uses edge thickness to represent frequency. This representation can easily reveal frequent and rare patterns in the data. They also presented *DiffTree* as another view that supports comparison between time series by showing differences between trees.

2.1.1.2 Visualization of multiple records in parallel

To support the analysis of multiple records, a common technique is to stack instances of single-record visualizations [37, 127]. Many compact and space-efficient

single-record visualizations were proposed.

In *PatternFinder* [37], each record is displayed with a *ball-and-chain* visualization. *LifeLines2* [131] uses a single-record visualization based on LifeLines idea (Figure 2.4). Vrotsou *et al.* [127] visualized social science diary data using the vertical axis to represent time of day while the horizontal axis represent records. The visualization is compact, using one vertical line with many color-coded sections that represent events to represent each record. Weber *et al.*'s spiral-based visualization [137] also supports multiple records by showing each record as one ring. CloudLines [65] aggregates overlapping events on the timeline using decay function to better display high and low density areas.

Those tools typically provide searching and filtering mechanisms to enhance the analysis [37, 131, 129]. LifeLines2 introduces Align-Rank-Filter (ARF) framework to support browsing and searching. The idea is to rearrange the time line by important events, rearrange records by event occurrence frequency, and filter by data characteristics. *ActiviTree* [129] combines the visualization in [127] with a framework for finding interesting sequential patterns. The query interface uses a tree-based visualization, showing all the events that occur before and after the current query pattern with their scores.

In the context of my inspirational case study, users of those tools could find patients who were transferred with a specific known sequence, or find patients who were admitted to the ICU at least once, but could not find out what the common sequences are or spot anomalous sequences.

Some tools allow the users to organize records into hierarchy [23] or groups [92].

Timeline Tree [23] is a tree with timelines at the leaf nodes. The tree groups the timelines into hierarchy based on their attributes. Timeline Tree also has *time bars* as an alternative view. The time or order of transactions is encoded using color coding and the measure is represented by the width of the boxes instead of their height. Phan *et al.* [91, 92] proposed a visualization called *progressive multiples* that allows the users to create folders to organize timelines into different groups. By default, all records are placed sequentially in the initial folder. Users can create a new folder and drag records into the new folder.

However, unlike LifeFlow which provides one visual abstraction that represents multiple records, these systems do not provide any abstraction. Without the overview abstraction, users lose the big picture when the number of records exceeds the maximum number of single-record visualizations that can be displayed on the screen. For example, the visualization in Figure 2.4 can display only 8 records from a total of 377 records on the screen. Also, it does not provide a summary or make common patterns within records stand out.

2.1.1.3 Visualization that aggregates multiple records

Some systems provide an overview of multiple records [48, 10, 132]. *Continuum* [10] shows a histogram of frequency of events over time while *LifeLines2* [132] has a *temporal summary*, which is a stacked bar chart that shows the distribution of event types within each period of time.

These methods can provide the distribution of events by time, which answers

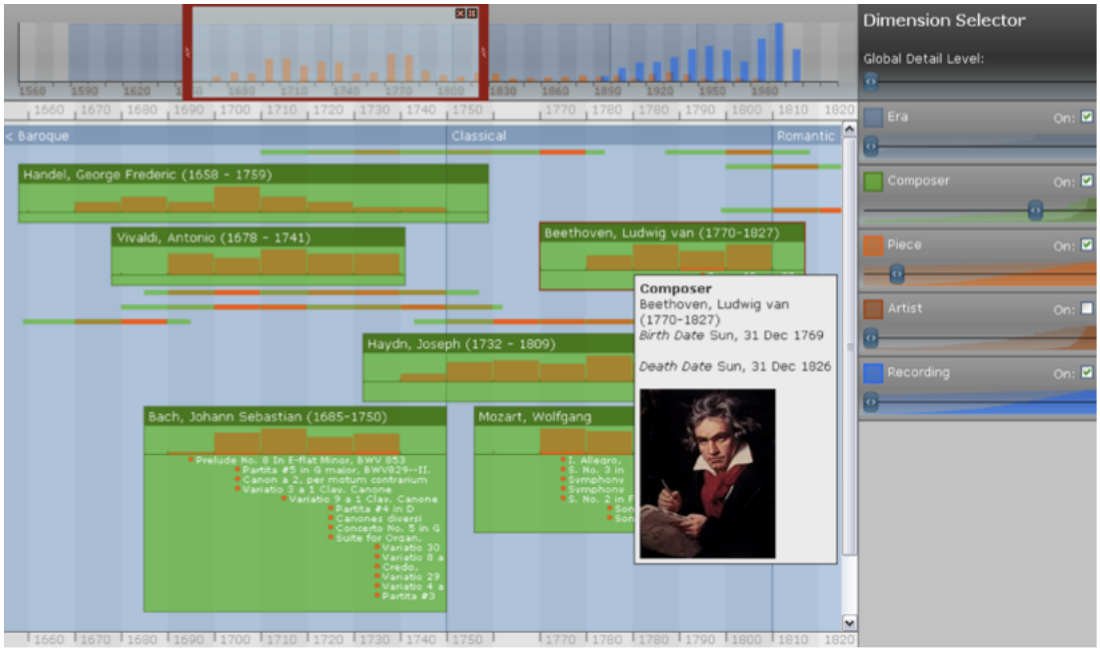

Figure 2.3: *Continuum*'s overview panel (top) displays a histogram as an overview of the data set. [10]

the questions related to the distribution, such as which type of event occurred most frequently in January 2007 or which type of event usually occurred within the first week after patients arrived at the hospital? However, the event sequences within the records are obscured and thus, it cannot answer questions related to sequences, such as where did the patient usually go directly after arrival or what is the most common transfer sequence?

2.1.2 Hierarchy Visualization

To visualize an overview of multiple event sequences, I group all records into a hierarchical structure called a *Tree of Sequences*. The overview visualization is then created from this hierarchical structure. According to Stasko and Zhang [118], many visualizations for displaying hierarchical structures were developed.

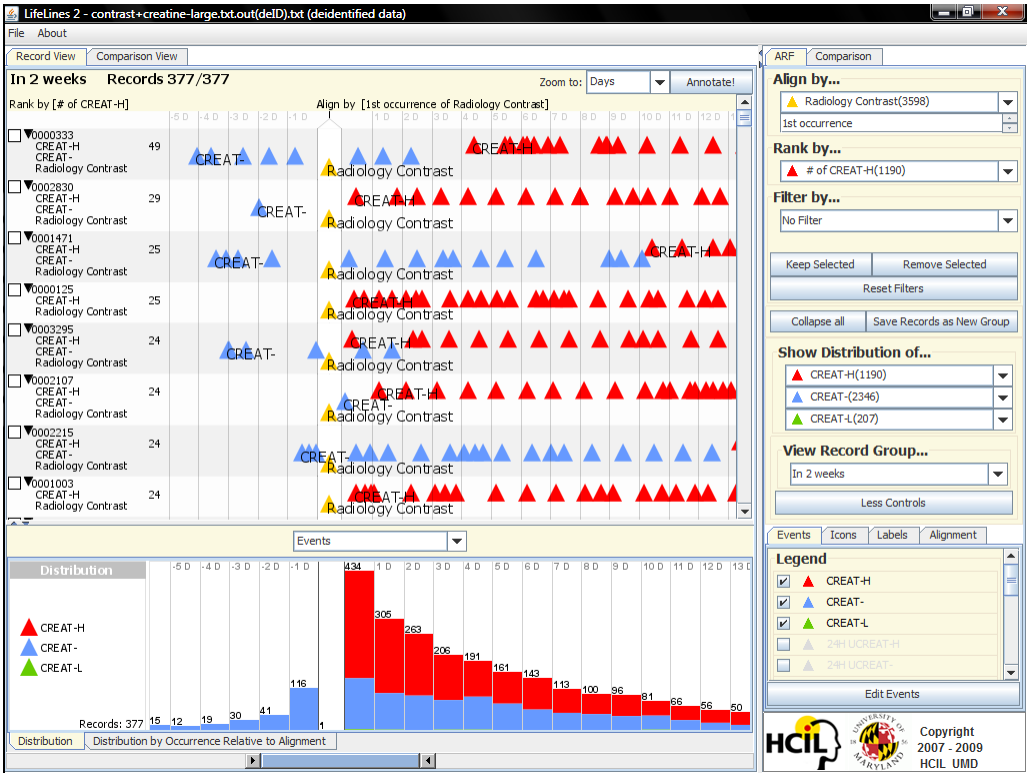

Figure 2.4: *LifeLines2* [131, 132] displays a *temporal summary* in the bottom as an overview of the data set.

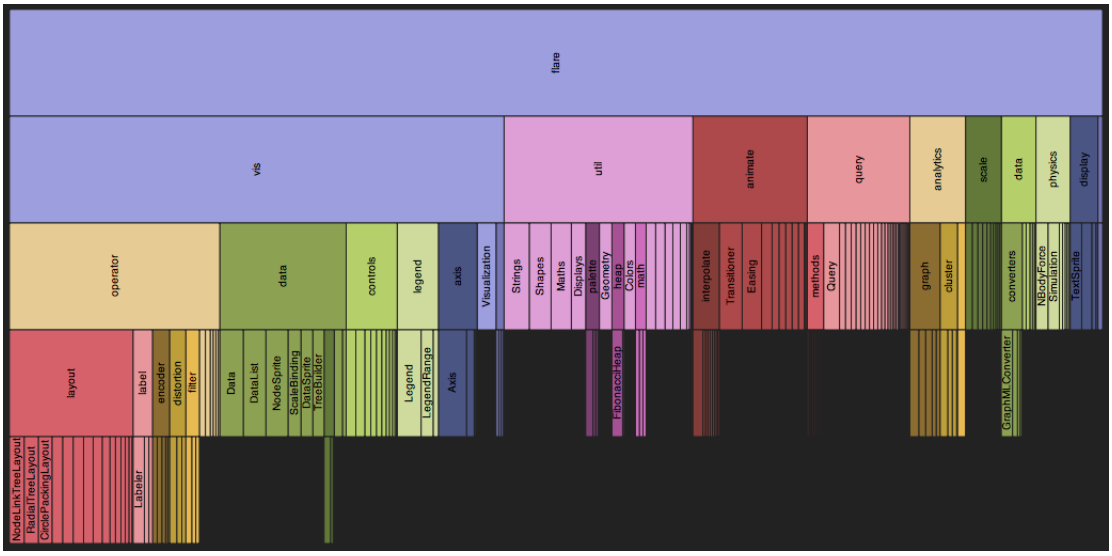

Figure 2.5: Icicle tree in ProtoVis. [21] The visualization is called an icicle tree because it resembles a row of icicles hanging from the eaves. [66]

The most common way is to display a node-link layout in 2D [35, 138, 101], or 3D [104] or hyperbolic space [68, 82]. Some visualizations add more interaction techniques to enhance the node-edge tree visualizations. For example, *SpaceTree* [96] allows node expansion on demand. *TreeViewer* [61] visualizes trees in a form that closely resembles botanical trees.

A *dendogram* [117, 47, 107] is a tree for visual classification of similarity, commonly used in biology for grouping species. A dendogram can show the relative sequence similarity between many different proteins or genes. Generally, the horizontal or vertical axis indicates the degree of difference in sequences and another axis is used for clarity to separate the branches.

Space-filling techniques use implicit containment and geometry features to present a hierarchy [55, 66, 38, 21, 118]. *TreeMaps* [55, 15] display hierarchical data as a set of nested rectangles. Each branch of the tree is given a rectangle, which is

then tiled with smaller rectangles representing sub-branches. A leaf node's rectangle has an area proportional to a specified dimension on the data. *Icicle tree* [66, 38, 21], also called *Icicle plot*, displays hierarchical data as stacked rectangles, usually ordered from top to bottom. This visualization directly inspired LifeFlow design (Figure 2.5). The root takes the entire width. Each child node is placed under its parent with the width proportional to the percentage it consumes relative to its siblings. *Sunburst* [118] is a radial space-filling technique, which is essentially the polar form of the Icicle tree. At the core is the root of the tree, each concentric ring represents the child nodes and is partitioned to represent the percentage a node consumes relative to its siblings.

Using these methods, the sequences of events can be represented. For example, the users can see what type of events usually occur after `A`. However, the length of time between events—which is important in many analyses—is not represented. For example, the users can see that `B` usually occurred after `A`, but they cannot see how long after `A` occurred that `B` occurred. Therefore, LifeFlow was designed to also display the gap between events.

A *phylogenetic tree* [69] is a branching diagram showing the inferred evolutionary relationships among various biological species. The edge lengths in some phylogenetic trees may be interpreted as time estimates, but each node in the tree represents only one species, while each node in LifeFlow represents multiple records of event sequences.

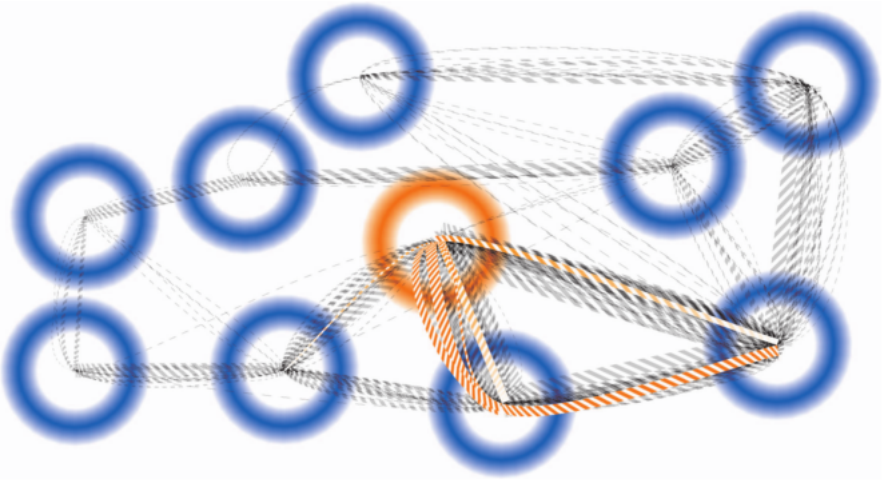

Figure 2.6: Blaas *et al.*'s state transition visualization: A smooth graph representation of a labeled biological time-series. Each ring represents a state, and the edges between states visualize the state transitions. This graph uses smooth curves to explicitly visualize third order transitions, so that each curved edge represents a unique sequence of four successive states. The orange node is part of a selection set, and all transitions matching the current selection are highlighted in orange. [17]

2.1.3 State Transition Visualization

An alternative approach to aggregate multiple event sequences is to consider each event sequence as a path through different states. In the simplest case, a state can be an event type. For example, an event sequence `A` → `B` → `A` represents a path from state `A` to state `B` and back to state `A` again. A state transition graph then can be created from multiple event sequences. This idea was explored in a spin-off project, *Outflow* (Appendix C).

Most of the state transition visualizations were inspired by a *state diagram* [20], or *state transition graph*. A state diagram is used in computer science and related fields to represent a system of states and state changes, e.g. finite state machines. State diagrams are generally displayed by simple node-link diagrams, where each node represents a state and each directed link represents a state transition [17].

Many visualizations based on the state diagrams were developed [125, 136, 99, 100, 17]. These approaches typically focus on displaying multivariate graphs where a number of attributes are associated with every node.

There are also extensions of a state diagram. For example, a *Petri net* (also known as a *place/transition* net or *P/T net*) [83] allows transitions from any number of states to any number of states. This property was designed to support the systems which can be in multiple states at once. Petri nets offer a graphical notation for stepwise processes that include choice, iteration, and concurrent execution. It was invented in 1939 by Carl Adam Petri—at the age of 13—for the purpose of describing chemical processes [90]. Petri nets are used to display the flow of distributed systems, where many concurrent processes are executed at the same time. Van der Aalst [83] adapted Petri nets for workflow management.

However, using an approach based on state transition visualizations also has some limitations since these visualizations were designed to emphasize the transitions between states (event types, in this case). It is hard to recognize the sequences of events longer than two events and, therefore, it is harder to find common sequences or outliers from the records. It is also difficult to incorporate temporal information into graph visualizations, which are often complex already when there are many nodes and edges.

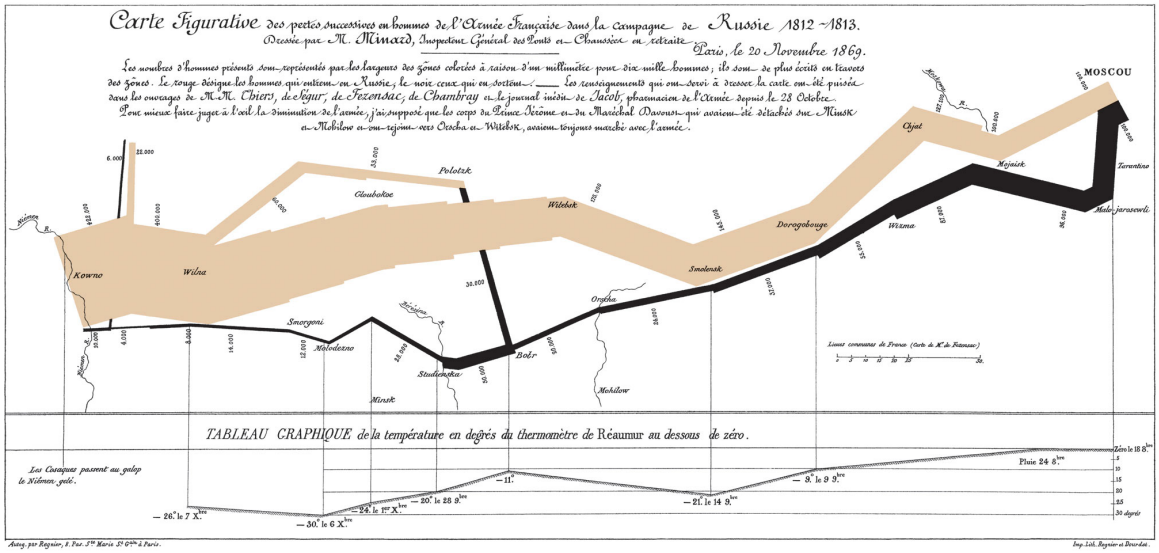

Figure 2.7: Charles Minard's Map of Napolean's Russian Campaign of 1812

2.1.4 Flow Visualization

Cartographers have long used *Flow Maps* to show the movement of objects from one location to another, such as the number of people in a migration, the amount of goods being traded, or the number of packets in a network [93, 102, 22]. The most famous one is Charles Minard's Map of Napolean's Russian Campaign of 1812 (Figure 2.7) [40]. However, these flow maps only focus on displaying proportion of the flow that splits in different ways, without showing temporal information or steps in the process.

2.2 Query Methods

With respect to my work in overview visualization, related research in visualizations are highlighted and reviewed in the previous section. In this section, I collect query methods for temporal data including important work in related areas,

and summarize them to provide background knowledge for querying event sequences.

2.2.1 Query Languages

A traditional approach to querying temporal data is to use database query languages. According to Chomicki [30] and Tansel and Tin [122], 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 [115], TSQL2 [116] and Historical Relational Data Model (HRDM) [31]. 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 support only exact match.

2.2.2 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[87], 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* [145], which was refined by others [26, 64, 146, 53, 86, 121].

Time-by-Example [121] followed the Query-by-Example idea and adopted subqueries concepts from *Aggregates-by-Example* [64] and *Summary-Table-by-Example* [86]

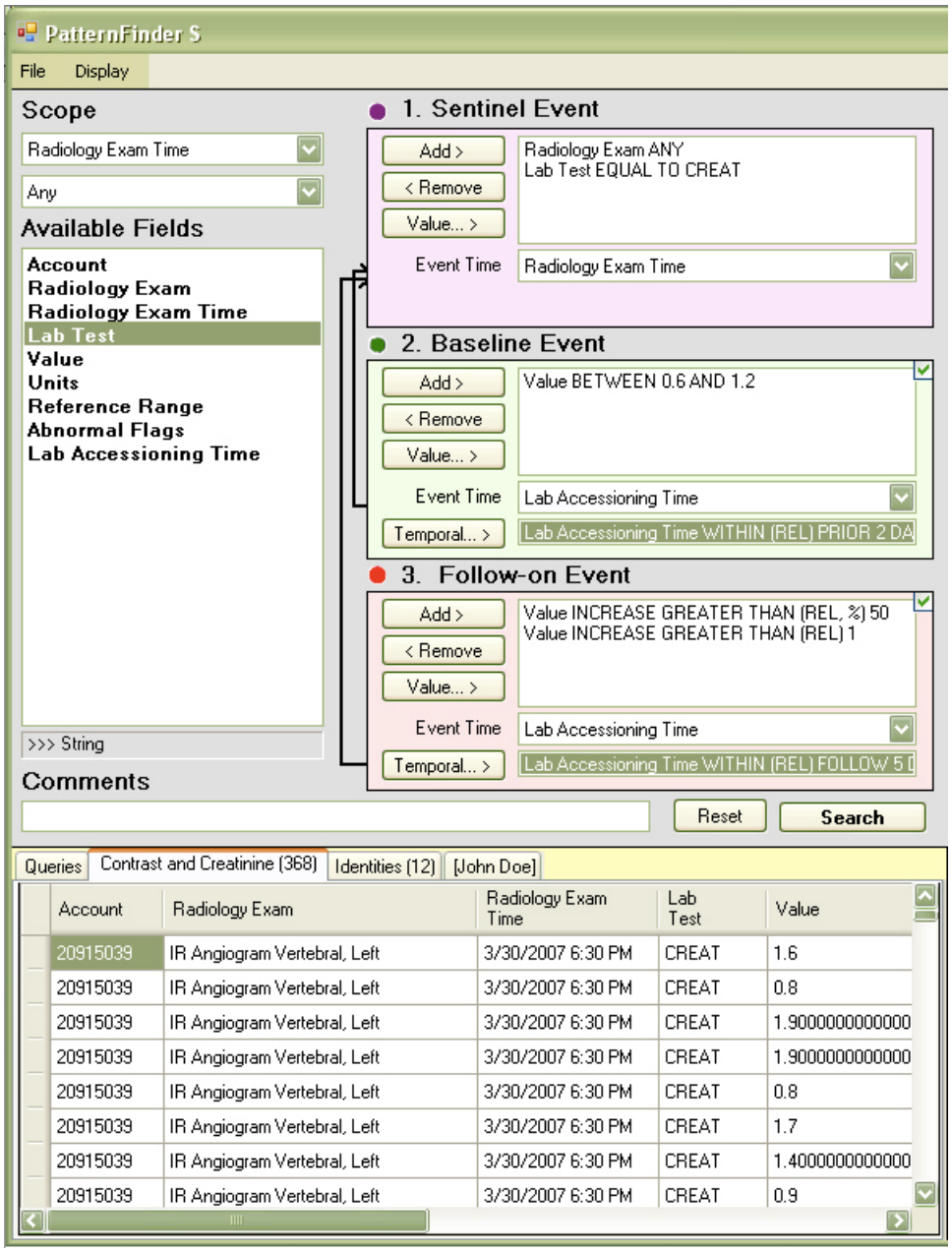

Figure 2.8: An example of query-by-filters: PatternFinder [37] allows users to specify the attributes of events and time spans to produce pattern queries. The parameters in the controls are converted directly into constraints that can be used to retrieve the records.

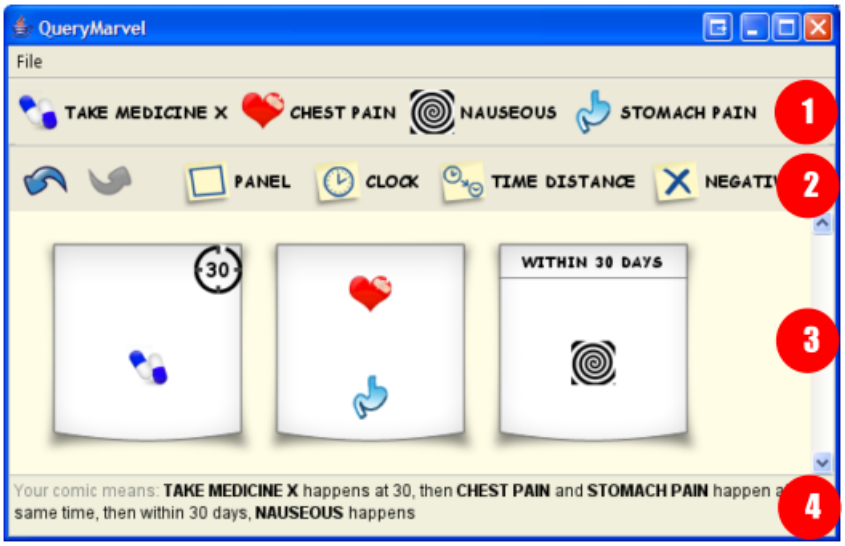

Figure 2.9: An example of query-by-example: QueryMarvel [54] allows users to draw comic strips to construct queries. With its exact result back-end, the comic strips are converted into rules, as seen in the status bar. (4).

to serve a historical relational database (HRDB). HRDB is an extension of RDB that stores 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 can keep track of the room values and Time-by-Example provides a way to query those values. For instance, the users can 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 supports only exact match, operates only on top of HRDM and still requires 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`".

2.2.3 Query by Graphical User Interfaces (GUIs)

2.2.3.1 Exact Match Approach

As the graphical user interfaces (GUIs) were becoming more common, many GUIs were developed for temporal data [5, 109, 62, 63]. 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 [59] presented a visualization called *xtg*, which allows users to explore temporal data and do simple searches for events. Hibino and Rundensteiner [50, 51] proposed a visual query language and user interface for exploring temporal relationships using slider filters with results displayed in a graph-like visualization. *PatternFinder* [37] allows users to specify the attributes of events and time spans to produce pattern queries that are difficult to express with other formalisms. *LifeLines2* [131, 132, 130] uses an alignment, ranking and filtering (ARF) framework to query for event sequences. *ActiviTree* [129] provides a tree-like user interface with suggestions about interesting patterns to query for sequences of events. *QueryMarvel* [54] utilizes and extends the semantic elements and rules of comic strips to construct queries. Instead of following the exact match approach, Similan follows the similarity search approach (Section 2.2.3.2), and applied the concept for querying event sequences.

2.2.3.2 Similarity Search Approach

Many GUIs follow the *similarity search* approach, in which users can draw an example of what they expect to see as a result of a query. The result from a query is a list of records, sorted by similarity to the given example. Kato *et al.* [60] presented *QVE* that accepts a sketch drawn by users to retrieve similar images or time series from the database. *IFQ (In Frame Query)* [71] is a visual user interface that supports direct manipulation [111] allowing users to combine semantic expressions, conceptual definitions, sketch, and image examples to pose queries. *Spatial-Query-by-Sketch* allows users to formulate a spatial query by drawing on a touch screen and translates this sketch into a symbolic representation that can be processed against a geographic database. Bonhomme*et al.* [19, 18] 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 uses visual metaphors, such as balloons and anchors, to express spatial and temporal criteria. *QuerySketch* [135] allows users to sketch a graph freehand, then view stocks whose price histories matched the sketch. Watai *et al.* [134] proposed a web page retrieval system that enables a user to search web pages using the user's freehand sketch. *WireVis*[27] introduces techniques to extract bank accounts that show similar transaction patterns. To the best of my 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 me to develop a similarity search tool for event sequences.

Timesearcher [52] visualizes multiple timelines as line charts on the same

plane, using horizontal and vertical axis to represent time and value, respectively. Users draw timeboxes, rectangular widgets that can be used to specify query constraints, on the timeline to query for all time series that pass through those timeboxes. In Timesearcher, users can draw an example (timeboxes) to specify the query, but the timeboxes are converted into exact rules, e.g. January $<$ time $<$ March and $100 <$ value < 200 , when processing the query in the background. Similan2 allows users to draw an example, but does not convert the example into any exact rule. Instead, it compares the example with each record directly and sorts the result by similarity to the example.

2.2.4 Similarity Measure

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

2.2.4.1 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 a series of categorical values measured over time. Hence, these approaches are not directly applicable to event sequences because they were designed to capture the difference between numerical values, not categorical.

Nevertheless, there are some common concepts that are worth mentioning here. The first concept is *lock-step* measure, which compares 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 is fixed, these measures are sensitive to noise and misalignments in time. The M&M measure is different from lock-step measures because it does not fix the mapping of i-th events together.

The second concept, *elastic* measure, allows comparison of one-to-many points (e.g., Dynamic time warping (DTW) [16] and one-to-many / one-to-none points (e.g., Longest Common Substring (LCSS)). The sequences are *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 does not allow one-to-many mapping.

2.2.4.2 String and Biological Sequences

Edit distance is the number of operations required to transform one string into another string. The lower the number, the more similar the strings are. *Hamming distance* [44], *Levenshtein distance* [70] or *Jaro-Winkler distance* [140] are some examples. The best known such distance is the *LCSS* distance [11]. A more completed survey can be seen from [84].

One neighbor area is biological sequence searching. There exist many algorithms for comparing biological sequence information, such as the amino-acid se-

quences of different proteins or the nucleotides of DNA sequences. BLAST [9], FASTA [88] and the TEIRESIAS algorithm [103] are some examples.

Mongeau and Sankoff [81] defined a similarity measure specifically for comparing musical pieces based on number of transformations required to transform one into another. Their measure allows one-to-many mapping called *consolidation/fragmentation*, which is similar to time warping. Gómez-Alonso and Valls [41] proposed a similarity measure for sequences of categorical data (without time) based on edit distance.

These approaches consider the difference in ordering and existence, but do 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 event 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.

2.2.4.3 Event Sequences

Mannila and Ronkainen [77] 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 allows only monotonic mapping, which means that the matched events in the target and candidate sequences must be in similar order, and do not offer a user interface. Sherkat and Rafiei [110] binned the timeline into intervals and compared events within each interval.

The Match & Mismatch (M&M) measure v.1 [144] calculates a similarity score from two types of difference: time difference of matched events and number of mismatches. It supports matching that may not preserve the order of event sequence (non-monotonic). *Timed String Edit Distance* [36] inserts timed null symbols into event sequences before matching. It allows matching between events with different event types and measured two types of difference: time difference and event type difference (symbol dissimilarity). Vrotsou *et al.* [126, 128] identified nine measures to cover several aspects of similarity. This approach also considers multiple occurrences of the target sequence in the candidate sequence. Obweger *et al.* [85] defined *single-event similarity* by comparing event attributes. Their event sequence similarity then combines 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 [76] 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 [78] 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 medical interests, such

as human activity event streams, business transactions, or legal actions.

2.3 Summary

Decades of innovations in temporal data visualization and query methods are surveyed and reviewed in this chapter. On one end, researchers had developed many visualization techniques aiming to present information in a meaningful way and reveal underlying patterns. There are techniques ranging from displaying a single record to displaying multiple records in parallel. However, as data collections grow, it is becoming more difficult to detect any pattern among records when the users can see only a small portion of a huge dataset. An overview is needed. A few histogram-based techniques were developed but there are still many scenarios that they cannot support. Hierarchical, graph and flow visualization techniques seem to be applicable to some extent but still, they were not designed to support time in the display. New techniques that can aggregate and provide an overview of multiple records are needed.

On the other end, many query languages and GUIs were invented for querying temporal data. Specific query languages are very expressive but found to be complex and difficult for typical end users. Graphical user interfaces (GUIs) were then used to bridge the gap. The majority of the GUIs are *exact match* interfaces, which are precise, but at the same time, inflexible for queries with uncertainty. A more flexible alternative, *similarity search* interface, was used for querying numerical time series and other non-temporal data. Designing a similarity search interface for querying

event sequences remains an open problem. One part of the puzzle is how to define a similarity measure that can capture different definitions of similarity. A further question is how to design an interface that can provide both precision of the exact match and flexibility of the similarity search.

These unsolved problems present exciting research opportunities. The following chapters describe how I approached these problems and explain my solutions in more detail.

Chapter 3

Providing an Overview of Temporal Event Sequences to Spark Exploration: LifeFlow Visualization

3.1 Introduction

Previous work on temporal data visualization can support many types of event sequence analysis, ranging from examining a single record in detail to various ways of filtering and searching multiple records. They can answer questions regarding the number of records that include a specific event sequence, but questions requiring an overview need innovative strategies. For example, a question such as "What are the most common transfer patterns between services within the hospital?" requires examination of all records one by one. Being able to see all records on the screen at once would greatly facilitate pattern discovery.

Squeezing a billion records into a million pixels [113] is a great challenge in information visualization. On one hand, researchers want to be able to display millions of event sequences within limited space—a computer monitor, for instance. That means the data must be somehow aggregated to create a scalable visualization. On the other hand, we want to be able to preserve as much important information as possible. So, a trade-off between what information needs to be preserved and what information should be compromised has to be made.

To address this challenge, I aggregate event sequences into a data structure called a *tree of sequences* and introduce a novel interactive visualization called *LifeFlow*, which can summarize all possible sequences and the temporal spacing of the events within sequences. In this chapter, I describe a motivating example from the medical domain, explain the tree of sequences data structure, then introduce the LifeFlow visualization, describe the user interface and basic features, present the results of a user study, and finally describe advanced features that I have developed while working with the domain experts in long-term case studies. An earlier version of this chapter has been published in [142].

3.2 Motivating Case Study

The use of information visualization to support patient care and clinical research is gaining momentum [5, 63, 29]. This section describes a particular case study that motivated the original design of LifeFlow. It was conducted with Dr. Phuong Ho, a practicing physician in an emergency department, who was interested in analyzing sequences of patient transfers between departments for quality assurance.

3.2.1 Event Definitions

These terms will be used when describing the case study.

1. *ER*: Emergency Room, the section of a hospital intended to provide treatment for victims of sudden illness or trauma, also called Emergency Department

(ED)

2. *ICU*: Intensive Care Unit, a hospital unit in which patients requiring close monitoring and intensive care are kept

3. *IMC*: Intermediate Medical Care, a level of medical care in a hospital that is intermediate between ICU and Floor

4. *Floor*: a hospital ward where patients receive normal care

3.2.2 Example question

One of Dr. Ho's particular interests was the monitoring of *bounce backs*, which occurs when a patients' level of care is decreased then increased again urgently. For example, a patient's condition might have improved enough to have him transferred from the `ICU` to `Floor`, but his condition worsened again and he had to be sent back to intensive care within 48 hours, suggesting he might have left the ICU too early. This pattern corresponds to a hospital quality metric that is very rarely monitored. Dr. Ho had been using an MS Excel spreadsheet to find these patients. In an interview he described the complex and time consuming effort to create the formulas and view the data. This is due in part to the fact that there are many room types and special conditions for using those rooms. My research group had previously worked with Dr. Ho using *LifeLines2* [131] and Similan [144] to locate patients with specific known event sequences such as the one described above. Once it had become easy to search for specific known sequences we identified other questions that could not be answered as easily; e.g.: what typically happens to patients after

they leave the ER, or the ICU; what are the most common transfer patterns, what is the percentage of patients transferred from ICU to Floor and how long does it take, are there any unexpected sequences? All those new questions require a summary of all the transfer sequences and their temporal attributes. To this end, I propose *LifeFlow* to provide an overview of all transfer sequences.

3.3 Data Aggregation: Tree of Sequences

The first step of creating LifeFlow is to aggregate the data into a *tree of sequences*. The tree of sequences data structure was designed to preserve two important aspects of event sequences:

1. *Sequence of events*

   e.g. Arrival→Intensive Care Unit (ICU)→Floor

2. *Time gaps between events*

   e.g. Emergency Room → *(and after 5 hours)*→ Transfer to ICU

Figure 3.1 illustrates the conversion from four records of event sequences to a tree of sequences and then LifeFlow visualization. Raw data are displayed on a horizontal timeline with colored triangles representing events (in the same approach as LifeLines2 [131]). Each row represents one record. All records are aggregated into a tree of sequences based on the prefixes of their event sequences. For example, a record that contains event sequence Arrival → ER → ICU and a record that contains event sequence Arrival → ER → Floor, share the same prefix sequence Arrival → ER. By default, the records are grouped event-by-event from the begin-

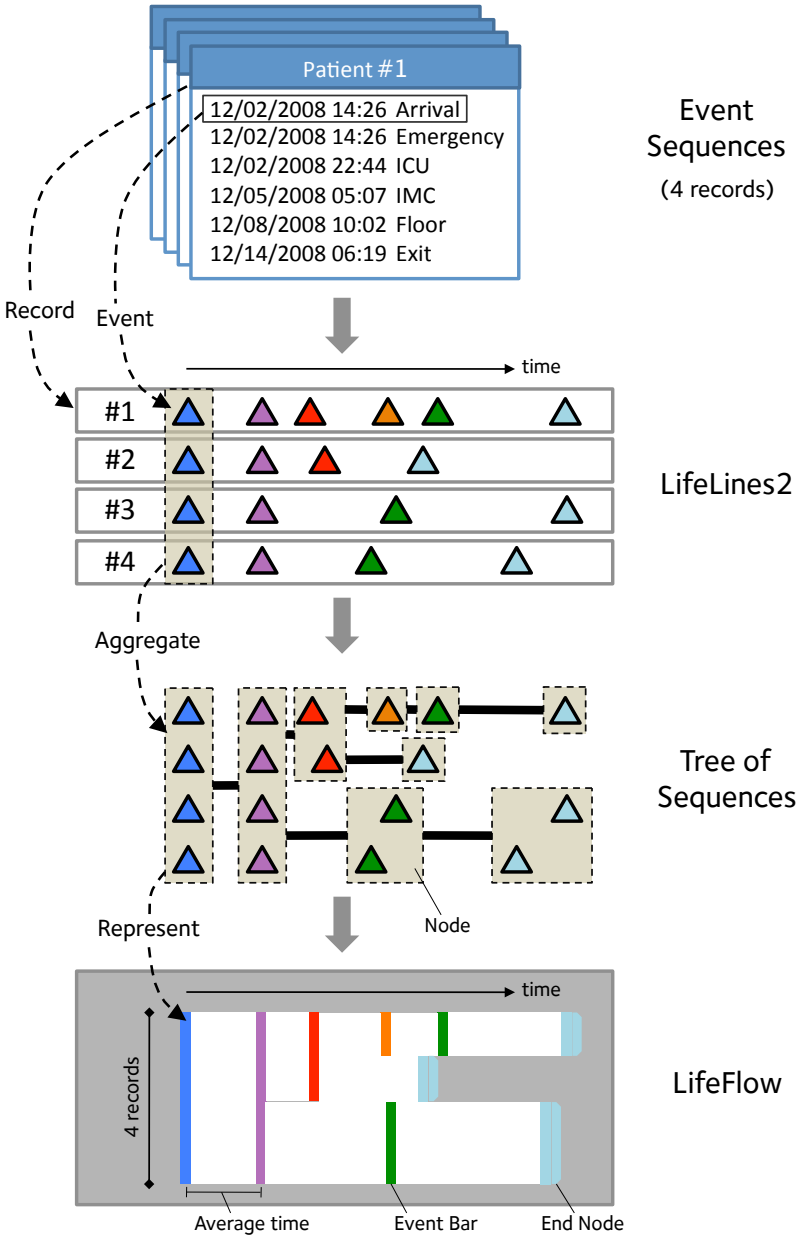

Figure 3.1: This diagram explains how a LifeFlow visualization can be constructed to summarize four records of event sequences. Raw data are represented as colored triangles on a horizontal timeline (using the traditional approach also used in LifeLines2). Each row represents one record. The records are aggregated by sequence into a data structure called a tree of sequences. The tree of sequences is then converted into a LifeFlow visualization. Each tree node is represented with an event bar. The height of the bar is proportional to the number of records while its horizontal position is determined by the average time between events.

ning of the event sequences to the end. In Figure 3.1, all records start with the blue event so they are grouped together (indicated by dashed rectangle) into a blue tree node. Then, they all also have the purple event, so they are still grouped together into a purple node. In the next step, two of them have red events while the other two have green events, so they are split into red and green nodes. Then do the same for the rest of the event sequences.

Figure 3.2 shows a tree of sequences in more detail. Each node in the tree contains an event type and number of records while each edge contains time gap information and number of records.

In some situations, users may choose to group consecutive events of the same type together when building the tree of sequences. For example, two consecutive transfers to Floor can be treated as one transfer with the second transfer ignored. A record with sequence `Arrival` → `ER` → `Floor` → `Floor` → `ICU` is treated as `Arrival` → `ER` → `Floor` → `ICU`. This aggregation option can be turned on or off as needed via the user interface.

Inspired by LifeLines2, LifeFlow allows users to choose any event type to be an *alignment* point. This supports tasks such as "what happened to the patients before and after they went to the `ICU`?", in which users can select `ICU` as an alignment point to answer. By default, an alignment point is not specified, so all records are aligned by the first event in the record (Figure 3.2). An invisible root node is used as a starting point because all event sequences may not start with the same event type, which can result in multiple trees. Root node can connect these trees together into one tree.

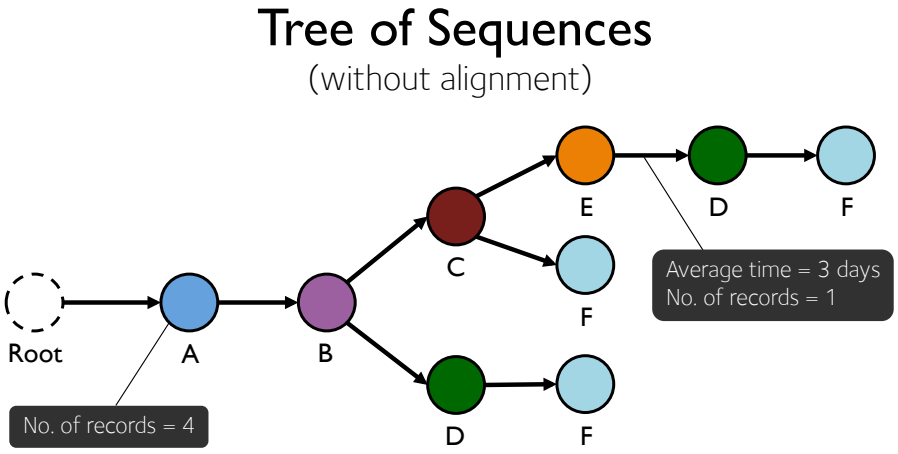

Figure 3.2: Tree of sequences data structure of the event sequences in Figure 3.1: Each node in the tree contains an event type and number of records. Each edge contains time gap information and number of records.

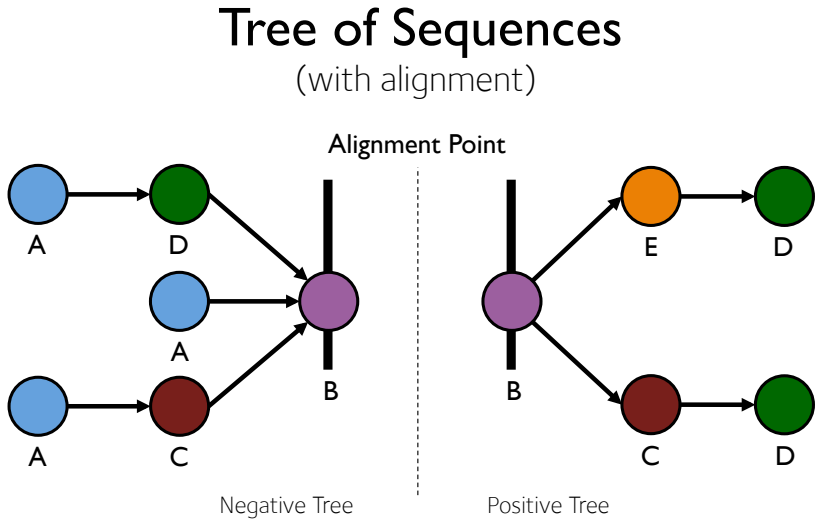

Figure 3.3: Two trees of sequences created from a dataset with an alignment point at B. A *positive* tree in forward direction and a *negative* tree in backward direction.

When an alignment point is specified, two trees are built separately from the alignment point. One tree for the sequences before the alignment (from right to left) and another tree for the sequences after the alignment (from left to right). Because the alignment point is time zero, the tree on the left is called *negative* tree because it occurs in negative time while the tree on the right is called *positive* tree. Figure 3.3 shows two trees of sequences built from another dataset with alignment point set at event type `B`.

The tree of sequences data structure can be created in $O(n)$ when n is a total number of events in a dataset. A tree building algorithm iterates through each record in a dataset and adds more information to existing nodes in a tree or grow new nodes if necessary. This approach reduces amounts of information from multiple event sequences into a data structure whose size depends on the number of patterns instead number of records, and makes it easier to be visualized. I then design LifeFlow visualization to visualize the tree of sequences. Inspired by the tree of sequences and LifeFlow, Lins *et al.* [75] develop an alternative visualization for the tree of sequences.

3.4 Visual Representation: LifeFlow Visualization

Once a tree of sequences is created, it can be encoded into a LifeFlow visualization (Figure 3.1). Each node of the tree is represented with a color-coded *event bar*, matching the color of the event type. The height of a bar is determined by the number of records in that node proportionally to the total number of records.

For example, the red node contains two out of four records, so the height of its corresponding event bar is 50% of the total height. The horizontal gap between a bar (e.g. purple bar) and its parent (the blue bar on its left) is proportional to the mean time between the two events (blue → purple). By default, the representative time gap is the mean, but users can select other metrics, such as the median. Each sequence ends with a trapezoid *end node*.

The LifeFlow visualization scales for number of records, but is limited by number of patterns in the dataset and number of events in each record. One dataset that has four records and another dataset that has four millions records may have the same number of patterns and can be displayed using the comparable amount of screen space. Recommended datasets should have less than 100 patterns, with each record having less than 40 events. The number of records should not be an issue theoretically, but in its current implementation, LifeFlow is recommended for analyzing datasets with less than 250,000 records.

3.5 Basic Features

I implemented a software prototype *LifeFlow* as a Java desktop application. (Please refer to the Appendix B for more implementation detail.) In addition to the compact visual representation, LifeFlow (Figure 3.4) includes the following interactions to support exploration:

1. **Semantic Zooming**: The horizontal zoom changes time granularity, while the vertical zoom allows the users to see rare sequences in more detail. Users

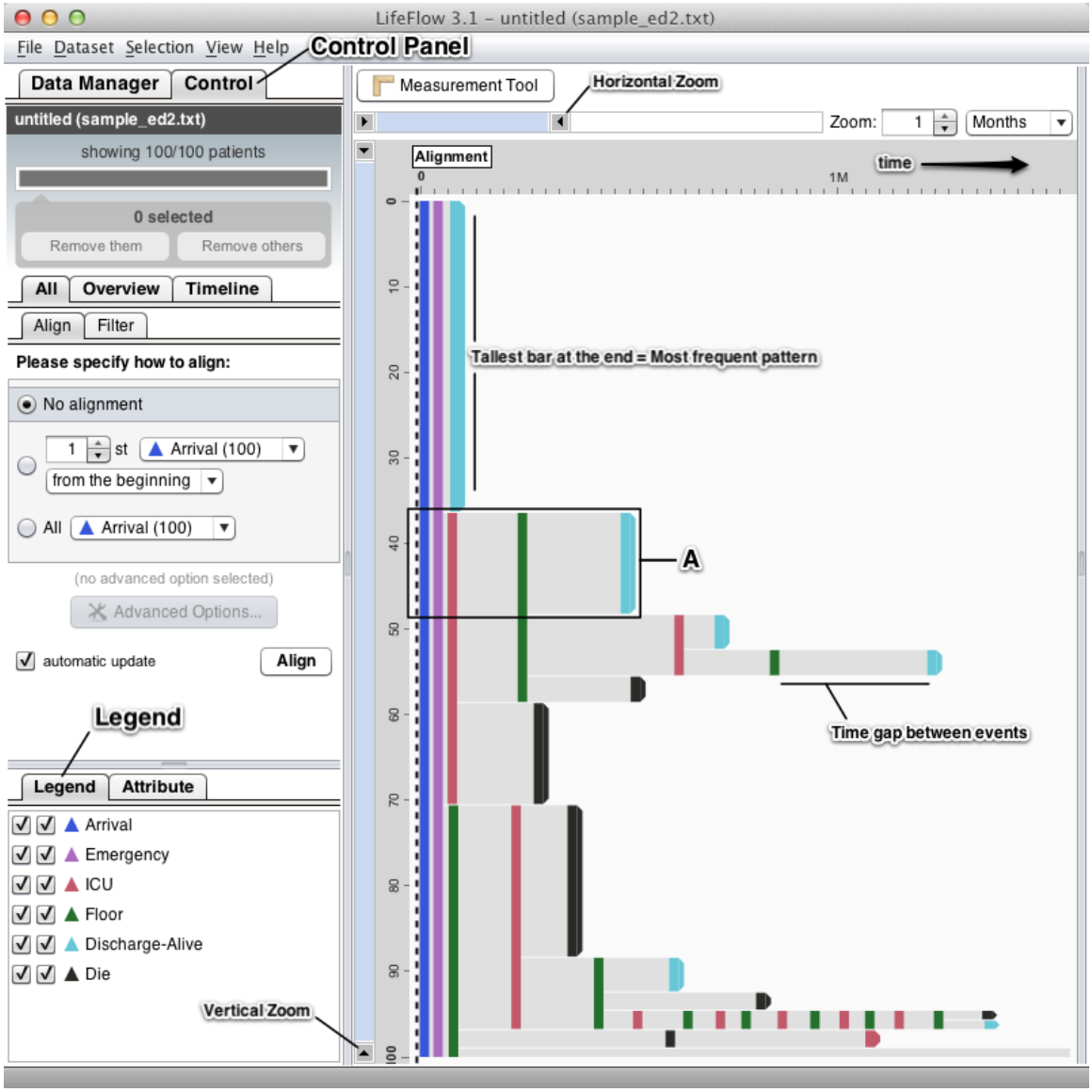

Figure 3.4: This screenshot of LifeFlow shows a random sample of patient transfer data based on real de-identified data. The way to read sequences in LifeFlow is to read the colors (using the legend). For example, the sequence (A) in the figure is `Arrival` (*blue*), `Emergency` (*purple*), `ICU` (*red*), `Floor` (*green*) and `Discharge-Alive` (*light blue*). The horizontal gap between colored bars represents the average time between events. The height of the bars is proportional to the number of records, therefore showing the relative frequency of that sequence. The bars (e.g. `Floor` and `Die`) with same parent (`Arrival` → `Emergency` → `ICU`) are ordered by frequency (tallest bar on top), as you can see that `Floor` (*green* bar) is placed above `Die` (*black* bar). The most frequent pattern is the tallest bar at the end. Here it shows that the most common sequence is `Arrival`, `Emergency` then `Discharge alive`. (Please note that this is only a sample dataset and does not reflect the real performance of any hospital.)

can also right-click any sequence and select "Zoom to this sequence". The visualization will animate and zoom to the selected sequence.

2. **Tooltip**: When users move the mouse cursor over an event bar (Figure 3.5), a tooltip displays the full sequence of events, and some statistical information, such as mean time between events, standard deviation, etc.

3. **Overlay distribution of gap between events**: Hovering the cursor over a bar displays the distribution of time gaps overlaid on the Lifeflow. Figure 3.5 shows the distribution of length of stay in the `ER` before the patients were `discharged alive`.

4. **Sort**: Users can sort the sequences with the same parent in different ways: by the number of records that the bars represent (tallest bar on top) (Figure 3.4) or by the average time to the end (longest time on top) (Figure 3.8). The default is to sort by number of records.

5. **Integration with LifeLines2**: LifeFlow can function as a standalone tool, but combining it with LifeLines2 facilitates exploration by allowing users to review individual records as details on demand [112]. By clicking on any event bar, users select all records that are included in that bar (Figure 3.6). Selected records are highlighted in the LifeLines2 view. Users can then choose to keep only the selection and remove everything else, or vice versa. In a symmetrical fashion, selecting a record in the LifeLines2 view highlights the pattern contained in that record in the LifeFlow view, allowing the users to

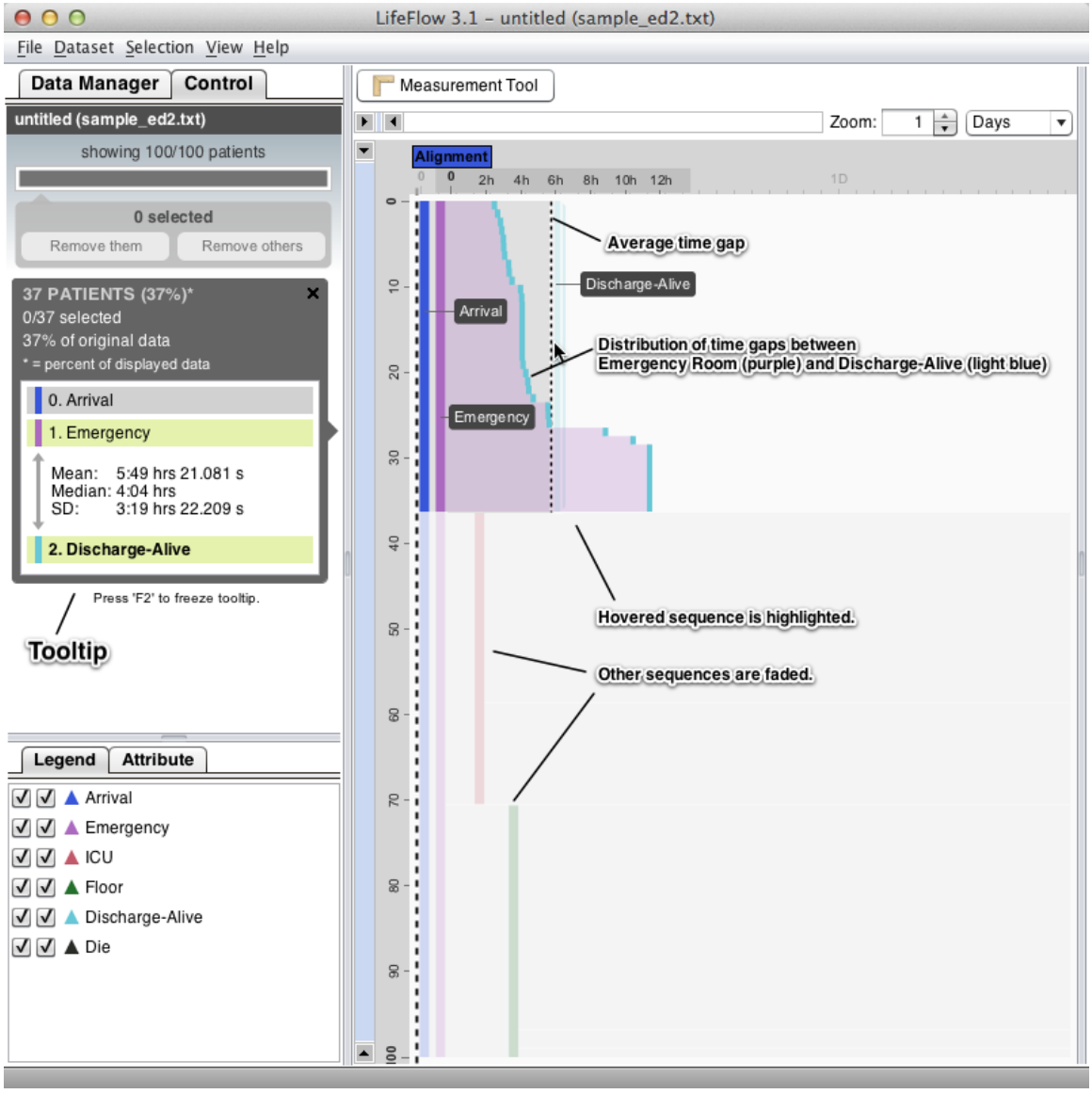

Figure 3.5: When users move the cursor over an event bar or gap between event bars, LifeFlow highlights the sequence and shows the distribution of time gaps. Labels help the users read the sequences easier. A tooltip also appears on the left, showing more information about the time gap. In this Figure, the distribution shows that most patients were discharged within 6 hours and the rest were mostly discharged exactly after 12 hours. (Please note that this is only a sample dataset and does not reflect the real performance of any hospital.)

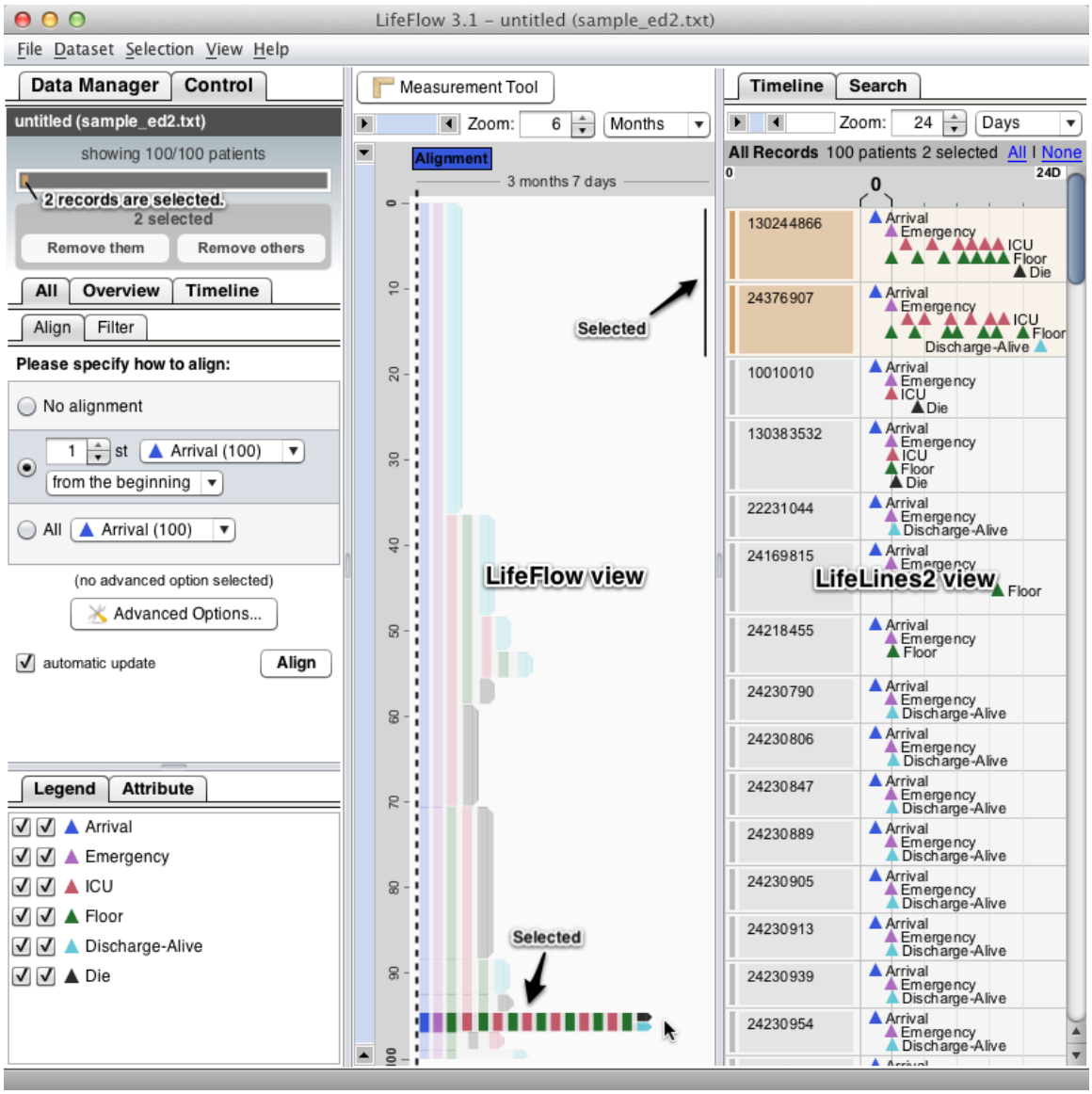

Figure 3.6: Here LifeFlow is used side-by-side with LifeLines2 so that individual records can be reviewed by scrolling. When a user clicks on a sequence in LifeFlow, the sequence is highlighted and all corresponding records are also highlighted and moved to the top in LifeLines2, allowing the user to examine them in more detail. In this example, a user noticed an uncommon pattern of frequent transfer back and forth between `ICU` (red) and `Floor` (green), so he selected those patients to see more detail.

find other records that contain the same sequence.

6. **Align**: As mentioned earlier, LifeFlow allows users to choose any event type to be the *alignment* point. Figure 3.7 shows LifeFlow with alignment. The vertical dashed line marks the aligned event. The left and right side are what happened before and after the alignment point, respectively. Now one can see that in this dataset, patients most often come to `ICU` from `Floor`. After that, they often died.

7. **Include/Exclude event types**: Using the legend on the left side of the screen users can check or uncheck event types to include or exclude them from the sequences. This simple functionality allows powerful transformations of the display to answer questions. For example, in Figure 3.8, the user unchecked all event types except `Arrival`, `Discharge-Alive` and `Die`, i.e. the beginning and end of hospital visits. All other events that could occur during a visit are ignored and LifeFlow regenerated to show only those three event types, allowing rapid comparisons between the patients who died and survived in terms of number of patients and average time to discharge.

3.6 User Study

I conducted a user study to investigate if LifeFlow was easy to learn, and if users could use the interface efficiently to answer representative questions. Another goal was to observe what strategies users chose and what problems they would encounter, and gather feedback and suggestions for further improvement. The dataset

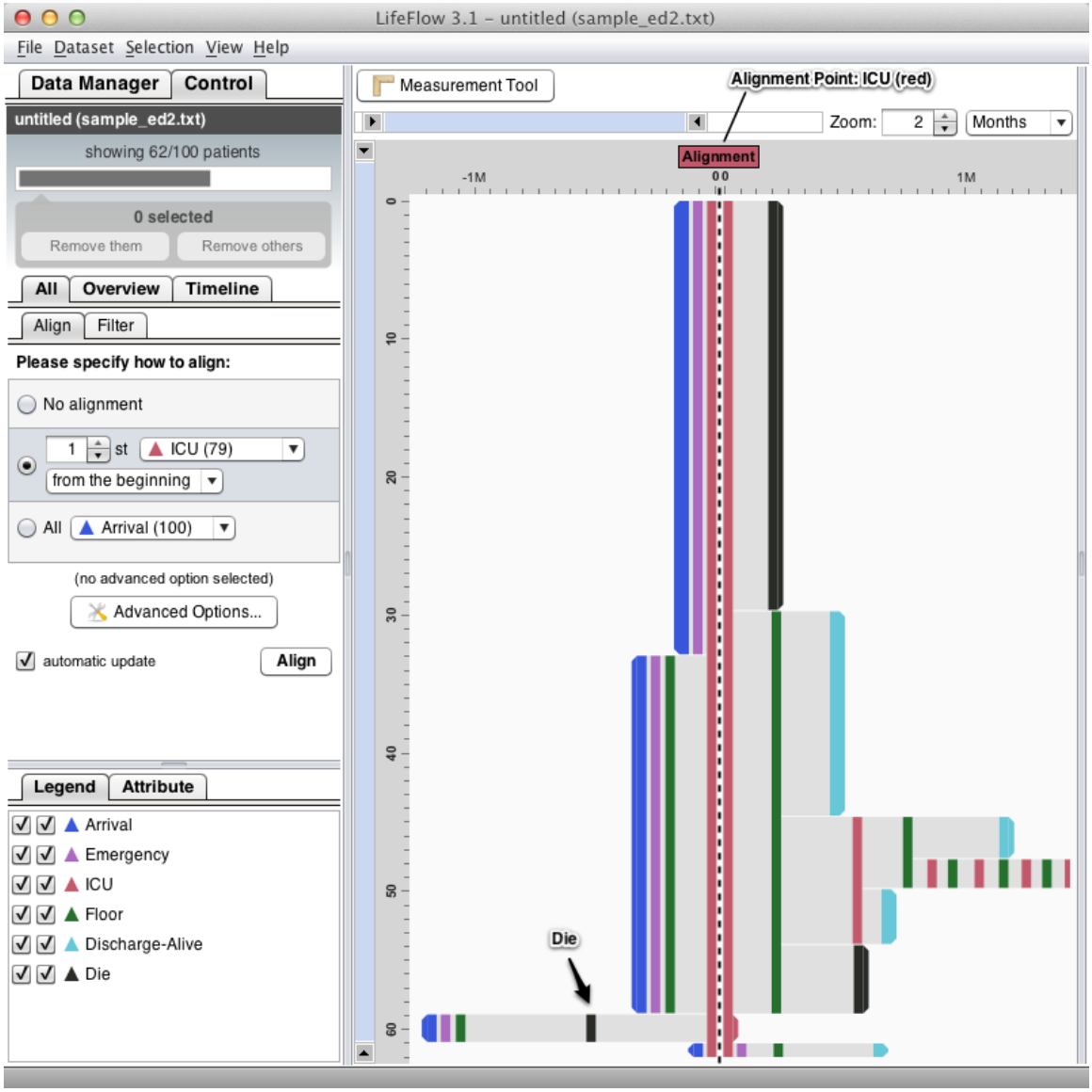

Figure 3.7: The same data with Figure 3.4 was aligned by `ICU`. The user can see that the patients were most likely to `die` after a transfer to the `ICU` than any other sequence because the black bar is the tallest bar at the end. Also, surprisingly, two patients were reported `dead` (see arrow) before being transferred to `ICU`, which is impossible. This indicates a data entry problem. (Please note that this is only a sample dataset and does not reflect the real performance of any hospital.)

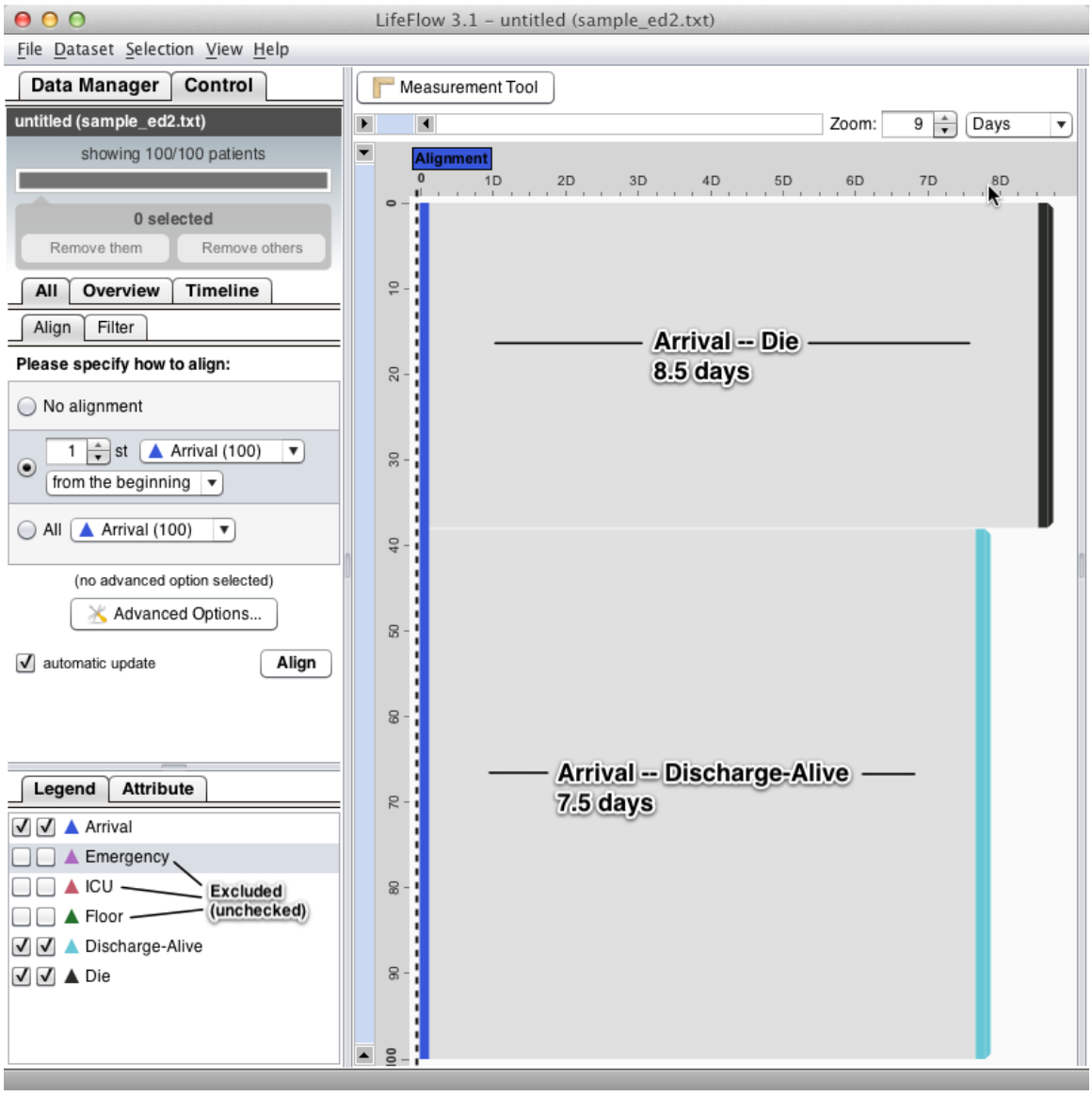

Figure 3.8: Using the same data with Figure 3.4, the user excluded all event types except `Arrival`, `Discharge-Alive` and `Die`, i.e. the beginning and end of hospital visits. All other events are ignored, allowing rapid comparisons between the patients who died and survived in terms of number of patients and average time to discharge. Patients who survived were discharged after 7.5 days on average while patients who died after 8.5 days on average. (Please note that this is only a sample dataset and does not reflect the real performance of any hospital.)

in this study included 91 records of hospital patient transfer (a subset of real de-identified data, which included known anomalies to see if participants could find them). Because medical professionals have very little availability, they are difficult to recruit for a user study. The data used in the study is simple enough to be understood by students, therefore, the participants in this study were graduate students (5 male and 5 female) from various departments of the University of Maryland. None of them were members of the LifeFlow development team.

3.6.1 Procedure

Training consisted of a 12-minute video and five training questions. When the participants could answer the questions correctly, they could start the study tasks. The order of the tasks was randomly permuted across participants. The tasks were representative of the questions proposed by domain experts during the case study, and designed to test the usability of the main interaction techniques of LifeFlow. Participants were encouraged to think aloud while performing the tasks. For the first 14 tasks, observers recorded completion time and errors, if any. Because the participants needed time to understand the tasks, they were instructed to read the task description, indicate when they are ready and then start the timer.

3.6.2 Tasks

3.6.2.1 Tasks 1-9: Simple Features

The first 9 tasks required understanding the LifeFlow visualization and using simple interactive features such as tooltips or zooming. The tasks included questions about frequent sequences (e.g. "Where did the patient usually go after they `arrived`?", "What is the most common pattern?"), number of records in specified sequences (e.g. "How many patients went from `arrival` directly into `ICU`?"), time between events (e.g. "How long is the average time from `ER` to `ICU`?") and comparison between sequences (e.g. "After `arriving` to the `ER`, patients might be transferred to Floor or the `ICU`. Were the patients transferred from the `ER` to the `ICU` faster than from the `ER` to the `Floor`?").

3.6.2.2 Task 10-14: Advanced Features

Four tasks required using advanced features, such as alignment or using LifeFlow and LifeLines2 in combination (e.g. "Usually, where were the patients before they were admitted to the `ICU` for the first time?", "Retrieve the IDs of all patients with this transfer pattern." or "How many patients had the same transfer pattern as patient no.10010010?")

3.6.2.3 Task 15: Overall analysis and finding anomalies

In the last task, I asked the participants to imagine themselves as a manager who was trying to evaluate the performance of the hospital. I gave them 10 minutes

to find any surprising, exceptional or impossible sequences that might indicate a problem in data entry or in hospital procedures, and explain why they thought it was a problem. I told them to report as many insights as they could in 10 minutes. I had planted three (realistic) data anomalies:

1. A few patients who died before being transferred to `ICU`.

2. Patients who bounce back and forth between the `ICU` and `Floor` several times.

3. Patients who stayed in the `Floor` for 3 months.

3.6.3 Results

3.6.3.1 Tasks 1-14

The participants were able to perform the simple and advanced tasks quickly. They were able to use the interactions to adjust the visualization and retrieve information that was not presented in the initial view. The average ± SD completion time for the simple and advanced tasks were 14.9 ± 12.7 seconds and 15.8 ± 12.5 seconds, respectively. Please note that the participants were also narrating their actions while performing the tasks, which might have slowed them down. Only one participant made one mistake in a complex task: while retrieving the IDs of all patients who were transferred with a particularly long sequence she misread a color and could not find the correct sequence. However, she knew what she needed to do to retrieve the IDs after the sequence was found.

3.6.3.2 Task 15

Eight out of ten participants were able to detect all three anomalies. Two participants reported only the first two anomalies, but when I directed their attention towards the third anomaly, they explained that they had noticed it but did not think it was abnormal because some patients (e.g., cancer patients) might stay in the hospital for a long time. In addition, they also provided insight about other sequences that were possible, but undesirable from a manager's perspective, such as instances when patients who died in the Floor unit. They also reported surprising patterns, such as many patients discharged alive directly from the `ICU`. I also observed the following common strategies:

1. Examine things that catch the eye first.
2. Scan all sequences systematically top to bottom: When they saw the normal sequence, e.g., `ICU` to `Floor`, they noted that this is good and moved on to the next sequence.
3. Align by each type of event.
4. Hovering over bars with the mouse to explore distribution and detect outliers from the distribution.
5. See more detail in LifeLines2: Although I did not display LifeLines2 at the beginning of this task, several participants opened it to use the combined view.

All participants used strategies 1-3 consecutively. Three participants also followed with strategy 4. Four participants followed with strategy 5.

3.6.3.3 Debriefing

During the debriefing, typical comments included: "The tool is easy to understand and easy to use.", "very easy to find common trends and uncommon sequences", "The alignment is very useful.", "In LifeFlow, it is easier to see the high level picture. With LifeLines2, you can check individuals. LifeFlow provides a great summary of the big picture." Common suggestions for improvement included increasing the bar width to make it easier to select and reorganizing the tooltip to make it more readable. Two participants also asked to analyze the data from their own research with LifeFlow.

3.6.4 Summary

Results suggest that users can learn to use LifeFlow in a short period of time and that LifeFlow's overview of the data allows them to understand patterns and find anomalies. There were several common strategies used when performing data analysis but not every participant used all strategies, which indicated the need for a framework to support data analysis.

3.7 Advanced Features

By collaborating with the domain experts in several case studies, I have developed a better understanding of users' needs and limitations of LifeFlow. Their feedback guided me to invent several advanced features along the way to enhance LifeFlow's ability to support more complicated tasks. These features are also gen-

eralizable and found to be useful for tasks in different domains.

1. **Including Non-Temporal Attributes**: Records also usually contain *non-temporal attributes*, e.g., patient's gender or category of traffic incidents. These attributes can be categorized into *record attributes* and *event attributes*.

   (a) *Record attribute* is attached to each record of event sequences, for example, the gender of the patient or the county in which the patient lives.

   While LifeFlow does not focus on displaying these attributes, it allows users to select record attributes and group records by the selected attribute before the sequences are aggregated. This feature was motivated by a case study with the Center for Advanced Transportation Technology (CATT) Lab where the users have 200,000 traffic incident logs from several traffic agencies and want to compare their performance, so we wanted to be able to group the records by agencies and other record attributes

   LifeFlow in Figure 3.9 groups traffic incident records by agency before aggregating by sequence, therefore allowing simple comparison between agencies. Several attributes can also be used in combination. Please see the case study in Section 6.3 for further analysis of Figure 3.9 and more examples of attributes.

   (b) *Event Attribute* is attached to each event. For example, the name of the attending physician is a property of the event `Arrival` in a patient record because the physician may be different for each visit.

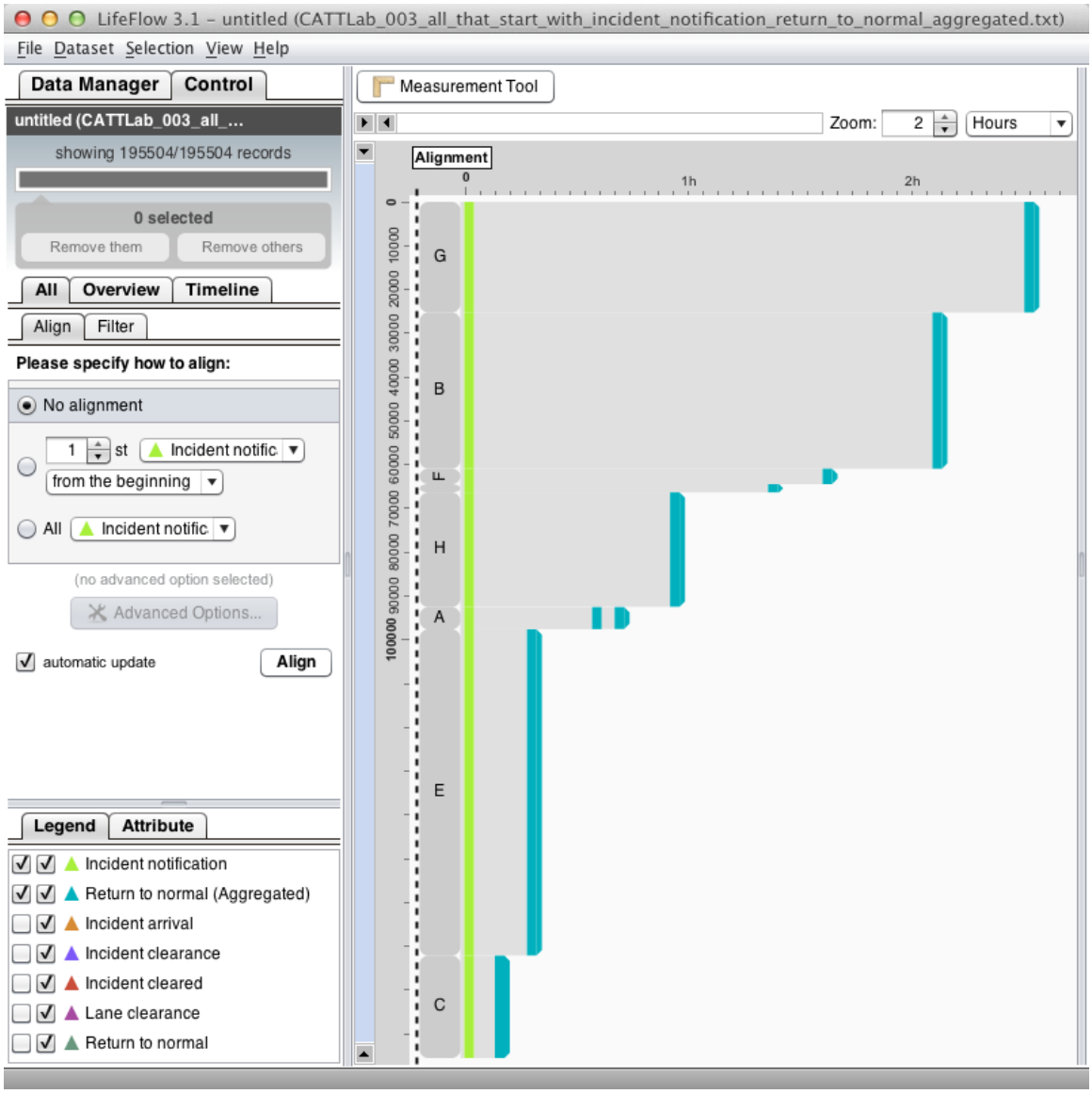

Figure 3.9: LifeFlow with traffic incidents data: The incidents are separated by agencies (A-G). Only `Incident Notification` and `Return to normal (aggregated)` events are shown. Other events are hidden. The agencies are sorted by a simple measure of agency performance (average time from the beginning to the end). Agency C seems to be the fastest to clear its incidents, followed by E, A, H, D, F, B and finally G.

I received a request from my physician partners that they were also interested in the event attributes. They wanted to know the common diagnoses of the patients for each visit and hoped to see a summary of them. Therefore, I implemented a way to quickly show the summary by adding "Show attributes summary..." feature to the right-click menu. Users can right-click at any event bar and open the summary table that summarize all event attributes and also record attributes (Figure 3.10. Selecting any row in the summary table will also select the records in LifeFlow and LifeLines2 view, allowing users to drill down to the detail.

2. **Custom Attributes**: Users can assign custom attributes to selected records. This is found useful when users detect records with a pattern of interest and want to separate them from the rest. They can assign a custom attribute using the attribute for grouping. For example, user assigned an attribute "Status" as "Alive" and "Dead" to the patients who had pattern `Arrival` → `Discharge-Alive` and `Arrival` → `Die` in Figure 3.8, respectively. After that, the user included other event types that were excluded earlier and chose to group records by attribute "Status" to show patterns of "Alive" and "Dead" patients (Figure 3.11).

3. **Selection from distribution**: Users sometimes find interesting patterns from the distribution and want to select only part of the sequence to see more detail. For example, users can see from the distribution in Figure 3.5 that many patients were release exactly after twelve hours. Users can select

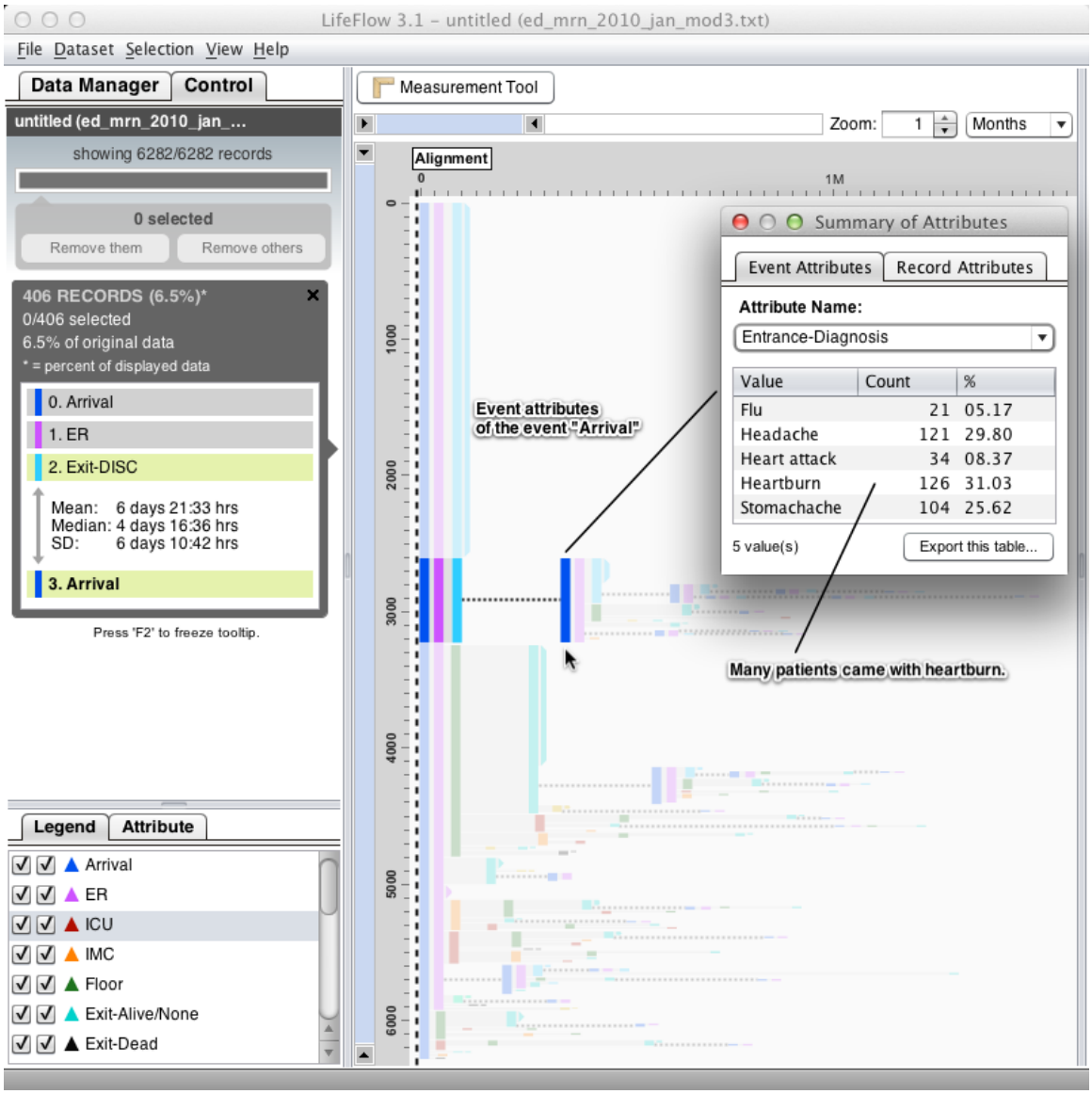

Figure 3.10: User right-click at the second `Arrival` event and select "Show attributes summary..." to bring up the summary table, which summarizes the common diagnoses. (Please note that this is only a sample dataset and does not reflect the real performance of any hospital.)

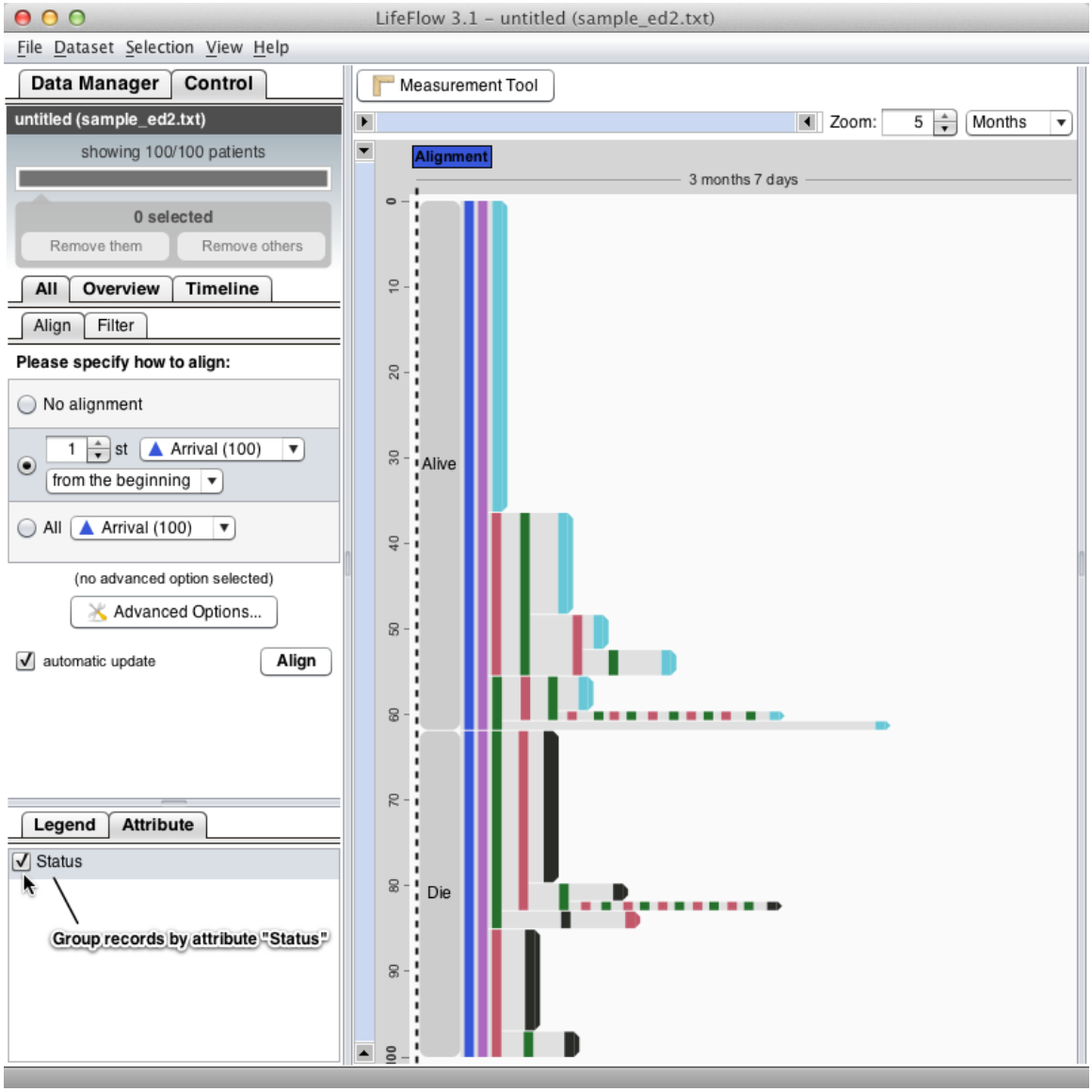

Figure 3.11: User assigned attribute "Status" as "Alive" and "Dead" to the patients who had pattern `Arrival` → `Discharge-Alive` and `Arrival` → `Die` in Figure 3.8, respectively. After that, the user included other event types that were excluded earlier and chose to group records by attribute "Status" to show patterns of "Alive" and "Dead" patients. Notice that the majority of the dead patients were transferred to `Floor` first and later transferred to the `ICU` before they died. (Please note that this is only a sample dataset and does not reflect the real performance of any hospital.)

only those patients to investigate more.

This feature also enhances the functionality of LifeFlow as a query tool. For example, to find patients who were transferred from `ICU` to `Floor` within two days, users can show the distribution of patients from `ICU` to `Floor` and draw a selection of patients whose transfer gaps were within two days.

4. **Measurement**: One limitation of LifeFlow is that it displays only time gaps between consecutive events. For example, for a sequence `A` $\rightarrow$ `B` $\rightarrow$ `C`, the horizontal gaps in the visualization only represent the mean/median time gaps for `A`$\rightarrow$`B` and `B`$\rightarrow$`C`, but the time gap between non-consecutive events (`A`$\rightarrow$`C`) is not shown. Users cannot sum the two gaps together because a sum of the mean/median time gaps between consecutive events (`A`$\rightarrow$`B` + `B`$\rightarrow$`C`) is not always equal to the mean/median time gap `A`$\rightarrow$`C`. See this counterexample:

$$(A, 1) \rightarrow (B, 2) \rightarrow (C, 3)$$
$$(A, 1) \rightarrow (B, 3) \rightarrow (C, 5)$$
$$(A, 1) \rightarrow (B, 6) \rightarrow (D, 9)$$
$$(A, 1) \rightarrow (B, 7) \rightarrow (D, 8)$$

$$\bar{X}(A \rightarrow B) = 3.5,\ \bar{X}(B \rightarrow C) = 1.5,\ \bar{X}(A \rightarrow C) = 3$$

To overcome this limitation, the measurement tool was developed to help users retrieve an accurate time gap between any two non-consecutive events. Users can select any event bar as start and end points and LifeFlow will display the

mean, median, SD on the tooltip and overlay the distribution at users' request. Users can also make a selection from this distribution (Figure 3.12).

5. **Displaying all event bars with equal height**: When data includes a large number of sequences, it could be difficult to review the rare sequences because they are represented with very thin bars. This option displaying all leaf nodes using equal height, regardless of the number of records, makes it easier to review and select rare sequences.

6. **Episode**: Some event sequences can be split into episodes. For example, a patient's medical history may consist of multiple visits. My physician collaborator wants to be able to see each visit easily. Therefore, I revised the visual representation to display a clear separation between episodes using dotted lines and allow users to select an event type that defines the beginning of an episode from the user interface. For example, in Figure 3.13, the physician selected `Arrival` event to separate hospital visits.

7. **Rank-by-feature**: The gap between events indicates how long it took to change from the previous event to the next event. With many sequences and gaps displayed on the screen, finding the longest time gap to locate a bottleneck in the process (for example, longest transfer time in the hospital) can be difficult, especially when some sequences are rare and displayed as very small bars. To support this task, I adopted the rank-by-feature technique and included a list of all gaps between events, sorted by chosen criterion (Figure 3.14). User can choose to sort by mean, median, minimum, maximum and

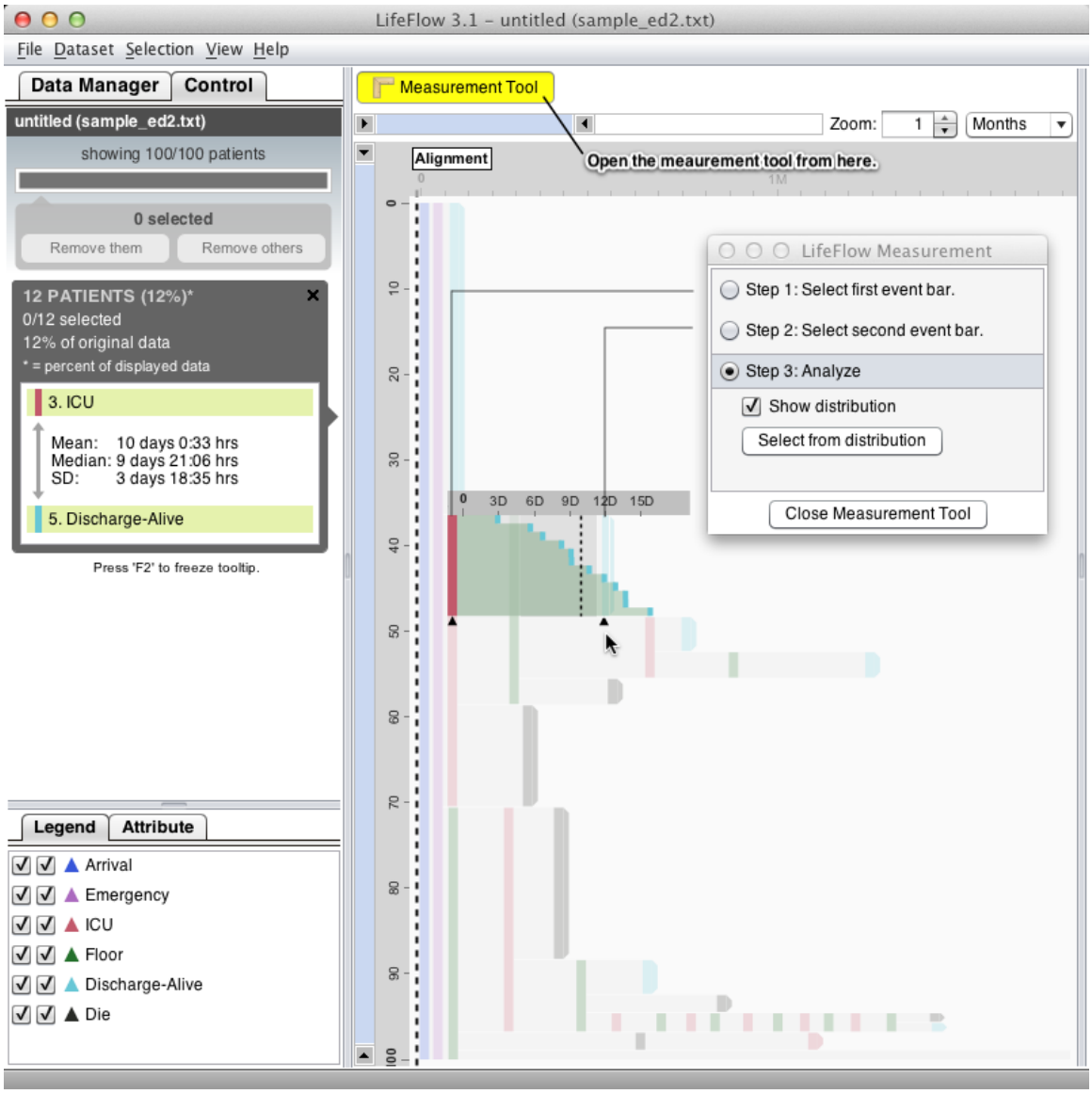

Figure 3.12: User used the *measurement tool* to measure the time from `ICU` (*red*) to `Discharge-Alive` (*light blue*). The tooltip shows that it took about ten days. The distribution of time gap is also displayed.

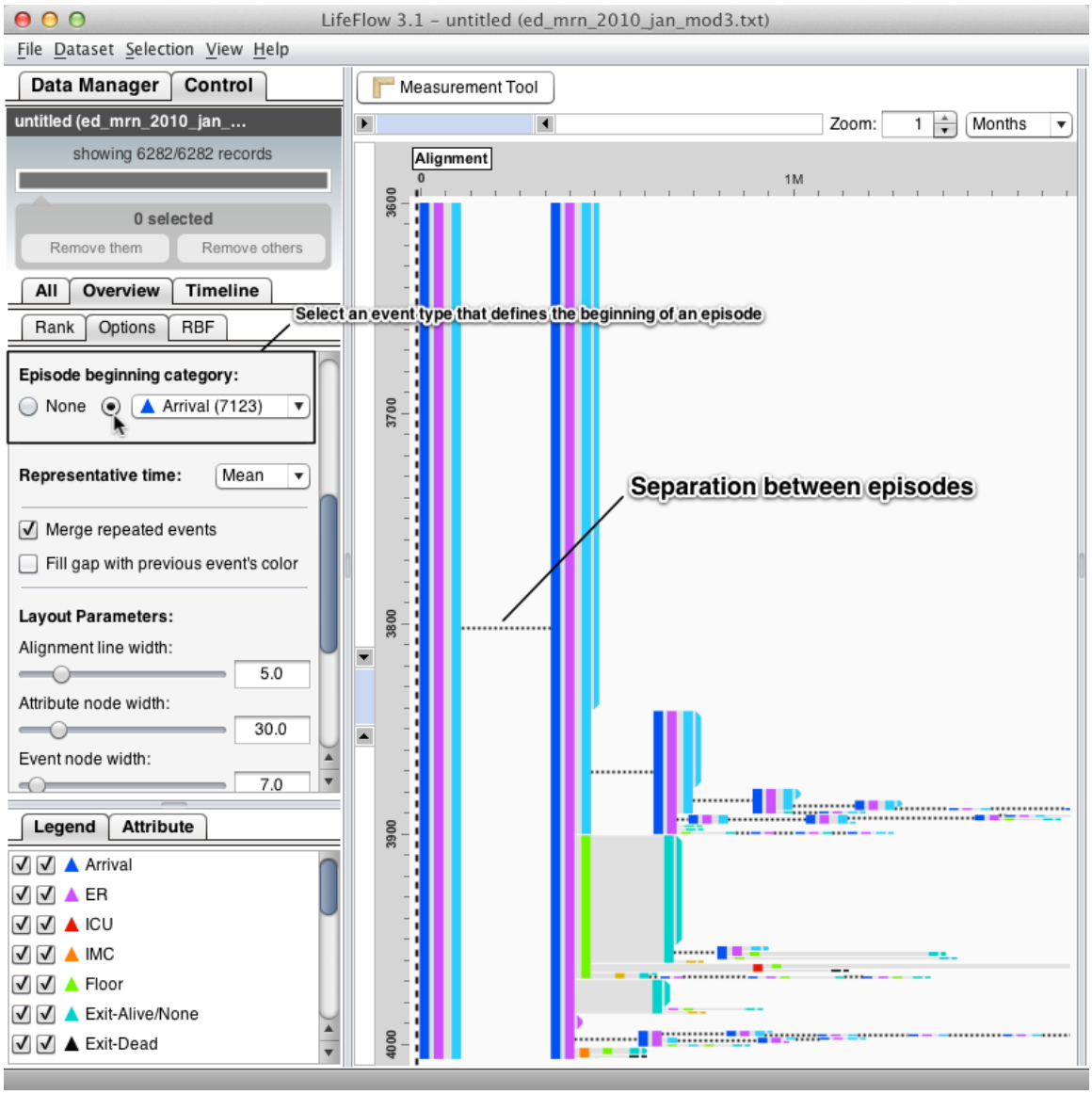

Figure 3.13: The sequences of medical records are split into *episodes* using an event type `Arrival` to define the beginning of an episode. Dotted lines show separation between episodes.

standard deviation of the gaps. Once user selects any gap from the table, the visualization will animate and zoom to the selected gap.

3.8 Summary

Analyzing large numbers of event sequences is an important and challenging task. Lacking ability to see all records on the screen at once makes it difficult to discover patterns. I introduce a new scalable visualization called *LifeFlow* that provides an overview of event sequences to support users' exploration and analysis. The LifeFlow visualization is based on aggregating data into a data structure called a *tree of sequences*, which preserves all sequences and temporal spacing within sequences. I report on a short-term user study with ten participants which confirmed that even novice users with 15 minutes of training were able to learn to use LifeFlow and rapidly answer questions about the prevalence of interesting sequences, find anomalies, and gain significant insight from the data. After the short-term study, I have worked with several domain experts in long-term case studies which led to deeper understanding of users' need and guided the improvement and extensions of LifeFlow to overcome its limitations and support more complex tasks.

Although it was inspired by a case study in the medical domain, LifeFlow can be applied to many other fields, where event sequences are the main focus, such as student progress analysis, usability study or web log analysis, and human activities log analysis in general. I have conducted several case studies that demonstrate LifeFlow's applications and report them in Chapter 6.

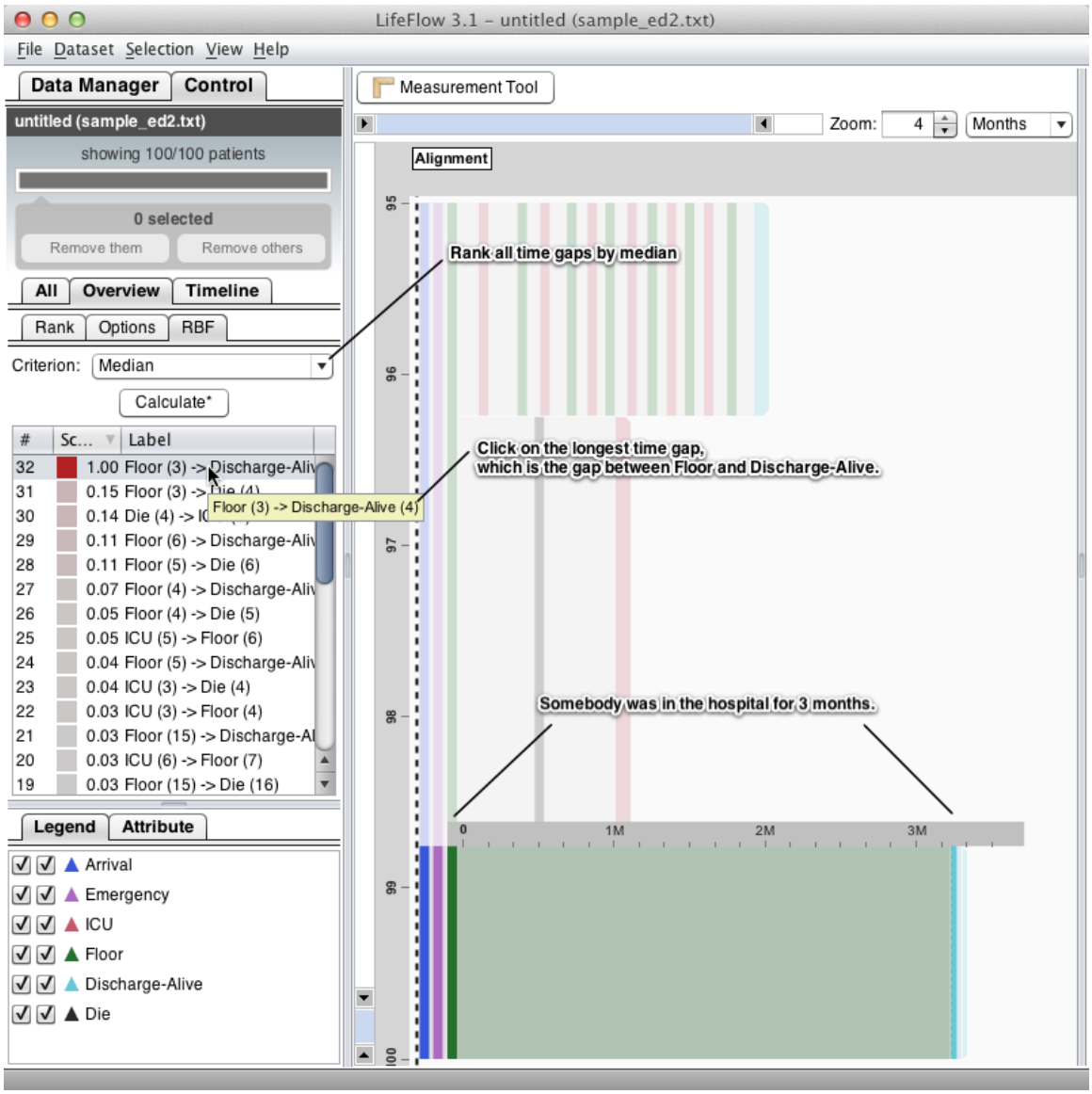

Figure 3.14: The sequences of medical records are broken into episodes using an event type `Arrival` to define the beginning of an episode. Dotted lines show separation between episodes.

Chapter 4

Querying Event Sequences by Similarity Search

4.1 Introduction

Querying event sequences to answer specific questions or look for patterns is an important activity. Such activities can be utilized in, for example, finding patients who were transferred from an `emergency room` to the `ICU (Intensive Care Unit)` and `died`, incidents in which the police `arrived` 2 hours after `incident notification` or a PhD student who `proposed` a dissertation topic twice before `graduated`.

4.1.1 Example of Event Sequence Query

My physician partners in the Emergency Department at the Washington Hospital Center are analyzing sequences of patient transfers for quality assurance. One of their interests is the monitoring of *bounce backs*, which occurred when a patient's level of care was decreased then increased 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 too many hours to create the formulas). I 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.)

4.1.2 Motivation for Similarity Search

Specifying temporal queries in SQL is difficult even for computer professionals specializing in such queries. I 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 [115, 116, 31], 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 my knowledge, existing event sequence query interfaces 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, an exploratory search [124, 139], 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.

A *similarity search* interface has been used 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 ranked results can provide more flexibility than exact match results, allowing users to capture the "just off" cases. The key behind similarity search 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.

4.1.3 Chapter Organization

In this chapter, I describe how I developed a new similarity search interface, *Similan*, and similarity measure for event sequences, *Match & Mismatch (M&M) measure*. Some parts of this chapter have been published in [144] and [143].

Section 4.2 introduces the first version of M&M measure and Similan user interface. The M&M measure defines similarity as a combination of the time differences between pairs of events and the number of mismatches. Similan allows the users to select an existing record from the database as a target record (query) and provides search result visualization. An evaluation of the first version is reported in Section 4.3.

Section 4.4 introduces the second version. To address the limitations of the first version, *Similan2*, allows the users to draw an example of an event sequence by placing events on a blank timeline and search for records that are similar to their example and improves how events are visualized on the timeline. The M&M measure v.2 supports richer and more flexible definitions of similarity and improves performance using dynamic programming.

Section 4.5 reports a controlled experiment compared exact match (*Similan2*) and similarity search interfaces (*LifeLines2* [131]). I summarize the advantages and disadvantages of each interface and suggest a hybrid interface combining the best of both in Section 4.6.

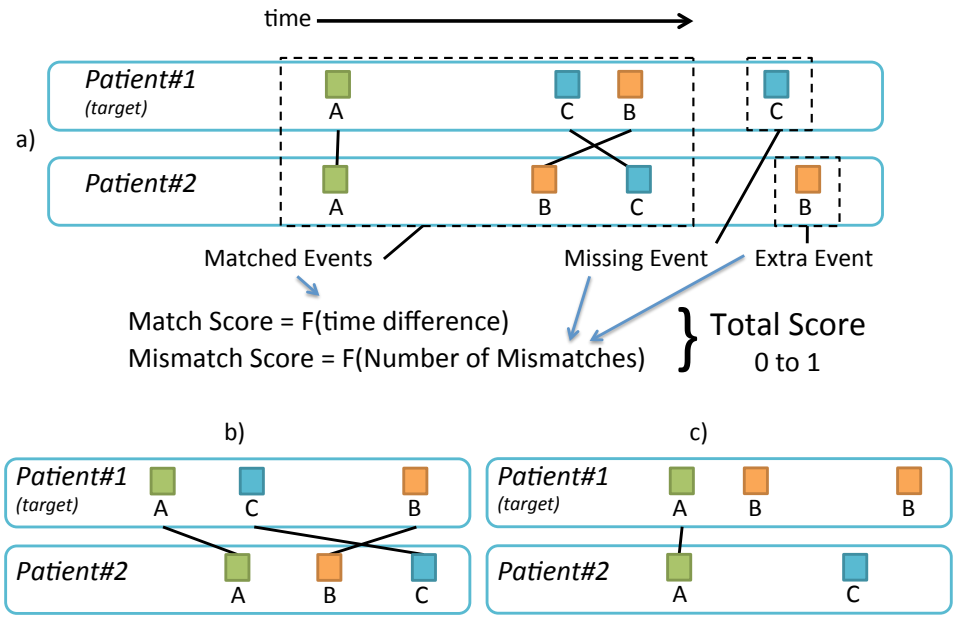

Figure 4.1: *(top)* The M&M measure *(bottom-left)* High time difference (low match score) but no mismatch (high mismatch score) *(bottom-right)* Low time difference (high match score) but high mismatches (low mismatch score)

4.2 Similan and the M&M Measure: The First Version

4.2.1 Introduction to the Match & Mismatch (M&M) Measure

The *M&M* measure is based on aligning temporal data by sentinel events [131], then matching events in the query record with events in the compared records. Since there can be many possible ways to match the events between the two records, I define an event matching process for the M&M measure, which will be explained in Section 4.2.3. After the matching is done, the M&M measure is a combination of two measures:

The first measure, *match score*, is for the *matched* events, events which occur both in the target record and the compared record. It captures the time difference between events in the target record and the compared record.

The second measure, *mismatch score*, is for *missing* or *extra* events, events which occur in the target record but do not occur in the compared record, or vice versa. It is based on the difference in number of events in each event type between the two records.

Match and mismatch score are combined into *total score*, ranging from 0.01 to 1.00. For all three scores, a higher score represents higher similarity.

4.2.2 Description of the User Interface: Similan

Given two event sequences, the M&M measure returns a score which represents the similarity between that pair of records. However, the score alone does not help the users understand of why records are similar or dissimilar. Also, the M&M measure can be adjusted by several parameters. A tool to assist users in understanding the results and customize the parameters is needed. To address these issues, Similan was developed to provide a visualization of the search results to help users understand the results, and an interface that facilitates the search and parameter customization. Similan was written in C# .NET using the Piccolo.NET [14] visualization toolkit. The design of Similan follows the Information Visualization Mantra: overview first, zoom and filter, details on demand [112].

4.2.2.1 Overview

Similan consists of 4 panels: main, comparison, plot and control, as shown in Figure 4.2. Users can start from selecting a target record from the main panel.

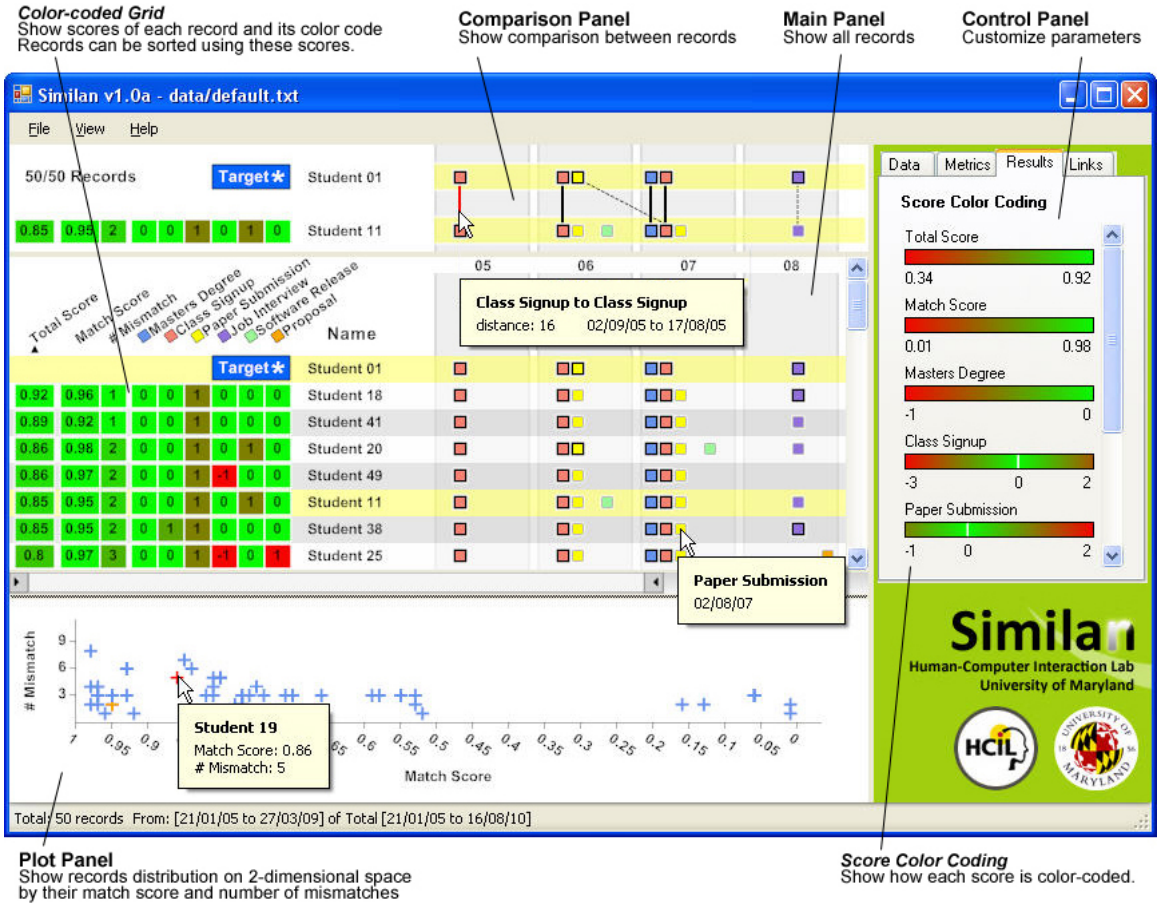

Figure 4.2: A screenshot of Similan, the predecessor of Similan2. Users can start by double-clicking to select a target record from the main panel. Similan will calculate a score that indicates how similar to the target record each record is and show scores in the color-coded grid on the left. The score 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. Although Student 41 and Student 18 both have one missing paper submission, Student 41 has a lower match score, therefore, Student 18 has a higher total score.

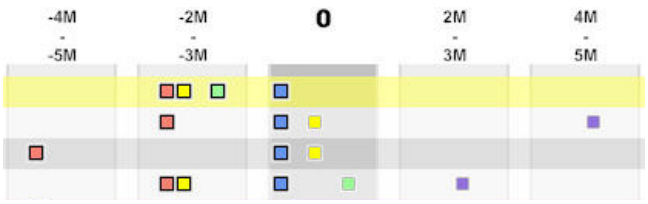

Figure 4.3: Relative Timeline: Time scale is now relative to sentinel events (blue). Time zero is highlighted in dark gray.

After that, the main and plot panels give an overview of the similarity search result. Filtering and ranking mechanisms help users narrow down the search result. Users then can focus on fewer records. By clicking on a particular record, the comparison panel shows relationships between that record and the target record on demand. Moreover, mouse hovering actions on various objects provide details on demand in the form of tooltips.

4.2.2.2 Events and Timeline

Colored squares are used to represent events. Each color represents a event type (category). Users can customize the colors and check the checkboxes in the control panel (Figure 4.4) to select interesting event types. The number of events in each event type is displayed behind the event type name.

Similan's timeline is not a continuous timeline but divided into bins. The bin interval is automatically calculated by the application window size and total time range of the data. As shown in Figure 4.2, the timeline is divided into years (05, 06, 07, 08). In each bin, events are grouped by event types and placed in the same order. Maintaining the same order allows for visual comparison between records.

Each record is vertically stacked on alternating background colors and iden-

tified by its name on the left (see Figure 4.2). Ranking scores (more details in Section 4.2.2.4) appear on the left hand side before the name. Events appear as colored squares on the timeline. By default, all records are presented using the same absolute time scale (with the corresponding years or month labels displayed at the top) and the display is sized so that the entire date range fits in the screen.

A double-click on any record marks that record as a target record. A target mark will be placed in front of the target record instead of a ranking score. Clicking on any record selects that record as a compared record. Both the target record and compared record will be highlighted. Users can move the cursor over colored squares to see details on demand in the form of tooltips. Also, zooming on the horizontal axis and panning are possible using a range slider provided at the bottom of the main panel.

4.2.2.3 Alignment

Users can select a sentinel event type from a drop-down list as shown in Figure 4.4. By default, the sentinel event type is set to none. When the sentinel event type is selected, the time scale will change from an absolute time, i.e. real time, into a relative time. The sentinel event becomes time zero and is highlighted (Figure 4.3).

4.2.2.4 Rank-by-feature

Similan is inspired by the idea of *rank-by-feature* from Hierarchical Clustering Explorer (HCE) [108]. These following ranking criteria are derived from the M&M

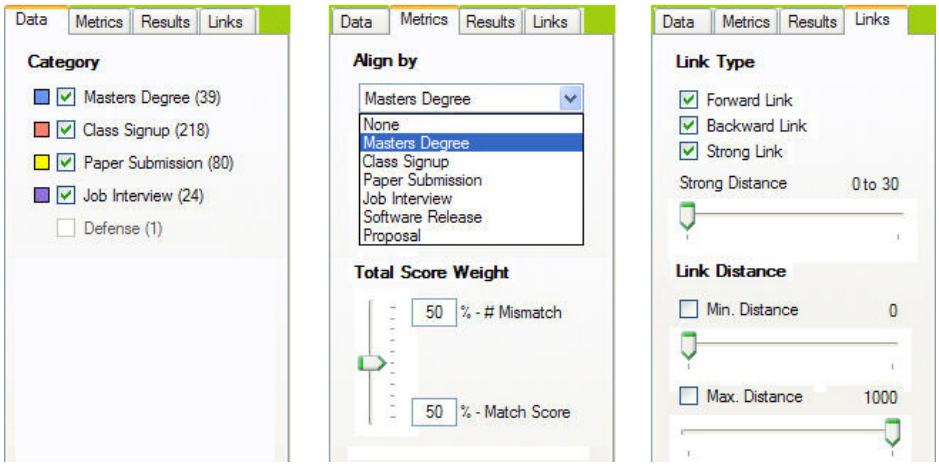

Figure 4.4: Control Panel: (left) Legend of event types (categories) (middle-top) Users can choose to align events by selecting sentinel event type. (middle-bottom) Weight for calculating total score can be adjusted using slider and textboxes. (right) Links in comparison panel can be filtered using these parameters.

measure proposed in this paper.

1. **Total Score** *ranging from 0.01 to 1.00*

   Total score is the final output of the M&M measure. It is a weighted sum of match and mismatch scores. The weight can be adjusted and users can see the result in real-time (Figure 4.4).

2. **Match Score** *ranging from 0.01 to 1.00*

   This is a score derived from the distance (time difference) between matched events. I choose to display match score instead of distance because the distance is a large number, so it can be difficult to tell the difference between two large numbers and understand the distribution.

3. **Number of Mismatches (#Mismatch)** *ranging from 0 to n*

   This is the total number of missing and extra events compared to the target record. The #mismatch is shown instead of the mismatch score because it is

more meaningful to the user. Furthermore, I break down the #mismatch into event types. Positive and negative values correspond to the number of extra and missing events, respectively.

Users can click on the ranking criteria on the top of the main panel to sort the records. By clicking on the same criteria once more, the order is reversed. A triangle under the header shows current ranking criterion. Legends in the control panel show the range of each ranking score and how they are color-coded. (See Figure 4.2.)

4.2.2.5 Scatterplot

In addition to displaying results as a list in the main panel, Similan also visualizes the results as a scatterplot in the plot panel (Figure 4.2). Each record is represented by a "+" icon. Horizontal axis is the match score, while vertical axis is the number of mismatches (#mismatch). Records in the bottom-left area are records with high match score and low number of mismatches, which should be considered most similar according to the M&M measure.

Moving the cursor over the + icon will trigger a tooltip to be displayed. Clicking on a + will set that record to be the compared record and scroll the main panel to that record. Users can also draw a region on the scatterplot to filter records. The main panel will show only records in the region. Clicking on the plot panel again will clear the region and hence clear the filter.

4.2.2.6 Comparison

The comparison panel is designed to show similarity and difference between the target record and the compared record. Lines are drawn between pairs of events matched by the M&M measure. Line style is used to show the distance value. *Strong links*, or links with short distance, are shown as solid lines. *Weak links*, or links with large distance, are shown as dashed lines. Events without any links connected to them are missing or extra events. Users can adjust the distance threshold for strong links in the control panel. (See Figure 4.4.) Moving the cursor over a link will display a tooltip showing the event type, time of both events and distance.

Furthermore, users can filter the links by using the filters (Figure 4.4). Users can filter by setting the minimum and/or maximum distance. By selecting link types, only the selected types are displayed. *Strong links* are links with a distance in the range specified by the slider. *Forward Links* are links which are not strong links and the event in the target record occurs before the event in the compared record, whereas, *Backward Links* are the opposite.

4.2.3 The Match&Mismatch (M&M) Measure

This section explains the M&M measure in more detail. The base idea is that similar records should have the same events and the same events should occur almost at the same time. Therefore, the M&M measure uses the time difference and number of missing and extra events as the definition of similarity.

The notation below is used to describe a record of event sequence, which is a

series of events (t, c). The i-th event in the record is denoted by x_i or (t_i, c_i).

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

4.2.3.1 Matching

The first step is to match the events in the target record with events in the compared record. There can be many possible ways to match the events into pairs. Therefore, I define a distance function based on a sum of time difference to guide the matching. The matching which produces the minimum distance (time difference) will be selected. Note that the distance from the M&M distance function is not the final result of the M&M measure, but only part of it. This distance is later converted to a match score.

M&M Distance Function I first define a distance function between each pair of events, as follows:

$$d((t, c), (u, d)) = \begin{cases} |t - u| & \text{if } c = d \\ \infty & \text{if } c \neq d \end{cases} \tag{4.1}$$

The distance is computed from the time difference if both events have the same type. The granularity of time difference (years, months, days, etc.) can be set. Matching between different event types is not supported, so I set the distance between every pair of events that has different event types to infinity.

A distance function between the target record X and the compared record Y

$$\begin{aligned} X &= \{(t_1, c_1), (t_2, c_2), ..., (t_m, c_m)\} \\ Y &= \{(u_1, d_1), (u_2, d_2), ..., (u_n, d_n)\} \end{aligned}$$

is described as the following:

$$\begin{gathered} D(X, Y) = min \sum_{i \in [1,m], j \in [1,n]} d(x_i, y_j) \\ \text{each value of } i \text{ and } j \text{ is used exactly once.} \end{gathered} \tag{4.2}$$

A distance function between two records is calculated by matching events from the two records into event pairs and summing up the distances $d(x_i, y_j)$ between each pair. However, this distance function works only when the number of events in both records are equal because it requires a perfect match between the two records. Also, even when the number of events are equal, this case can occur:

$$\begin{aligned} X &= \{(t_1, \text{“}A\text{”}), (t_2, \text{“}A\text{”}), (t_3, \text{“}B\text{”})\} \\ Y &= \{(u_1, \text{“}A\text{”}), (u_2, \text{“}B\text{”}), (u_3, \text{“}B\text{”})\} \\ &\text{“A”,“B”} \in \text{event types} \end{aligned}$$

This will certainly create at least one pair of different type events, which is not preferred. Hence, the distance function fills in some null events $(null, null)$ to equalize numbers of events between the two records in each event type. The two

lists above become.

$$\begin{aligned} X &= \{(t_1, \text{“}A\text{”}), (t_2, \text{“}A\text{”}), (t_3, \text{“}B\text{”}), (null, null)\} \\ Y &= \{(u_1, \text{“}A\text{”}), (null, null), (u_2, \text{“}B\text{”}), (u_3, \text{“}B\text{”})\} \end{aligned}$$

The distance function between each pair of events is revised.

$$d'((t,c),(u,d)) = \begin{cases} \infty & \text{if } c \text{ and } d = null \\ 0 & \text{if } c = null,\ d \neq null \\ 0 & \text{if } c \neq null,\ d = null \\ d((t,c),(u,d)) & \text{if } c \text{ and } d \neq null \end{cases} \tag{4.3}$$

The null events should not be paired together so the distance is infinity. The pairs that have one null event indicate missing or extra events. The distance function does not include extra penalty for missing or extra events. Penalty for missing and extra events will be handled separately by the mismatch score (Section 4.2.3.2). Therefore, the distance is zero in these cases. Last, if the pair does not contain any null events, the original distance function is used.

Finally, a distance function between a target record X and a compared record Y becomes:

$$\begin{gathered} D'(X,Y) = min \sum_{i \in [1,m], j \in [1,n]} d'(x_i, y_j) \\ \text{each value of } i \text{ and } j \text{ is used exactly once.} \end{gathered} \tag{4.4}$$

Minimum Distance Perfect Matching The problem is how to match every event in X to an event in Y to yield minimum distance. This problem can be

converted into an assignment problem [67].

> "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 that all tasks are performed by assigning exactly one agent to each task in such a way that the total cost of the assignment is minimized."

Let events from X ($x_i = (t_i, c_i)$) become agents and events from Y ($y_j = (u_j, d_j)$) become tasks. Cost of the assignment is $d'(x_i, y_j)$. Then use the Hungarian Algorithm to solve the problem.

The time complexity of the Hungarian Algorithm is $O(n^3)$ when n is the number of events in each record. If there are m records in the database, the time to perform a matching between the target record and all records, assuming that each record has approximately n events is $O(mn^3)$.

4.2.3.2 Scoring

Once the matching is completed. The match, mismatch and total score can be derived from the matching.

Match Score The distance from M&M distance function captures the time difference between the two records. However, the distance can be a large number, which users find difficult to compare. Therefore, I normalize the distance into a *match score*, ranging from 0.01 to 1.00. A higher score represents higher similarity. Only records with zero distance will yield a score of 1.00. Otherwise, the highest

possible match score for non-zero distance is bounded to 0.99. The lowest score is bounded to 0.01 because zero score may mislead the users to think that the target and compared record are not similar at all. Let n be total number of records in the dataset. X and Y are the target and compared record, respectively. The match score ($M(X,Y)$) is calculated from the following equations:

$$D'_{max} = Max_{j\in[1,n]} D'(X,Y_j) \tag{4.5}$$

$$M(X,Y_i) = \begin{cases} 1.00 & \text{if } D'(X,Y_i) = 0 \\ \frac{D'_{max} - D'(X,Y_i)}{D'_{max}} * .98 + .01 & \text{otherwise} \end{cases} \tag{4.6}$$

Mismatch Score When the number of events in two records are not equal, there are missing or extra events. A *missing* event is an event that occurs in a target record but does not occur in a compared record. An *extra* event is an event that does not occur in a target record but occurs in a compared record. For example, imagine a target record for a patient who has chest pain, followed by elevated pulse rate, followed by a heart attack diagnosis. If the compared record has only chest pains and heart attack diagnosis, it has one missing event.

I count a *number of mismatches* ($N(X,Y)$), a sum of missing or extra events in each event type, and normalize it into a *mismatch score* ($MM(X,Y)$), ranging from 0.01 to 1.00. Only records with no mismatch events will yield a score of 1.00. Other records will score within 0.01 to 0.99 range.

$$N_{max} = Max_{j\in[1,n]} N(X,Y_j) \tag{4.7}$$

$$MM(X, Y_i) = \begin{cases} 1.00 & \text{if } N(X, Y_i) = 0 \\ \frac{N_{max} - N(X,Y_i)}{N_{max}} * .98 + .01 & \text{otherwise} \end{cases} \tag{4.8}$$

Total Score The *match score* and *mismatch score* are combined into *total score* ($T(X, Y_i)$) using weighted sum.

$$T(X, Y_i) = w * M(X, Y_i) + (1 - w) * MM(X, Y_i) \; ; \; w \in [0, 1] \tag{4.9}$$

Increasing the weight w gives match score more significance while decreasing w gives mismatch score more significance. The default value for weight is 0.5. (Both are equally significant.) For example, the users may not care whether there is any missing or extra event so the weight should be set to 1. Similan user interface allows users to manually adjust this weight and see the results in real-time. (See Section 4.2.2.4.)

4.2.3.3 Discussion

The concept that the similar records should have the same events (low number of mismatches) and the same events should occur almost at the same time (low time difference) is transformed into the M&M measure. Time difference and number of mismatches are two important aspects of similarity captured by the M&M measure. Records with high match score are records with low time difference while records with high mismatch score are records with low number of mismatches. The M&M measure can be adjusted to give significance to match or mismatch score. By de-

fault, the match score and mismatch score are assigned equal weights, so the most similar record should be the record with low time difference and also low number of mismatches.

4.3 User Study

A usability study for Similan was conducted with 8 participants. The goals in this study were to examine the learnability of Similan, assess the benefits of a scatterplot, learn how the number of events and event types affect user performance, and determine if users could understand the M&M measure in the context of its use. I also observed the strategies the users chose and what problems they encountered while using the tool. Synthetic data based on graduate school academic events, such as admission, successful dissertation proposal, and graduation, are used. This choice of data was intended to make the tasks more comprehensible and meaningful to participants, who were technically oriented graduate students.

4.3.1 Usability Study Procedure and Tasks

Two versions of Similan were used in this usability study: one with full features (S-Full) and another without a scatterplot (S-NoPlot). All usability sessions were conducted on an Apple laptop (15 inch widescreen, 2.2 Ghz CPU, 2GB RAM, Windows XP Professional) using an optical mouse.

The study had two parts. In the first part, participants had an introduction to the M&M measure and training with the Similan interface without a scatterplot

(S-NoPlot). Then, the participants were asked to perform this task with different parameters:

Given a target student and dataset of 50 students. Each student record has x event types of events and the total number of events is between y to z events. Find 5 students that are most similar to the target student using S-NoPlot.

Task 1 : $x = 2$, $y = 4$ and $z = 6$; Task 2 : $x = 4$, $y = 6$ and $z = 10$; Task 3 : $x = 6$, $y = 8$ and $z = 16$

In the second part, participants were introduced to the scatterplot and asked to perform task 4, 5 and 6 which are performing task 1, 2 and 3, respectively, but using S-Full instead of S-NoPlot.

The datasets used in task 1-3 and 4-6 were the same but the students were renamed and the initial orderings were different. Task 1 and 4 were used only for the purpose of training. The results were collected from tasks 2, 3, 5 and 6.

In addition to observing the participants' behaviors and comments during the sessions, I provided them with a short questionnaire, which asked specific questions about the Similan interface. Answers were recorded using a seven-option Likert scale and free response sections for criticisms or comments.

4.3.2 Results

For the first part of this 30-minute study, all participants were observed to use the following strategy: first select the target student, and then use the ranking mechanisms to rank students by the total score. In their first usage, some partici-

pants also selected the student who had the highest total score to see more detail in the comparison panel. Afterwards, they just studied the visualization and reported that these students with high total scores are the answer.

For the second part of the study, which focused on the scatterplot, most of the participants were observed to use the following strategy: first select the target student, draw a selection in the plot panel, and then use main panel's ranking mechanisms to rank students by the total score. However, a few participants did not use the scatterplot to do the task at all. They used the same strategy as in the first part.

Users spent comparable time on tasks 2 and 3 and on tasks 5 and 6. There was no difference in performance times between tasks 2 and 3 or between tasks 5 and 6, even though there were more events in tasks 3 and 6. This is understandable since participants reported that they trusted the ranking provided by the interface. However, users spent more time doing the tasks while using the scatterplot.

All of the participants trusted the *total score* ranking criterion and used it as the main source for their decisions. They explained that the visualization in the main panel convinced them that the ranking gave them the correct answers. Therefore, in the later tasks, after ranking by total score and having a glance at the visualization, they simply answered that the top five are the most similar.

All of them agreed that the main panel is useful for its ranking features and the comparison panel is useful in showing the similarity between the target and a compared student. However, they had different opinions about the scatterplot. Some of the participants mentioned that it was useful when they wanted to find

similar students. They explained that the similar students can easily be found at the bottom left of the scatterplot. One participant said that she had to choose two parameters (*#mismatch* and *match score*) when she used the scatterplot. On the other hand, while using the main panel, she had to choose only one parameter (*total score*), which she preferred more. A few of them even mentioned that it is not necessary to use the scatterplot to find similar students. Although they had different opinions about its usefulness in finding similar students, they all agreed that the scatterplot gives a good overview of the students' distribution. It can show clusters of students, which could not be discovered from other panels. Also, one participant pointed out that the main and comparison panels are helpful in showing how students are similar, while the plot is more helpful in explaining how students are dissimilar.

Participants had positive comments on Similan's simple, intuitive and easy to learn interface. Most of the participants got started without assistance from the experimenter. Nevertheless, some user interface design concerns were noted. Some participants noticed that the binned timeline could be misleading in some situations.

Overall, participants liked the simple yet attractive Similan's interface and strongly believed that Similan can help them find students who are similar to the target student. Ranking in the main panel appears to be useful. By contrast, participants had difficulties in learning the M&M measure, since it combines two kinds of scores. The scatterplot did not benefit the tasks in this study but it may prove useful for more complex databases.

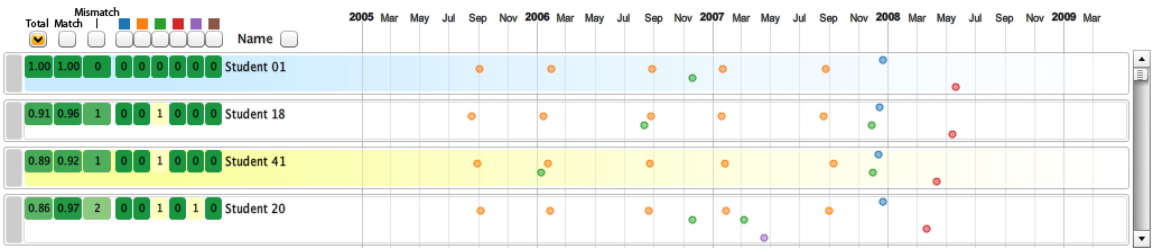

Figure 4.5: Similan 1.5 prototype: The timeline is continuous and events are split into rows by event type.

4.3.3 Pilot Study of a New Prototype

According to the user feedback, using a binned timeline can be misleading in some situations. A pair of events in the same bin can have a longer distance than a pair of events in different bins. Also, order of events within the same bin is hidden. Therefore, I develop a new prototype that adopts the continuous timeline used in Lifelines2 [131] and includes several improvements (Figure 4.5.)

I did a pilot study with 5 participants to compare the binned timeline in the original version and continuous timeline in the new version and received these comments: The continuous timeline requires more space for each record and looks more complicated. The binned timeline is more compact, simpler and therefore more readable. It gives users less detail to interpret. However, the continuous timeline does not mislead users when comparing distances or ordering. Both types of timeline have advantages and disadvantages depending on the task. The binned timeline is suitable for general tasks that do not require fine-grain information while the continuous timeline is more suitable when fine-grain information is required.

4.4 Similan and the M&M Measure: The Second Version

The first Similan's usefulness is limited for several reasons: it only allows users to select an existing record from the database as a query (not to specify an example of their choice), the binned timeline visualization can be misleading and frustrating to users in some situations (Figure 4.2), and the similarity measure is not flexible enough to support different definitions of similarity for different tasks.

Therefore, to address these limitations, a new version of the similarity measure and the user interface were developed. *Similan2* becomes a query interface which allows the users to draw an example of an 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. 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.

4.4.1 Description of the User Interface: Similan2

4.4.1.1 Overview

Similan2 (Figure 4.6), is an Adobe Air Application using the Adobe Flex 3 Framework. The designs of LifeLines2 and Similan2 have evolved in parallel.

Similan2 adopted the basic display of records from LifeLines2 to solve the binned timeline issue: each record is stacked on the main panel, events are colored

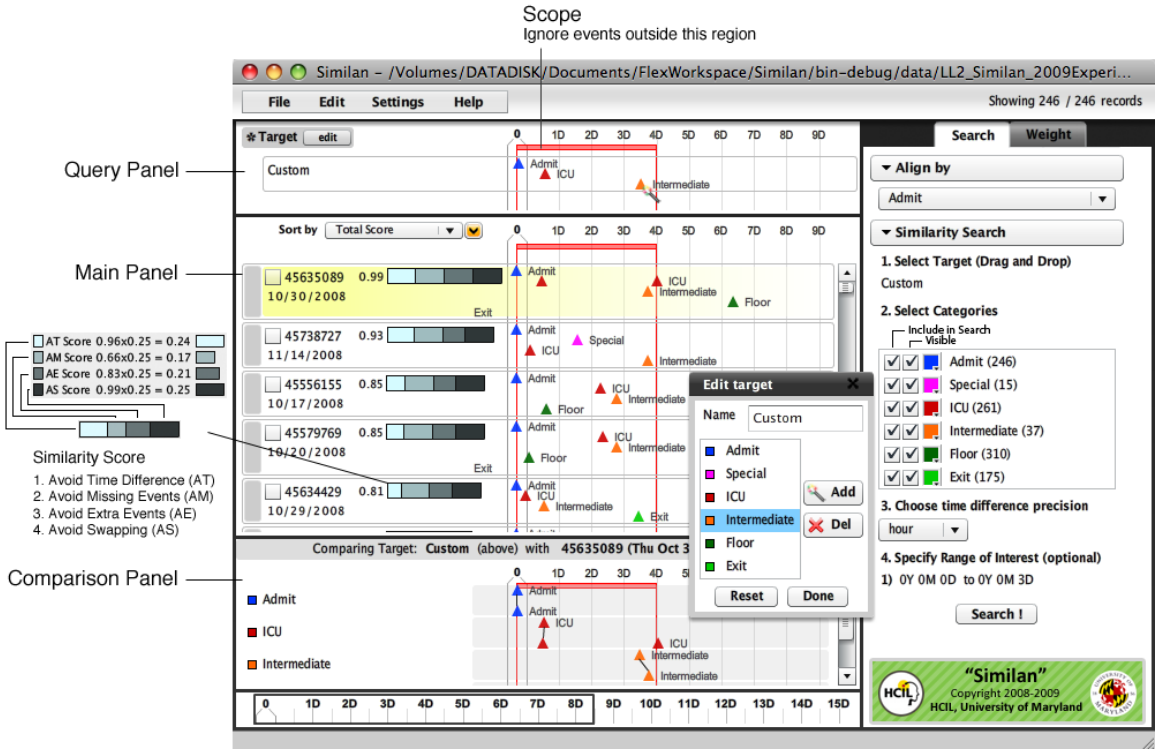

Figure 4.6: Similarity search interface (Similan2) with the same query as in Figure 4.11. Users specify the query by placing events on the query panel. To set the time range of interest and focus on events within this range, users draw a red box. After clicking on "Search", all records are sorted by 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 have longer leftmost sections than the third record because they have lower time difference so the Avoid Time Difference Score (AT) is high, resulting in longer bars. Figure 4.7 shows how users can adjust the weight.

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 4.6). Similan2 also employs an improved similarity measure (M&M measure v.2), which will be explained in Section 4.4.2.

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.

4.4.1.2 Query

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 onto the empty timeline (i.e. users can select `Admit` from the list of event types 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 4.6 shows the patterns they created `Admit`, `ICU` and `Intermediate` at time 0, on the first day and fourth day, respectively. The only type of time constraint that is currently supported by Similan2 is specifying when each event occurred.

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 4.6). In the 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 be selected by default. The unit for time differences (e.g. hours or days) can be selected from a drop-down list. Event types 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 (placing records with the highest scores on the top). Each record has a *score indicator*, a rectangle with four sections of different color (See Figure 4.6.), inspired by *ValueCharts* [24], 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.

4.4.1.3 Comparison

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. Hover over a link brings up a tooltip showing the event type, time of both events and time difference.

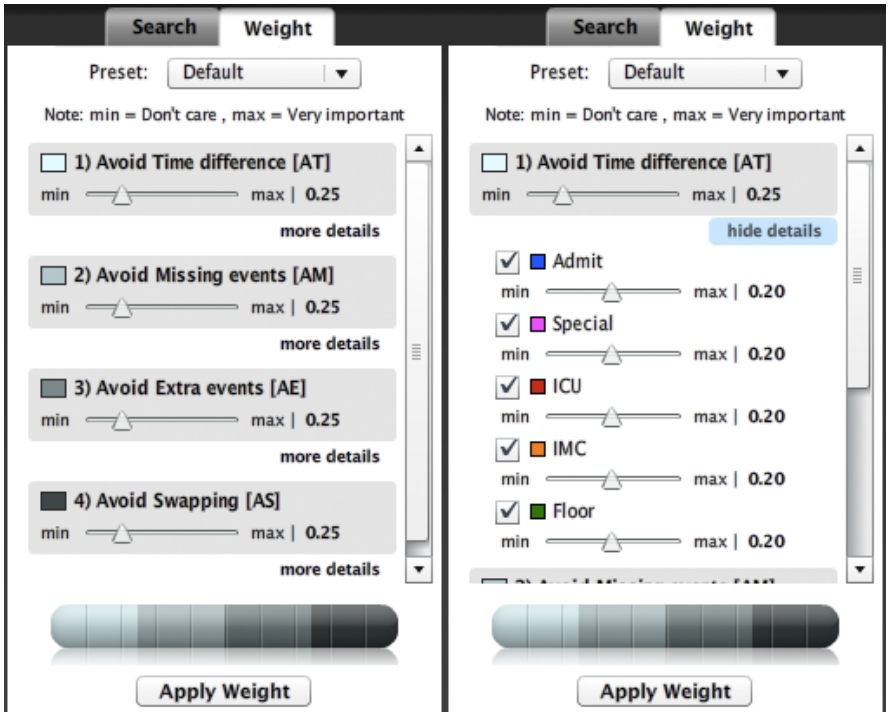

Figure 4.7: Similan2's control panel has 2 tabs. The first tab is "search" as shown in Figure 4.6. Another tab is "weight and detailed weight", for which users can adjust the weight of the four decision criteria using the four sliders in the left figure. For more advanced customization, they can even set the weight for each event type within each decision criterion by clicking on "more details" (right figure).

4.4.1.4 Weights

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 4.7.) 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".

4.4.2 The Match and Mismatch (M&M) Measure v.2

The M&M measure v.2 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,

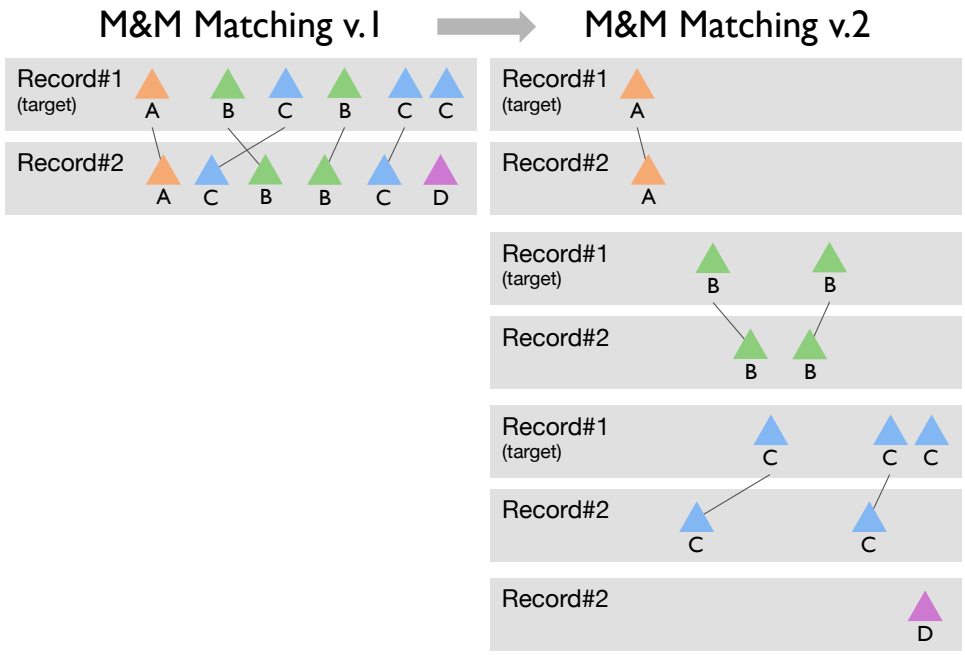

Figure 4.8: (left) M&M Matching v.1 (right) M&M Matching v.2: Events in each event type are matched separately.

it increases the flexibility by adding more customizable parameters. The M&M measure v.2 still consists of two steps: *matching* and *scoring*.

4.4.2.1 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

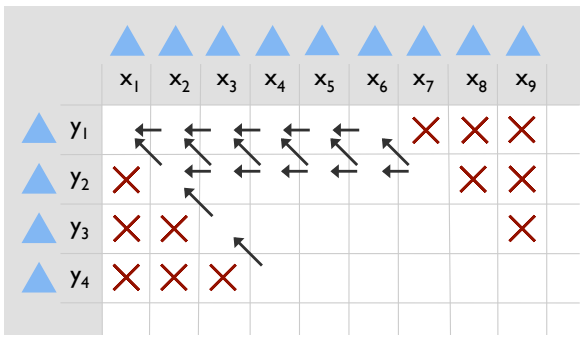

Figure 4.9: M&M Matching v.2: Dynamic programming table

with event `A` and cannot match with event `B` or `C`. (See Figure 4.8.) Therefore, the matching can be reduced into a simpler problem by separating the matching for each event type. 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).

$$X = \{(t, c) \mid t \in Time \text{ and } c \in Categories\} \tag{4.10}$$

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

$$\begin{aligned} X &= \{(t_1, \text{“}A\text{”}), (t_2, \text{“}A\text{”}), (t_3, \text{“}B\text{”})\} \\ Y &= \{(u_1, \text{“}A\text{”}), (u_2, \text{“}B\text{”}), (u_3, \text{“}B\text{”})\} \end{aligned}$$

“A”,“B” $\in$ Categories

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

$$\begin{aligned} X_A &= \{(t_1, \text{“}A\text{”}), (t_2, \text{“}A\text{”})\} \quad , \quad X_B = \{(t_3, \text{“}B\text{”})\} \\ Y_A &= \{(u_1, \text{“}A\text{”})\} \quad , \quad Y_B = \{(u_2, \text{“}B\text{”}), (u_3, \text{“}B\text{”})\} \end{aligned} \tag{4.11}$$

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 4.8.) 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 match between X_A and Y_A that produces the minimum time difference.

Dynamic Programming Matching Figure 4.9 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 have 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 2) 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:

$$cell(i, j) = min \begin{cases} d(x_i, y_j) + cell(i-1, j-1) \\ cell(i-1, j) \end{cases} \tag{4.12}$$

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,

Algorithm 2 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
```

and $n_A > m_A$, the time to match the events between X_A and Y_A with dynamic programming is

$$O((n_A - m_A) * m_A) \tag{4.13}$$

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

$$O((max(n_A, m_A) + max(n_B, m_B) + max(n_C, m_C) + ...)^3) \tag{4.14}$$

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

$$O((n_A - m_A) * m_A + (n_B - m_B) * m_B + (n_C - m_C) * m_C + ...) \tag{4.15}$$

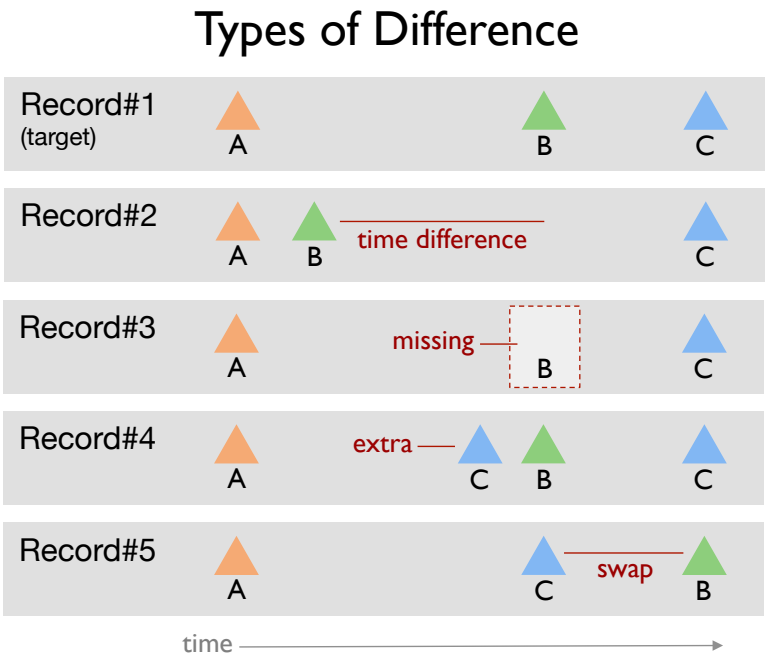

Figure 4.10: Four types of difference: time difference, missing events, extra events and swaps

4.4.2.2 Scoring

Once the matching is completed. The scores can be derived from the matching. The first version of the M&M measure considers only two types of difference: time difference and number of mismatches (missing or extra events). In this second version, I 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. I also included the number of swaps because sometimes the users want the events in order but sometimes 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.
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 *swap* occurs when the order of the events is reversed. The *number of swaps (NS)* is counted for each pair of event categories. For example, in Figure 4.10, 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 not be 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 ($\sum w^{TD} = 1$).

$$\sum TD = w_A^{TD} * TD_A + w_B^{TD} * TD_B + ... \tag{4.16}$$

Likewise, the *total number of missing events* ($\sum NM$), *total number of extra events* ($\sum NE$) and *total number of swaps* ($\sum NS$) are calculated from weighted sums.

$$\sum NE = w_A^{NE} * NE_A + w_B^{NE} * NE_B + ... \quad (4.17)$$

$$\sum NM = w_A^{NM} * NM_A + w_B^{NM} * NM_B + ... \quad (4.18)$$

$$\sum NS = w_{A,C}^{NS} * NS_{A,C} + w_{B,C}^{NS} * NS_{B,C} + ... \quad (4.19)$$

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 swaps ($\sum NS$) 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 Swaps (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 ($\sum w = 0.99$).

$$T = w_{AT} * AT + w_{AM} * AM + w_{AE} * AE + w_{AS} * AS \tag{4.20}$$

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

4.5 User Study

4.5.1 Motivation for a Controlled Experiment

Using the Multi-dimensional In-depth Long-term Case Study methodology [114], I worked with a physician by assisting him through the analysis of his data using two query tools: LifeLines2 [131, 132] (an exact match user interface from my 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 looking at the results ranked by similarity was difficult and sometimes frustrating as

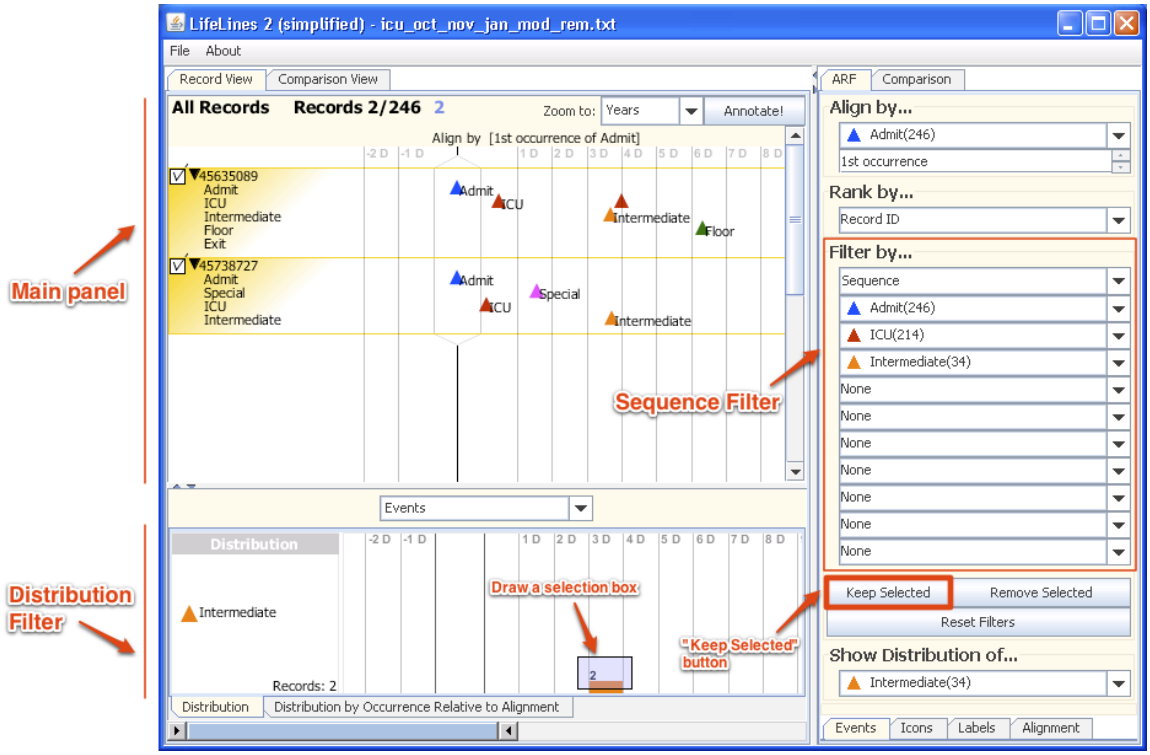

Figure 4.11: Exact match interface (LifeLines2) showing the results of a query for patients who were admitted to the hospital then transferred to the Intensive Care Unit (ICU) within a day, then to an Intermediate ICU room on the fourth day. The user has specified the sequence filter on the right selecting `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.

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 me to design the controlled experiment to see if I could confirm those impressions and better understand which query method is better suited for different tasks.

4.5.2 Description of the User Interface: LifeLines2

LifeLines2 was developed during a previous HCIL project, which is described here only for the purpose of the controlled experiment. LifeLines2 (Figure 4.11) is a Java application, utilizing the Piccolo 2D graphics framework [14]. 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 4.11 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 event types 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 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 first 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 first to the fourth day and "Keep selected". (See Figure 4.11.) A similar process can be used for consecutive interval specification using different alignments and filtering.

4.5.3 Method

I conducted a controlled experiment comparing 2 interfaces: LifeLines2, an exact match interface, and Similan2, a similarity search interface. My 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 I wanted to compare.

4.5.3.1 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 time and accuracy between the performance of different tasks while using each interface?

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

4.5.3.2 Participants

Eighteen graduate and senior undergraduate students participated in the study. I recruited computer science students who are assumed to have a high level of comfort with computers, but have no knowledge of either interface. The participants included 13 men and 5 women, 20 to 30 years of age. Participants 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.

4.5.3.3 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.

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). I 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 a 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."

Data – I used a modified version of the deidentified patient transfer data provided by my 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).

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?

4.5.3.4 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. I used equivalent datasets for each interface.

To control learning effects, the presentation order of the LifeLines2 and Similan2 interfaces was counterbalanced. To avoid situations in which the users would

always find repeating the tasks on the second system easier, since they already know the answer, I also used two sets of questions and datasets, one for each interface. The questions in the two sets are different but have the 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.

4.5.3.5 Procedure

Participants were given training which included a brief description of the data and ten-minute tutorials on how to use 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, I debriefed the participants to learn about their experience while using the interfaces for the different tasks and their suggestions for improving the interfaces.

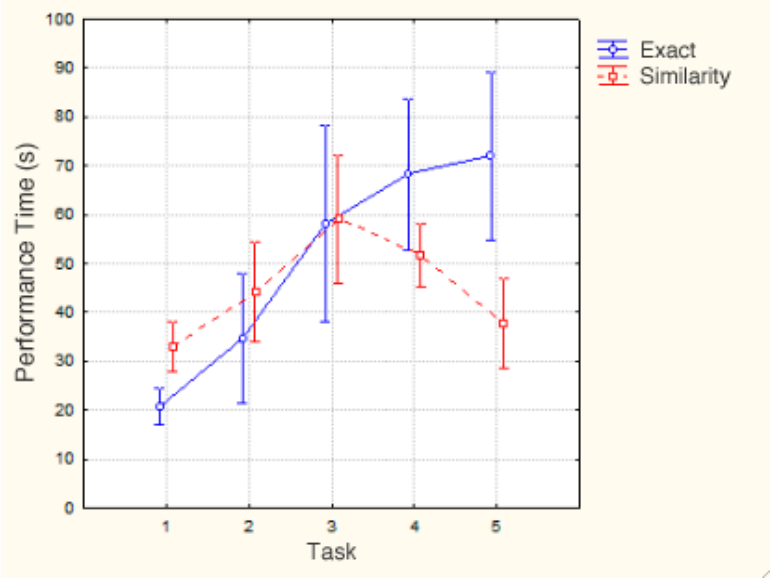

Figure 4.12: Performance time a function of the interface type and the tasks (1-5). Vertical bars denote 0.95 confidence intervals.

4.5.4 Results

4.5.4.1 Performance Time

To examine the effects of the type of interface and the task on time to perform tasks 1-5, I 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 4.12 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±SD of 26.83±10.90 s, 39.58±23.92 s and 58.67±33.60 s, respectively). However, the average performance times for tasks 4 and 5 are shorter using the similarity search interface (M±SD of 51.73±13.21 s and 37.74±18.63 s, respectively) than while using the exact match interface (M±SD of 68.33±31.18 s and 72.05±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 difference in performance time was between task 3 to tasks 1 and 5 ($p<.05$).

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.

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

4.5.4.2 Error Rates

To compare the error rates between the two interfaces while performing the different tasks, I 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, I 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 rates of task 4 ($\chi^2(1)$=3.06,p=.08) and 5 ($\chi^2(1)$=1.13,p=.29).

4.5.4.3 Subjective Ratings

To compare the difference between the subjective ratings given by the participants to the two interfaces, I conducted a paired-sample t-test for each question. The results of the analysis are presented in Table 4.1. 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).

4.5.4.4 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 what 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.")

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 distinguish 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."

4.6 Lessons Learned and Ideas for Hybrid Interface

The experiment showed that exact match interface 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 interface 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, the similarity search interface 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 makes the search process more transparent because they could see the big picture of their query. However, similarity search interface was more difficult for tasks that involve counting. The participants requested a better way to support counting tasks.

The exact match and similarity search interfaces each have their advantages. How can I combine the best features from these two interfaces to create a new interface that can support tasks with uncertainty and time constraints as well as simpler and counting tasks? Based on the results of the experiment and my observations during the longitudinal study with my partners, I 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, I found many users used more filters than necessary.

4.7 Summary

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 are exact match interfaces, 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 interface allows users to sketch an example of what they are seeking and find similar results, which provides more flexibility.

In the beginning, I introduce the M&M measure, a similarity measure for event sequences. Briefly, the M&M measure is a combination of time differences between events, and number of missing and extra events. The M&M measure was developed alongside with Similan, an interactive tool that allows users to search for event sequence records that are similar to a specified target record and provides search results visualization. A user study showed promising direction but also identified rooms for improvement.

To address the limitations from the first version, I developed Similan2, an in-

terface that allows users to draw an event sequence example and search for records that are similar to the example, with improved search results visualization. The M&M measure was also refined. The M&M measure v.2 is faster and can be customized by four decision criteria, increasing its flexibility.

Finally, I 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. These hybrid ideas were explored and led to the *Flexible Temporal Search (FTS)* in Chapter 5.

Chapter 5

Combining Exact Match and Similarity Search for Querying Event Sequences: Flexible Temporal Search

5.1 Introduction

My research aims to support users in searching for event sequences when they are uncertain about what they are looking for. For example, the physician wants to find the records that have event `Surgery` followed by event `Die` within approximately 2 days. The value 2 days is just an approximation.

Many methods are exact match, which does not leave much room for flexibility. The exact match creates a clear *cut-off* point for records that pass the query and excludes records that do not pass from the result set. A too narrow query (e.g. Surgery→2 days→Die) can inadvertently exclude records that might be slightly beyond the cut-off point (2.5 days instead of 2 days). In a situation when the users are not 100% confident about the query, the exact match prevents them from seeing beyond the cut-off point what they might have missed.

Trying to overcome the limitations of the exact match, many researchers have developed similarity search for event sequences. These methods calculate the similarity score between each record and the query and return all records sorted by their similarity scores. The result set now can tell the users which records are more or less

similar to the query. However, I have learned from the user study in the previous chapter that using similarity search also has its limitations. Without a clear cut-off point, users could not easily or confidently count how many records are acceptable results. It is more suitable for finding the most similar records to the query.

These lessons guided me to set design goals for a new hybrid user interface.

1. Users must be able to see a clear a cut-off point between records that pass the query and records that do not.

2. Users must be able to see records that are beyond the cut-off point, sorted by similarity to the query.

Following these goals, I have designed a new hybrid interface called *Flexible Temporal Search (FTS)* which combines the precision of the exact match and flexibility of the similarity search. In this chapter, I will explain the FTS in more detail, starting from an explanation of the design of the query in Section 5.2, an algorithm for similarity score computation in Section 5.3, and a user interface in Section 5.4.

5.2 Query

5.2.1 How to define a cut-off point? Mandatory & Optional Flags

The FTS result set is split into two bins: *exact match* results and *other* results. Records that are within the cut-off point are put in the *exact match results* while records that are beyond the cut-off point are put in the *other results*.

The big question here is how could the interface know where the cut-off points

should be? The answer lies in the queries. Even in a query with uncertainty, users may have some parts of the query that they are certain about, e.g. patients must be in the `ICU`. If users can specify which part of the query they are certain about, the cut-off points can be derived from the certain part of the query. Using this concept, I introduce the *mandatory* and *optional* flags that let the users define which part of the query is flexible or not.

1. **Mandatory**: When users are certain about some constraints, they specify them as mandatory, so any record that violates any of these constraints will be excluded from the exact match results and put in other results.

2. **Optional**: When users are flexible about some constraints, they can specify them as optional. Records that pass these constraints will receive bonus similarity scores, but records that violate these constraints are not excluded from the exact match results. Records with higher similarity scores are displayed higher in the result set. Therefore, optional constraints allow user to specify what they prefer to see on the top of the list.

Comparing to the existing interfaces, in exact match (such as *LifeLines2*), all events are mandatory while in similarity search (such as *Similan*), all events are optional. Figure 5.1 guides how to decide whether a constraint is mandatory or optional. In summary, users should use mandatory as much as they can, and use optional to specify what they prefer to see on the top of the list.

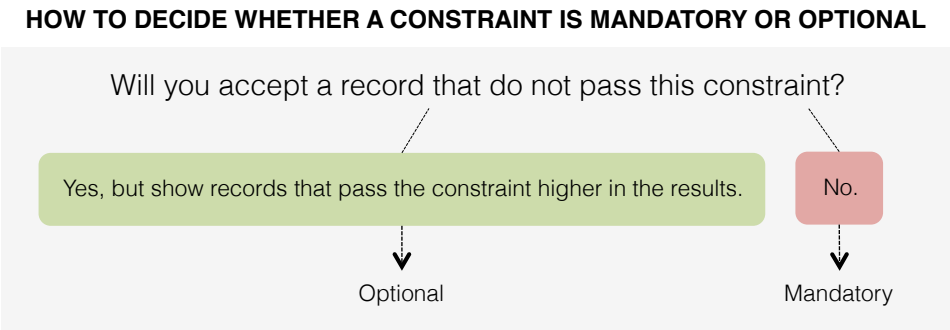

Figure 5.1: How to decide whether a constraint is mandatory or optional

5.2.2 Specification

Users can specify queries in terms of events, negations and gaps between consecutive events. For simplicity and performance, FTS does not support a query that has events which occurred at the same time.

1. **Event**: Users add an event that they want and indicate if it is mandatory or optional.

   (a) *Mandatory* (X): Event `X` **must** occur.

   - $A \rightarrow B \rightarrow C$

     Find records with pattern A followed by B and C. For example, find all patients who arrived (A), were transferred to ED (B) and discharged (C).

   (b) *Optional* (x): Event `X` **should** occur, but I can tolerate if it does not.

   - $A \rightarrow b \rightarrow C$

     Find records with pattern A followed by b and C. b is preferred but not required. For example, find all patients who were transferred

from ICU (A) to Floor (C). Being transferred to the IMC (b) in between is preferred, but it is acceptable if the patients were transferred to Floor without passing the IMC first.

For each event, users can also specify when they want it to occur. If the time constraint is not specified, the query will only look for existence of the event and ignore its time. The time constraint can be mandatory or optional as well.

(a) *Mandatory* (X_t): Event `X` **must** occur within time t.

- $A_{May2011} \rightarrow B$

  Find records with pattern A followed by B and C. A must occur in May 2011. For example, find all patients who had surgeries (A) in May 2011 and died (B).

(b) *Optional* ($X_{\approx t}$): Event `X` **should** occur **near** time t.

- $A_{\approx Dec15,1642} \rightarrow B_{\approx Mar20,1727}$

  Find records with pattern A followed by B. A should occur near Dec 15, 1642 while B should occur near Mar 20, 1727. For example, a history student wants to find famous scientists who was born (A) and died (B) closest to Sir Isaac Newton (Dec 25, 1642 – Mar 20, 1727). In this case, the user is certain that the two events (born and die) must occur, so the events are mandatory. However, the user is not certain about the timing. He only wants them to be as close to Newton's as possible and does not have any time limit, so he set the timing as optional.

2. **Negation or Not-occur**: Users add negation for an event that they do not want and indicate if it is mandatory or optional.

   (a) *Mandatory* ($\neg X$): Event `X` **must** not occur.

   - $A \rightarrow \neg B \rightarrow C$

     Find records with pattern A followed by C without B in between. For example, find all patients who arrived (A) and were discharged (C) without being in the ICU (B).

   (b) *Optional* ($\neg x$): Event `X` **should** not occur, but I can tolerate if it does.

   - $A \rightarrow \neg b \rightarrow C$

     Find records with pattern A followed by C. It is more preferable not to have b between A and C, but not prohibited. For example, find all patients who were in the floor (A) and were discharged (C). It is more preferable to avoid any additional transfer to another floor (b), i.e. I want to see patients who exited without being transferred to another floor on the top, and if there were any patients who were transferred to another floor, they can be included but show them in the bottom of the list.

3. **Gap**: Users can specify gap between consecutive events. Each gap consists of minimum duration, maximum duration, or both. The gap can also be mandatory or optional.

   (a) *Mandatory*: The gap **must** be within specified range.

- $A \rightarrow (0, 10mins] \rightarrow B$

  Find records with pattern A followed by B within ten minutes. For example, a coach wants to know how many times in this season that his team conceded (B) a goal within ten minutes after scoring (A).

(b) *Optional*: The gap **should** be **close** to specified range.

- $A \rightarrow\approx [10mins] \rightarrow B \rightarrow\approx [10mins] \rightarrow C$.

  Find records with pattern A followed by B in approximately ten minutes and followed by C in approximately ten minutes. For example, a commentator is narrating a soccer match that the home team have scored (A), the opponent have retaliated by scoring (B) in ten minutes, and the home team have just scored again (C) after ten minutes. He then wants to find another match in the past that is most similar to his current match. The time gaps in this example are not mandatory because the user does not have a clear range of what he expects in the results and he will accept all records no matter how much the gaps are. He just wants to use these gaps to find the most similar matches. This is different from the previous example that the coach knows exactly what he is looking for.

5.2.3 Grade & Similarity Score

When the search is executed, FTS compares each record to the query and calculates its *grade* and *similarity score*. The grades are either *pass* or *fail*, allowing the

interface to group records into *exact match* and *other results*, respectively. Records that violate any of the mandatory constraints are failed. The rest are passed.

The similarity score is computed from 4 types of difference.

1. *Missing Events*: Events that are in the query but do not exist in the record.
2. *Extra Events*: Events that are not specified in the query but appear in the record. It can be broken down into three sub-criteria if the users choose to:
   (a) Extra events before the first match
   (b) Extra events between the first match and the last match—in other words, interrupting the query pattern
   (c) Extra events after the last match
3. *Negation Violations*: Events that are listed as negations but appear in the record.
4. *Time Difference*: Deviation from the time constraints

Users can set the weight for each type of difference. The total similarity score is computed from a weighted sum of these criteria (See Section 5.3.4).

5.3 FTS Similarity Measure

5.3.1 Preprocessing

During preprocessing, a query is cut into blocks ($[X]$) that can be matched with the events in each records.

1. Events and negations are automatically cut into blocks, e.g., $A \rightarrow \neg B \rightarrow C$ is converted into $[A] \rightarrow [\neg B] \rightarrow [C]$.

2. Each gap is merged with the following event, e.g., $A \rightarrow B \rightarrow (0, 2days) \rightarrow C$ is converted into $[A] \rightarrow [B] \rightarrow [(0, 2days) \rightarrow C]$.

3. Consecutive negations are merged into special blocks when their order is not taken into account, e.g., $A \rightarrow \neg B \rightarrow \neg C$ is converted into $[A] \rightarrow [\neg B|\neg C]_{\#1} \rightarrow [\neg B|\neg C]_{\#2}$. When B is already matched in the first block, the algorithm will remember and not match B in the second block.

After the preprocessing is completed, the computation consists of two steps: matching and scoring.

5.3.2 Matching

I use a typical dynamic programming approach [4] to find the optimal matching between the blocks in the query $Q = b_1, b_2, b_3, ..., b_m$ and events in each record $R = e_1, e_2, ..., e_n$. Given the query Q and record R, I use $s(i, j)$ to denote the maximum *similarity vector* of matching the first i blocks of the query Q and the first j events of the record R. With this definition, the maximum similarity vector of matching query Q and record R is $s(m, n)$.

The base conditions and recurrence relation for the value $s(i, j)$ are

$$
\begin{aligned}
s(0,0) &= \text{empty vector} \\
s(i,0) &= s(i-1,0) + skip(b_i) \\
s(0,j) &= s(0,j-1) + skip(e_j) \\
s(i,j) &= max \begin{cases} s(i-1,j) + skip(b_i) \\ s(i,j-1) + skip(e_j) \\ s(i-1,j-1) + match(b_i, e_j) \end{cases}
\end{aligned}
\tag{5.1}
$$

where $skip(b_i)$ and $skip(e_j)$ are costs of skipping block b_i and event e_i, and $match(b_i, e_j)$ is a cost of matching block b_i and event e_j. A match is possible only when the block and event have the same event type.

To compare two similarity vectors $s(i, j)$ for matching, the values within both vectors are compared in the following order. If the values are equal, the next values are used. Otherwise, the rest of the values are ignored.

1. *number of matched events (mandatory)*: The higher number, the more similar the vectors are.

2. *number of matched events (optional)*: The higher number, the more similar the vectors are.

3. *number of negations violated (optional)*: In reality, having lower number should be more similar. However, if it is like that, the algorithm will always skip the negations, so I reverse the comparison just for matching to trick the algorithm.

4. *number of negations violated (mandatory)*: The lower number, the more similar the vectors are, but the comparison is reversed for the same reason as above.

5. *number of time constraints (gap & time of occurrence) violated (mandatory)*: The lower number, the more similar the vectors are.

6. *time difference*: The less different, the more similar the vectors are.

7. *number of extra events*: The lower the number, the more similar the vectors are. If the users select to treat extra events separately, these three values will be used. They will be sorted by the weights that the users set for these sub-criteria.

   (a) number of extra events before the first match

   (b) number of extra events between the first and last match

   (c) number of extra events after the last match

Skipping a block ($skip(b_i)$) in the query does not add any change to the vector while skipping events in the record ($skip(e_j)$) increases the number of extra events. Matching ($match(b_i, e_j)$) may increase the number of matched events (if the block is an event) or negations violated (if the block is a negation), incur time difference or violate time constraints (if the block has any time constraint). For a block with gap, if the previous block is not matched, the gap is ignored. Algorithm 3 describes the FTS matching in more detail.

Algorithm 3 FTS Matching

```
1: s(0,0) = new SimilarityVector();
2: for i = 1 to m do
3:    s(i,0) = s(i-1,0).clone();
4: end for
5: for j = 1 to n do
6:    s(0,j) = s(0,j-1).clone().incrementExtraEvent();
7: end for
8: for i = 1 to m do
9:    for j = 1 to n do
10:      {// Compare between skipping b_i and e_j.}
11:      s(i,j) = max( s(i-1,j).clone() , s(i,j-1).clone().incrementExtraEvent() );
12:      if b_i.eventType == e_j.eventType then
13:         s_m = s(i-1,j-1).clone(); {// similarity vector for matching b_i and e_j.}
14:         if b_i.isNegation() then
15:            if b_i.isMandatory() then
16:               s_m.incrementViolatedNegation();
17:            else
18:               s_m.incrementViolatedOptionalNegation();
19:            end if
20:         else
21:            if b_i.isMandatory() then
22:               s_m.incrementMatchedEvent();
23:            else
24:               s_m.incrementMatchedOptionalEvent();
25:            end if
26:            if b_i.hasTime() then
27:               diff = b_i.getTime().diff(e_j);
28:               s_m.addTimeDifference( diff );
29:               if diff > 0 and b_i.getTime().isMandatory() then
30:                  s_m.isTimeViolated = true;
31:               end if
32:            end if
33:            if i > 1 and b_i.hasTimeGap() then
34:               if s(i-1,j-1).hasMatched(b_{i-1}) then
35:                  gap = e_j.time− previousMatchedEvent.time;
36:                  diff = b_i.getTimeGap().diff(gap);
37:                  s_m.addTimeDifference( diff );
38:                  if diff > 0 and b_i.getTimeGap().isMandatory() then
39:                     s_m.isTimeViolated = true;
40:                  end if
41:               end if
42:            end if
43:         end if
44:         s(i,j) = max( s(i,j) , s_m );
45:      end if
46:   end for
47: end for
```

Algorithm 4 FTS Grading

```
1: if s.mandatoryEventMatchedCount() < totalMandatoryEventsCount then
2:     return false;
3: else if s.mandatoryNegationViolatedCount() > 0 then
4:     return false;
5: else if s.isTimeViolated == true then
6:     return false;
7: else
8:     return true;
9: end if
```

5.3.3 Grading

After the matching is completed, the similarity vector of each record is converted into a boolean flag *passExactMatch* indicating whether a record is included in the exact match or other results. If any of the mandatory events is not matched, any of the mandatory negations is matched or any mandatory time is violated, the flag is set to false. Otherwise, the flag is set to true. (Algorithm 4)

5.3.4 Scoring

The similarity vector is also converted into a numerical similarity score. Users can set weights $(0-100)$ for optional events, optional negations, time difference and extra events (and the three sub-criteria of extra events if necessary) to customize the conversion. The weights for mandatory events and mandatory negations are fixed to 100.

Before calculating the score, the weights are normalized ($w' = \frac{w}{\sum w}$) to make the total sum equal to 100 ($\sum w' = 100$). Then the total similarity score can be computed from the equation below (# = number of). To avoid division by zero, I

used a custom function called *sDivide* (Algorithm 5) instead of standard division. If a denominator is zero, this function returns a specified default value (0 or 1). This means that if any type of contraint is not set in the query, each record receives a maximum score ($w' * 1$) for that type of constraint.

$$
\begin{aligned}
score \;=\; & w'_{\text{mandatory events}} && * \; sDivide(\frac{\#\text{ matched mandatory events}}{\#\text{ mandatory events in query}}, 1) \\
+\; & w'_{\text{optional events}} && * \; sDivide(\frac{\#\text{ matched optional events}}{\#\text{ optional events in query}}, 1) \\
+\; & w'_{\text{mandatory negations}} && * \; (1 - sDivide(\frac{\#\text{ mandatory negations violated}}{\#\text{ mandatory negations in query}}, 0)) \\
+\; & w'_{\text{optional negations}} && * \; (1 - sDivide(\frac{\#\text{ optional negations violated}}{\#\text{ optional negations in query}}, 0)) \\
+\; & w'_{\text{time difference}} && * \; (1 - sDivide(\frac{\text{time difference}}{\text{maximum time difference}}, 0)) \\
+\; & w'_{\text{extra events}} && * \; (1 - sDivide(\frac{\#\text{ extra events}}{\text{maximum }\#\text{ extra events}}, 0))
\end{aligned}
\tag{5.2}
$$

Algorithm 5 function sDivide($\frac{\text{dividend}}{\text{divisor}}$, defaultValue)

```
1: if divisor == 0 then
2:     return defaultValue;
3: else
4:     return dividend/divisor;
5: end if
```

If the three sub-criteria of extra events are used, the last line of the equation above is replaced with three lines of the sub-criteria instead. The final similarity score is within the range 0 to 100.

5.3.5 Performance

The time complexity of a matching between query and one record is $O(mn)$ when m is the number of blocks in a query and n is the number of events in compared

record. Therefore, the time complexity for a search in dataset is $O(mN)$, when N is the number of events in the dataset.

5.3.6 Difference from the M&M Measure

The FTS measure has evolved from the M&M Measure (Section 4.4.2). In summary, there are several difference. (1) The FTS also provides a grade (pass/fail) while the M&M does not. (2) The FTS can support mandatory and optional constraints. (3) The FTS handles negations. (4) The FTS does not handle swap. This change is due to the matching. In FTS, all events from all event types are matched at the same time while in the M&M, events are matched seperately for each event type. A matching from dynamic programming must be monotonic and, therefore, cannot include swaps. (5) FTS score ranges from 0.00 to 100.0 while M&M score ranges from 0.00 to 0.99.

5.4 User Interface

I have developed a hybrid user interface for querying event sequences with FTS and integrated these features into the LifeFlow software (Figure 5.2). The user interface consists of three main parts: control, query and result panels. The control panel, inherited from LifeLines2 [131] allows users to align the records by sentinel events, if necessary. Users can specify query on the query panel and the search results will appear in the result panel. The following sections (5.4.1, 5.4.2 and 5.4.3) describe visual representation of FTS query, query specification and search results

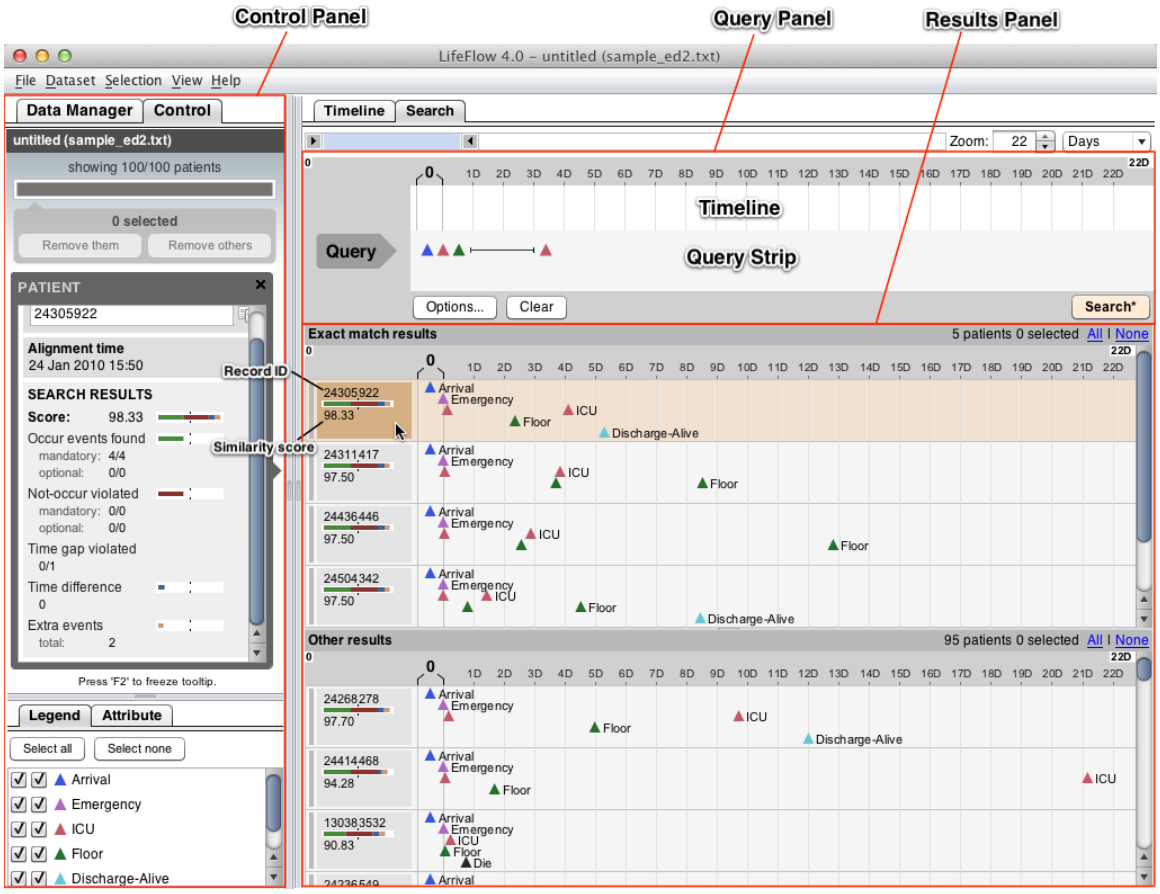

Figure 5.2: Flexible Temporal Search (FTS) implemented in LifeFlow software: User can draw a query and retrieve results that are split into two bins: exact match and other results. In this example, a user is querying for bounce back patients, which are patients who `arrived` (*blue*), were later moved to the `ICU` (*red*), transferred to `Floor` (*green*) and transferred back to the `ICU` (*red*) within two days. The results panel display all bounce back patients in the exact match results while showing the rest in the other results, sorted by their similarity to the query. The top patient in the other results has a pattern very similar to bounce back but the return time to `ICU` was 4 days. The physician then can notice this patient and use his/her judgment to decide whether to consider this case as a bounce back patient or not.

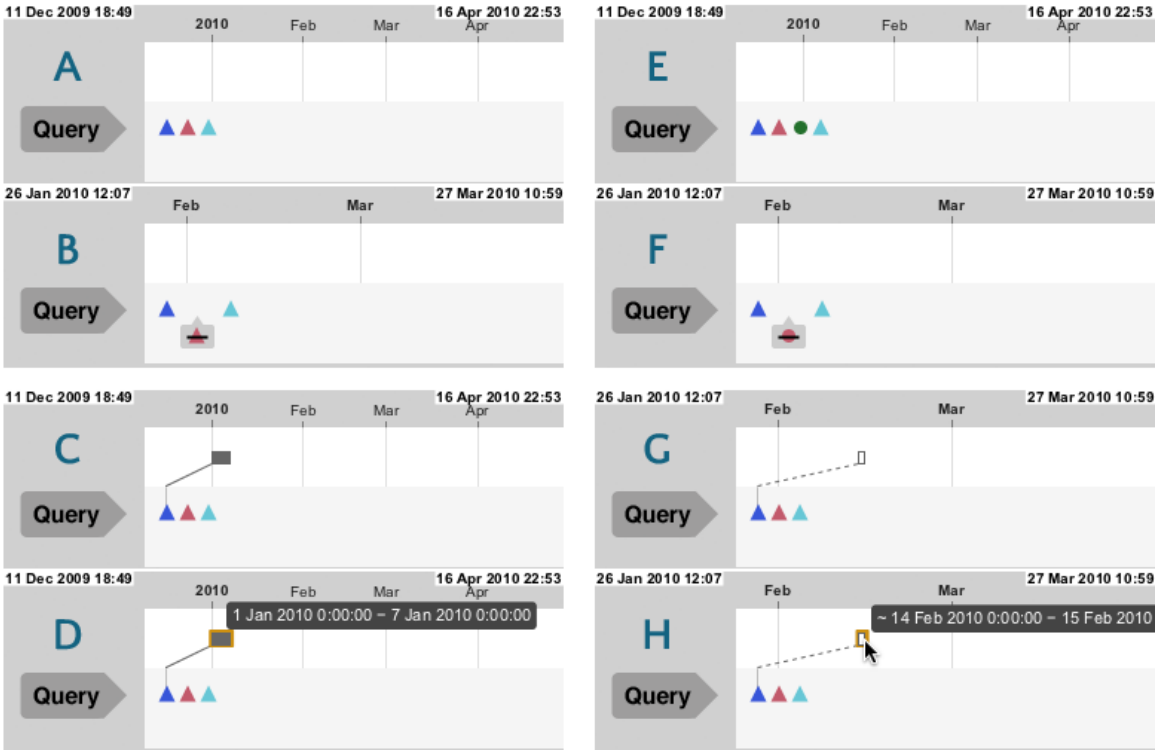

Figure 5.3: Query Representation: (A) Triangles are mandatory events. (B) Mandatory negation (*red*) is a triangle with a strike mark through the center placed in a balloon . (C) An event that has a time constraint (*blue*) has a path drawn from its glyph in the query strip to the position on the timeline that represents the beginning of its time constraint. The duration of the time constraint is rendered as a rectangle on the timeline. Solid line and filled rectangle represent mandatory constraint. (D) Query C with tooltip (E) A circle is an optional event (*green*). (F) An optional negation is a mandatory negation that uses a circle instead of a triangle. (G) An optional version of a time constraint in Query C. Dashed line and hollow rectangle are used instead. (H) Query G with tooltip

in more detail.

5.4.1 Query Representation

FTS query panel is a combination of timeline and comic strip approaches, as inspired by Similan [144] and QueryMarvel [54], respectively. The panel is split into two parts: *timeline* and *query strip*.

All events and gaps are rendered as glyphs and placed back to back on the

Figure 5.4: Gap Representation: The gaps on the left (I,J,K,L) are mandatory while the gaps on the right are optional (M,N,O,P). Mandatory gaps are filled and use solid lines while optional gaps are not filled and use dashed lines.

query strip (Figure 5.2). The timeline is used to represent time constraints only. This design can eliminate occlusion and help users read the query easily. Tooltips are also available for all these glyphs to help users interpret them. All specifications in Section 5.2.2 are converted to the following visual representations:

1. **Event**: All mandatory events are converted into triangles while optional events are converted into circles. Colors of the glyphs represent event types.

2. **Time constraint**: Each event that has a time constraint will have a path drawn from its glyph in the query strip to the position on the timeline that represents the beginning of its time constraint. Mandatory time constraints are drawn as solid lines while optional time constraints are drawn as dashed lines. The duration of the time constraint is rendered as a rectangle on the timeline. The rectangle is filled when the constraint is mandatory and hollow when it is optional.

3. **Negation**: Each negation is rendered similar to an event, but with a strike mark through the center. Consecutive negations are grouped and placed in a balloon.

4. **Gap**: Each gap is represented with a box plot (Figure 5.4). The width of the box plot represents the time using the same scale with the timeline. The box part represents minimum value and the extended part represents maximum value. An arrow is added to the end if there is no maximum value: for example, a gap more than 2days. Mandatory gaps are filled and use solid lines while optional gaps are not filled and use dashed lines.

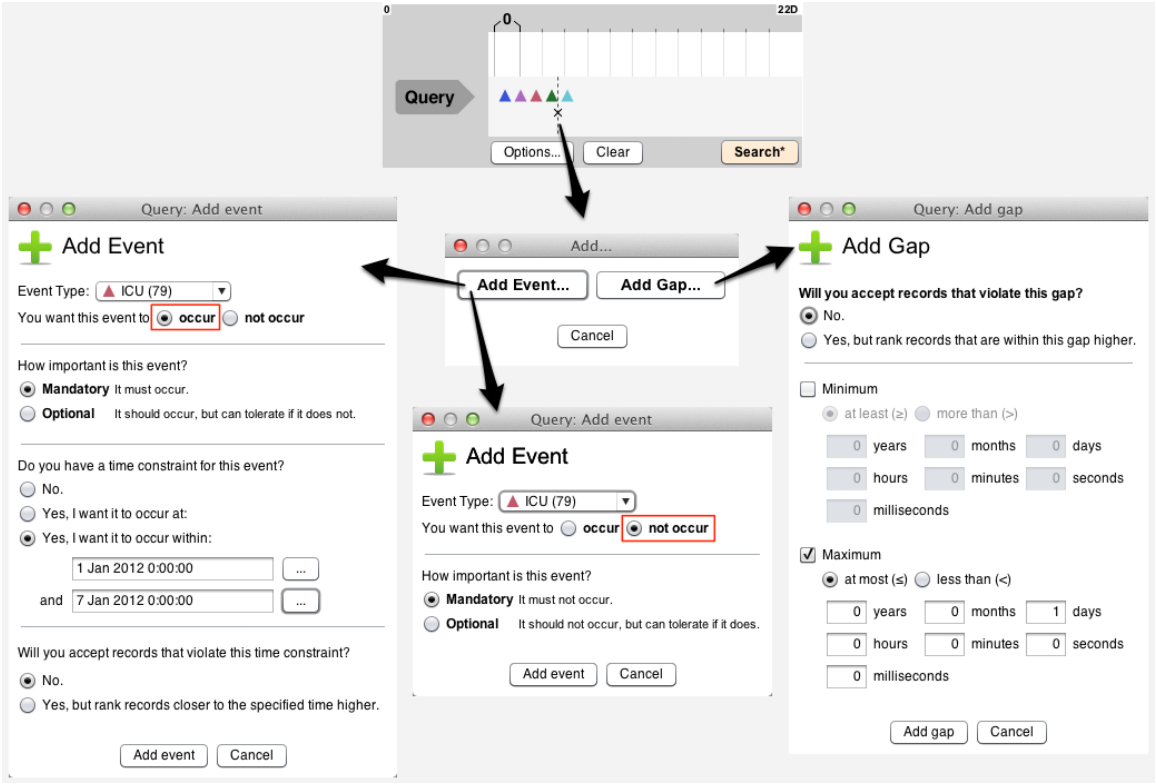

Figure 5.5: User can click on an empty location on the query strip to add an event or a gap to a query. In this example, the user is going to add something between `Floor` (*green*) and `Discharge-Alive` (*light blue*). The location is marked with a cross and an "Add..." popup dialog appear. User then choose to add an event or a gap, which will open another dialog.

5.4.2 Query Specification

The user interface supports three ways to specify a query:

1. **Create a query from scratch**: Users can click on an empty location on the query strip area to add an event or a gap to a query. Click in front of or behind existing events to add an event before and after, respectively. The clicked location will be marked with a cross and an "Add" dialog will appear (Figure 5.5). At this point, users can choose to add an event or a gap. If it is not possible to add a gap because the clicked location does not have events

on both side, the "Add Gap..." button will be disabled.

(a) *Add an event*: The "Add Event" dialog contains a few questions about the event. First, do users want this event to occur (event) or not (negation)? Second, is this mandatory or optional? The dialog box provides some explanation to help the user choose. If the users want this event to occur, another question will be asked about timing ("Do you have a time constraint for this event?"). If the answer is no, then it is done. If the answer is yes, users can set the time constraint, which can be a point in time ("occur at") or an interval ("occur within"). Then users have to answer the last question ("Will you accept records that violate this time constraint?"), which will decide whether the time constraint is mandatory ("No") or optional ("Yes."). After answering all questions and clicking on "Add event", the new event is added to the query strip.

(b) *Add a gap*: The "Add Gap" dialog contains one question that asks whether users will accept records that violate the gap. The answer to this question is used to decide whether the gap is mandatory ("No") or optional ("Yes."). The rest of the dialog are controls for specifying a time gap. Once users click on "Add gap", the new gap is added to the query strip.

User can modify or delete an event by clicking on its glyph. This will bring up a "Modify Event" dialog, which is similar to the "Add Event" dialog, but the buttons are changed from "Add event" and "Cancel" to "Apply", "Delete"

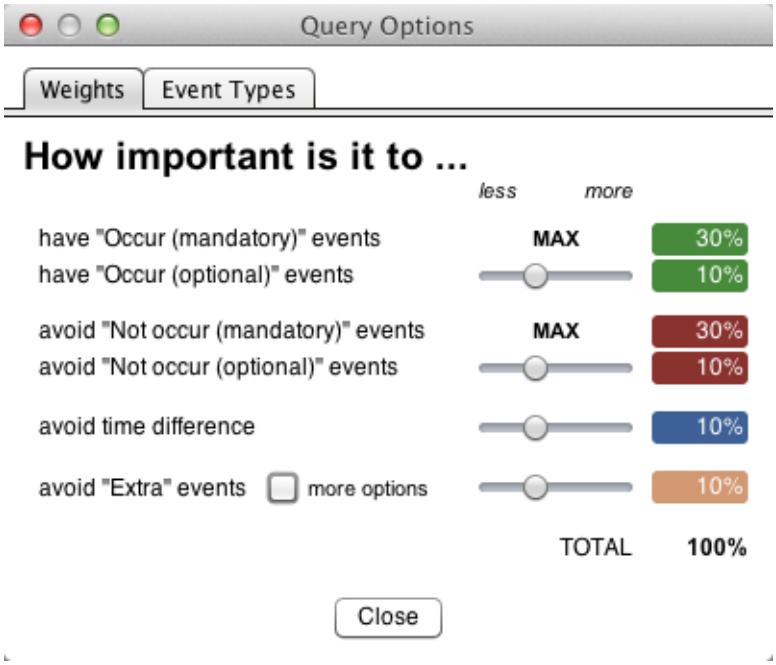

Figure 5.6: Weights: Users can adjust the importance of each kind of difference.

and "Close". User can modify the parameters, click on "Apply" to see the changes take effect and "Close" or simply click "Delete" to delete the event. The same interactions are used for modifying and editing gaps.

2. **Create a query from a record**: Instead of creating from scratch, users can right-click on any record in the list and click on "Set as query" to use that record as a template (Figure 5.7. This can be used for searching similar records as well. All events in the selected record are automatically converted into mandatory events in the query. Users can choose between two options:

   (a) *Use specific time*: Each event use its original time from the selected record as an optional time constraint.

   (b) *Use gap between events*: All gaps between events in the selected records are converted into optional gaps in the query.

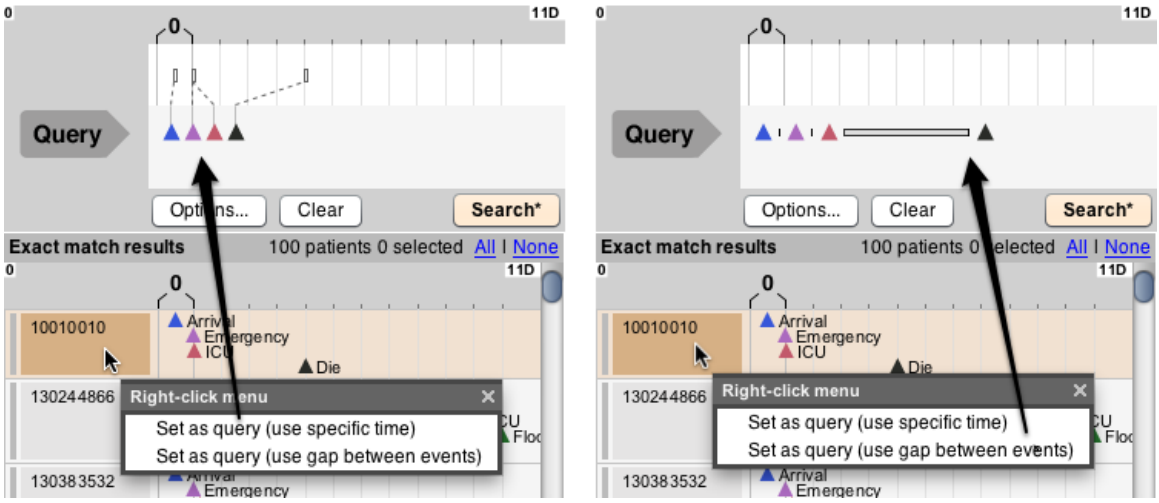

Figure 5.7: Creating a query from a record: In this example, a user chooses the record "10010010" as a template. (left) In "use specific time" mode, each event use its original time from the selected record as an optional time constraint. (right) In "use gap between events", all gaps between events in the selected records are converted into optional gaps in the query.

Users then can modify the query to suit their needs.

3. **Create a query from an event sequence in LifeFlow**: Users can pick any event sequence in LifeFlow as a starting point by right-clicking and selecting "Set as query" from the context menu (Figure 5.8). All events in the sequence are automatically converted into mandatory events. After that, users can modify the query as needed.

The user interface also includes options to adjust the weights and choose event types to include in the search (Figure 5.6). These weight sliders correspond to the weights in Section 5.3.4.

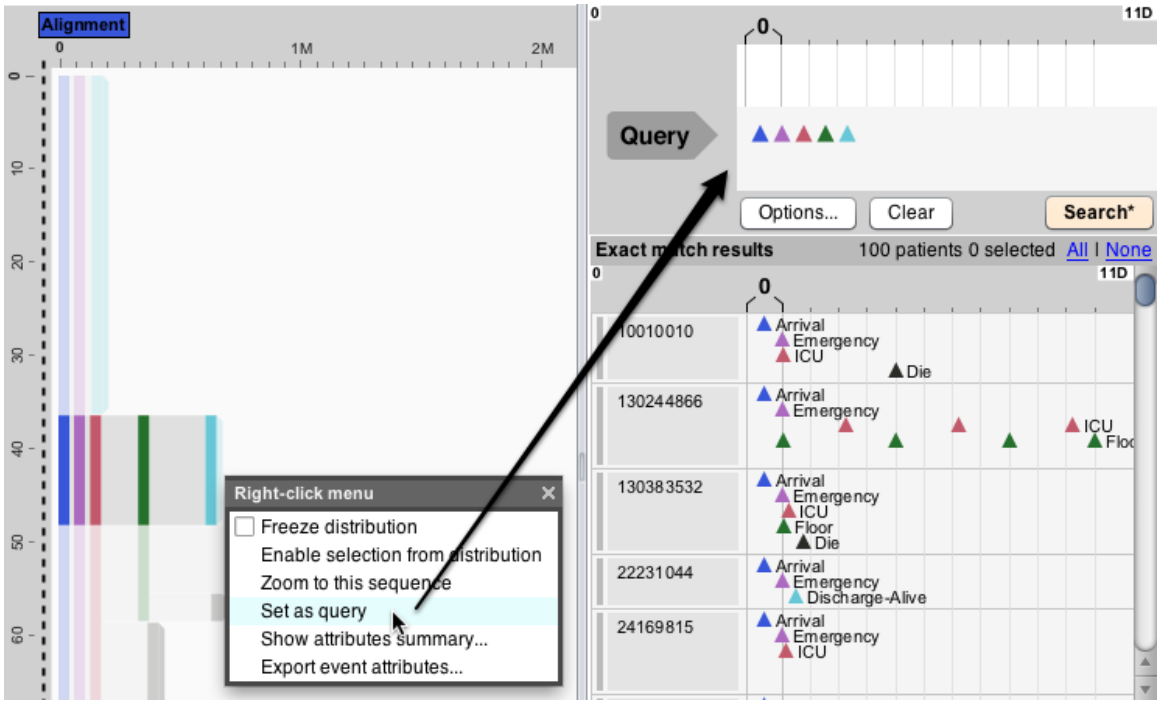

Figure 5.8: Creating a query from an event sequence in LifeFlow: In this example, a user chooses the sequence `Arrival` (*blue*), `Emergency` (*purple*), `ICU` (*red*), `Floor` (*green*) and `Discharge-Alive` (*light blue*) as a template.

5.4.3 Search Results

The results are split into two bins: exact match and other results. Exact match results are comparable to emails in inbox while other results are comparable to emails in spam folder. Users can examine the exact match results (inbox) first, then look into other results (spam folder) when they cannot find something, or want to confirm that they are not missing anything. Each bin is displayed using a list of single-record visualizations, similar to LifeLines2 and Similan. All records within each bin are sorted by their similarity score. The scores are shown in the front section of each row using ValueChart [24]. The length of each section of the colored bars represents the value (after weighted) of each of the four criteria in the similarity measure. User can place the cursor over each record to see a tooltip that

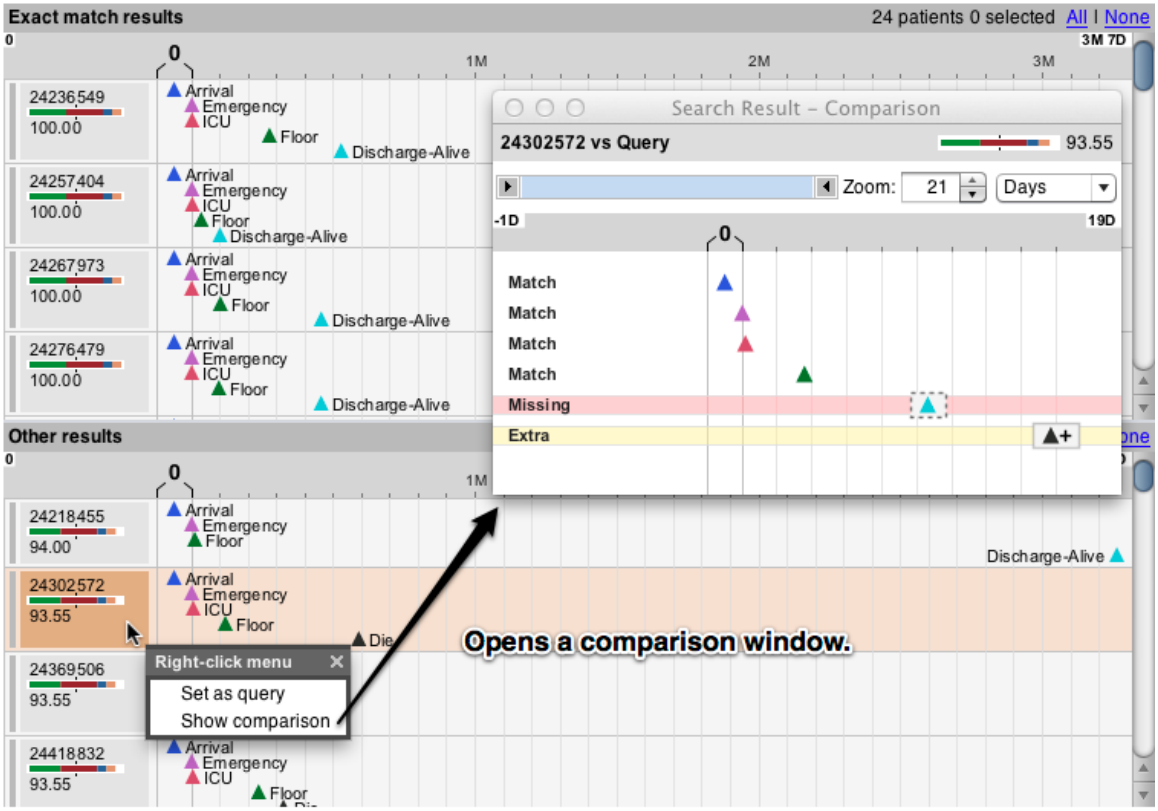

Figure 5.9: Comparison: Four events in the query are matched while one is missing (*light blue*) and another one is extra (*black*).

explains the similarity score on the left (Figure 5.2). To see detailed comparison, user can right-click and select "Show comparison" to open a comparison window that explains whether an event is a matched, extra or missing events (Figure 5.9).

5.5 Use case scenario

As mentioned earlier in Section 4.1.1, the bounce back study was the original motivation of my interest in event sequence query. Bounce back patients are patients whose level of care was decreased then increased again urgently. The common cases are patients who were transferred from the `ICU` to the `Floor` (normal bed) and then back to the `ICU` within a certain number of hours.

Therefore, I revisit this scenario to demonstrate how the FTS interface can support this task and provide the benefits of both the exact match and similarity search interfaces. Figure 5.2 shows a screenshot of a bounce back query: `Arrival` (at time 0) → `ICU` → `Floor` →≤ 2 days → `ICU`. The search results return 5 patients as exact match results and 95 patients as other results. The exact match results can be counted quickly and confidently similar to using an exact match interface and better than when using a similarity search interface. Looking at the other results shows what is right beyond the cut-off point, which is not possible when using the exact match interface. The top patient in the other results has a pattern very similar to bounce back but the return time to `ICU` was 4 days. While examining the results, the physician can easily notice this patient and use his/her judgment to decide whether to consider this case as a bounce back patient or not. If an exact match interface was used to query this dataset, this patient might have been left unnoticed.

5.6 Summary

Exact match and similarity search are the two extremes in querying event sequences. Both have their own advantages and limitations. In this chapter, I explain the design of a hybrid interface, *Flexible Temporal Search (FTS)*, that combines the benefits of both interfaces. FTS allows users to see a clear cut-off point between records that pass the query and records that do not, making it easier to count and give more confidence. All records are sorted by similarity, allowing users to see the

"just-off" cases that are right beyond the cut-off point. In order to do this, I designed a new similarity measure which includes information for each constraint whether it is *mandatory* or *optional*. The mandatory constraints are used to decide whether a record is an exact match or not. The optional constraints are used as bonus points for ranking results, but are not used for excluding a record from exact match results. The FTS similarity measure also supports negations, time of occurrences and time gaps between consecutive events. In addition, I have developed a user interface that allows users to specify FTS query by placing events on a timeline. Users can start from an empty timeline, or select one record or an event sequence from LifeFlow as a template. The user interface also supports search results interpretation and provides detailed comparison between query and each record. Finally, I have illustrated an example use case scenario to demonstrate the usefulness of FTS. More applications of the FTS are included in the case studies in Chapter 6.

Chapter 6

Multi-dimensional In-depth Long-term Case Studies

6.1 Overview

Usability testing and controlled experiments are commonly used to evaluate user interfaces. These approaches have limitations due to their short time limit. It is difficult to learn how the users will use the tool to solve more difficult problems or test advanced features that require understanding of more complex concepts from participants who have just known the user interface for only 1–2 hours. In this chapter, I describe several case studies that were conducted following the multi-dimensional in-depth long-term case studies (MILCs) approach.

I recruited participants who expressed interest in using *LifeFlow* to analyze their data, learned about their problem, then facilicated the initial phase, such as data cleaning and conversion, to help them start. After that, I provided necessary support and periodically collected feedback per convenience of the users. In some cases, we scheduled meetings (in person or sometimes virtually via Skype) to look at the data together. In some cases, the participants analyzed their data independently and later provided feedback through interviews or emails. The early participants were also involved in an iterative design of the interface where I revised many designs according to their feedback. Many useful features of the software were added from these early studies. All stories, analyses, findings and feedback are reported in this

	Domain	Data Size (records)	Duration (of the study)	Highlighted Results
6.2	Medical	7,041	7 months	Found patients reported dead before being transferred to the ICU; Suggested potential use for long-term monitoring of the ED and quality control.
6.3	Transportation	203,214	3 months	Found incidents that were reported to be longer than 100 years; Comparison between traffic agencies showed that the data were reported in inconsistent ways and could not provide a meaningful comparison.
6.4	Medical	20,000	6 months	Analyzed drug prescribing patterns and drug switching; Detected patients with possible drug holidays; Led to a spin-off project to support events with interval.
6.5	Medical	60,041	1 year	Explored patients' data from a managerial perspective in several aspects: admission, visit, mortality, diagnoses, etc.; Found that most dead patients died after admission instead of dying in the ED.
6.6	Web logs	7,022	6 weeks	Studied how visitors read children's book online; Discovered that many users also accessed the online books in backward direction.
6.7	Activity logs	60	5 months	Analyzed activity logs of two user interfaces; Highlighted the different usage characteristics between two user interfaces; Found activities that are often used together.
6.8	Logistics	821	6 weeks	Tracked cement trucks using automatic sensors and analyzed delivery patterns; Pointed out inaccurate sensory input; Detected a plant that took longer delivery time than others.
6.9	Sports	61	5 weeks	Picked interesting soccer matches; Reported interesting facts; Found an unknown defender who scored against the champions twice. Those two goals were his only two goals in three years.

Table 6.1: Summary of all case studies

chapter. Table 6.1 highlights results from all case studies.

In addition to the case study results, I have derived a process model and a set of design recommendations for exploring event sequence from usage statistics, case study results and observations of user behaviors. The process model and design guidelines are described in Section 6.11 and 6.12, respectively.

6.2 Monitoring Hospital Patient Transfers

6.2.1 Introduction

As mentioned earlier in Section 3.2, the design of LifeFlow was motivated by this patient transfer case study with Dr. Phuong Ho, a practicing physician in the emergency department at Washington Hospital Center. Dr. Ho was interested in analyzing sequences of patient transfers between departments for quality assurance. As I developed LifeFlow, I continued to work with Dr. Ho to analyze patient transfer data in more detail.

6.2.2 Procedure

I had 1–2 hour meetings with him tentatively every two weeks for seven months. Sometimes he visited the Human-Computer Interaction Lab (HCIL). Sometimes I visited him at the hospital. Before the meeting he provided me with the data that he wanted to analyze, and a few initial questions. I converted the data and during the meeting, we sat down and looked at the data together. After discussing the questions sent in advance, he would come up with additional questions and gave

feedback about the user interface, therefore closely guiding the development.

6.2.3 Data

A dataset in this study includes 7,041 patients who came to the ER in January 2010 and their 35,398 hospital events. Each record contains room assignments, time that the patient was assigned to each room, and how he/she was discharged from the hospital: dead, alive, leave without being seen (`LWBS`) and absence without leave (`AWOL`). I preprocessed the data by grouping the room numbers (e.g. ER15-P, 2G06-P) into types of room:

1. *ER*: Emergency Room, a section of a hospital intended to provide treatment for victims of sudden illness or trauma
2. *ICU*: Intensive Care Unit, a hospital unit in which patients requiring close monitoring and intensive care are kept
3. *IMC*: Intermediate Medical Care, a level of medical care in a hospital that is intermediate between ICU and Floor
4. *Floor*: a hospital ward where patients receive normal care

6.2.4 Analysis

6.2.4.1 First impression

The first time I showed LifeFlow to Dr. Ho using patient transfer data, he was immediately enthusiastic and confident that the tool would be useful for looking at

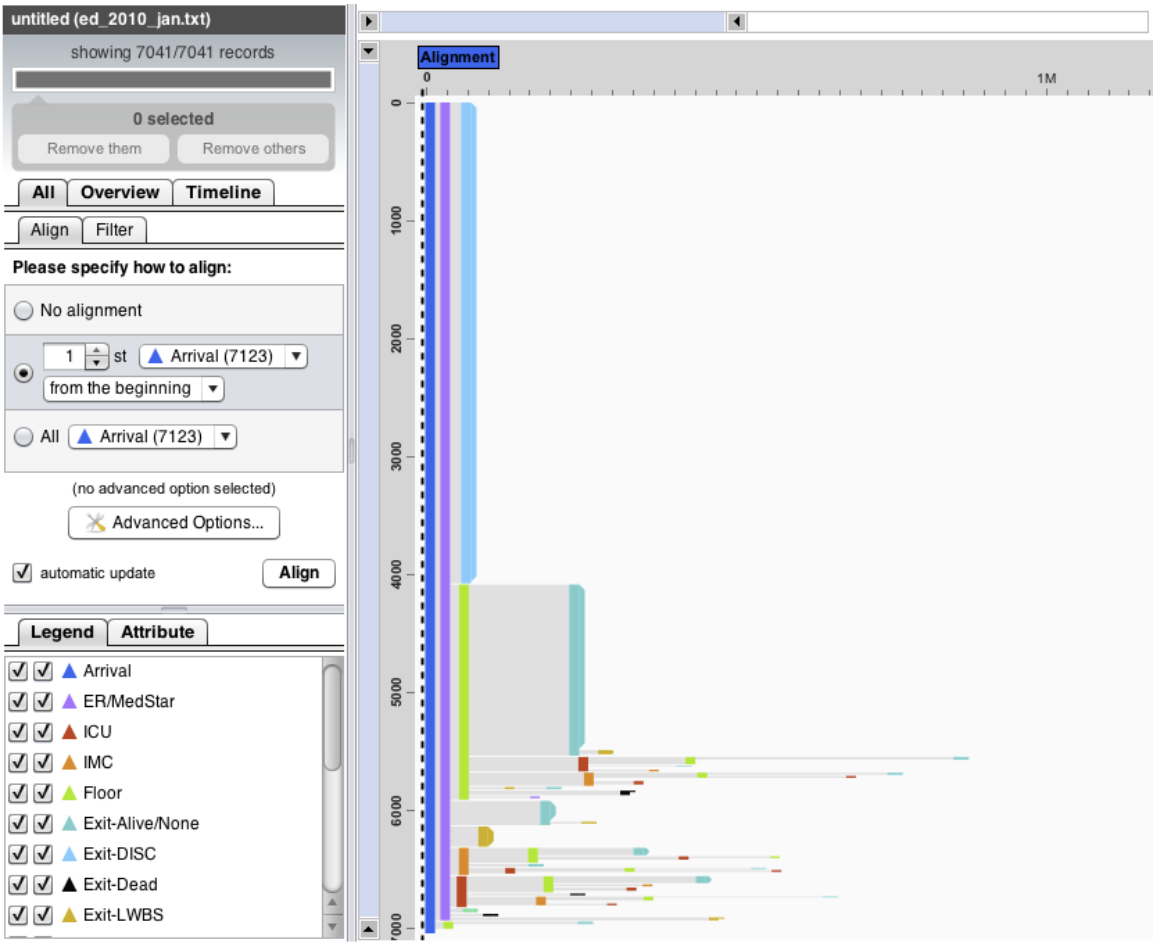

Figure 6.1: Patients who visited the Emergency Room in January 2010.

all patients who came to the hospital and, in particular, the emergency room (ER). He knew that many people would want to see the typical flow of the patients and the transfer time between rooms. In another meeting, I received additional feedback from Dr. Mark Smith, the director of the Emergency Department. Finding the bounce back patients visually in the display elicited comments such as "Oh! This is very cool!" and led to a discussion of the possibilities of using this tool to analyze hospital data in the long run.

6.2.4.2 Understanding the big picture

In a meeting, I loaded the data into LifeFlow and asked Dr. Ho to review the flow of patients in the hospital (Figure 6.1). The first thing that he noticed was the most common pattern, `Arrival` → `ER` → `Discharge-Alive`. 4,591 (65.20%) of the patients were not admitted to the hospital (discharged directly from the `ER`). This is regular and consistent with what he had expected because most of the patients who visited the `ER` were not in severe condition and could leave immediately after they received their treatment, so I removed these 4,591 patients from the visualization to analyze other patterns. The second most common pattern, `Arrival` → `ER` → `Floor` → `Discharge-Alive` (1,016 patients, 14.43%), now became more obvious. We decided to remove it too because it was also regular. We followed the same strategy, selecting regular common sequences and removing them from the visualization to detect the uncommon cases that might be irregular. Dr. Ho noticed that 193 patients (2.74%) left without being seen while 38 patients (0.54%) were absent without leave. These two numbers could be compared with the hospital standard for quality control. Then, he saw two patterns that he was interested in (`Arrival` → `ER` → `Floor` → `IMC` and `Arrival` → `ER` → `Floor` → `ICU`). These patterns correspond to another quality control metric called *step ups*, which occurs when the patients were admitted to a lower level of care (`Floor`), but later transferred to a higher level of care (`IMC` or `ICU`). Dr. Ho could quickly see from the visualization that the patients were transferred from `Floor` to `ICU` faster than `Floor` to `IMC` on average so he used the tooltip to see the distribution. He captured screenshots to

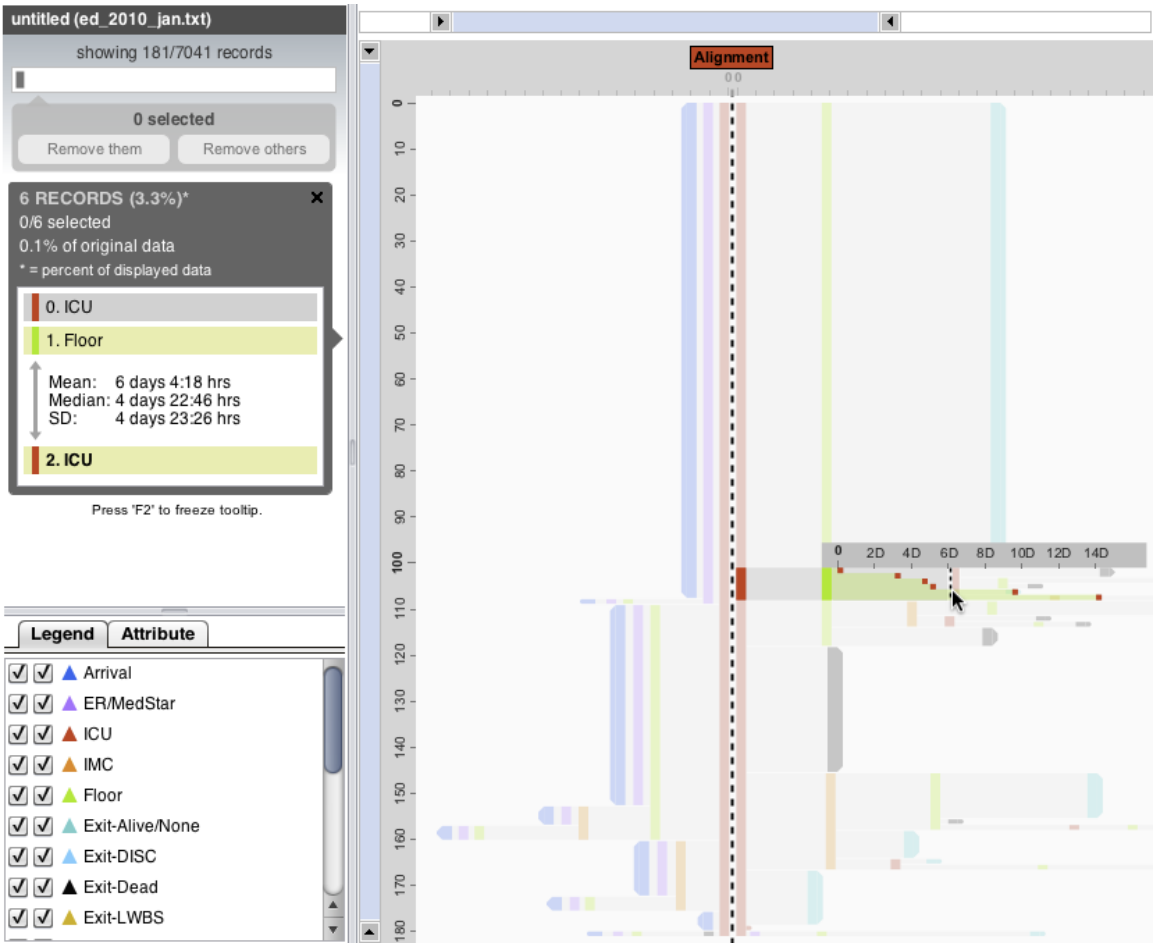

Figure 6.2: Six patients were transferred from `ICU` (*red*) to `Floor` (*green*) and back to `ICU` (bounce backs). The average transfer time back to the `ICU` was six days. The distribution shows that one patient was transferred back in less than a day.

compare with practices reported in the research literature, but also commented that the average time seemed quite good from his knowledge.

I also demonstrated the alignment and used it to analyze the transfer flow before and after the patients were admitted to the `ICU` (Figure 6.2). From the total 181 `ICU` patients, 85 (46.96%) of them were transferred from the `ER` and 119 (66.75%) of them were transferred to the `Floor` after that. However, six patients were transferred back from `Floor` to `ICU` (bounce backs). We saw from the distribution that one patient was transferred back in less than a day. Dr. Ho requested

to see these six patients in more detail, so I clicked on the bar, which highlighted these patients in LifeLines2 view and noted down these patients' ID. In addition, he also noticed some anomalous sequences, e.g. a few patient records showed a transfer to ICU after being discharged dead, pointing to obvious errors in data entry or at least reflecting the possible delays in data entry. Although we did not identify other surprising transfers (which on the other hand, reflected good quality of care), this still showed that the tool is useful for monitoring the patient transfer flow. I also received additional questions from Dr. Ho after the meeting. Some questions included clear request for alignments ("I want to see this in LifeFlow. Specifically I want to see a view that shows me where all my [ICU] patients are coming from.") indicating that he could understand what the tool is capable of and how to use it for his task.

6.2.4.3 Measuring the transfer time

Because LifeFlow can easily calculate an average time, Dr. Ho formulated many queries asking about average time, such as "Of patients who came to the ICU from the ER, what was the average time it took for transfer of the patient to the ICU? More specifically, if they went to the ICU, how long did it take from the time they had arrived at the ER to get to the ICU? Same question for IMC..." or "For all the quarters, Jan-Mar 09, Apr-Jun 09, Jul-sep 09, Oct- Dec 09 and Jan-Mar 10, I want an average time from ER to 2G [which is a specific type of ICU room]."

6.2.4.4 Comparison

Another use of LifeFlow was to compare different data sets by inspecting the difference between two side-by-side LifeFlow visualizations. Dr. Ho had a hypothesis about whether `IMC` patients were transferred faster during the day (7am-7pm) than during the night. I opened the same dataset in two windows and filtered the records by time-of-day filtering, making the two windows contain only patients who arrived during the day and during the night, respectively. We inspected the difference between the two visualizations but no significant difference was found. In another case, we compared patients who were admitted to the `IMC` at least once in four quarters: Jan–Mar, Apr–Jun, Jul–Sep and Oct–Dec 2008. We opened the four datasets in four LifeFlow windows and noticed a difference in the patients who were transferred from `ER` to the `ICU`. In the first, third and fourth quarter, these patients were later transferred to `IMC` and `Floor`, with majority were transferred to the `IMC`. However, in the second quarter, all patients were later transferred to the `IMC`, suggesting further investigation whether this occurred by chance or any particular reason.

6.2.5 Discussion

I found that the domain expert (Dr. Ho) was able to understand LifeFlow rapidly and that LifeFlow was useful to provide an overview of the entire data set, and to compare and measure the transfer times. Once the data was loaded, he could quickly see the big picture and find anomalies. Dr. Ho expressed that being able

to formulate queries easily gave him more time to look at the data and formulate new hypotheses or think about other interesting questions. Although he might have been able to answer some of the questions in SQL, it would be very difficult and error prone. He also mentioned that LifeFlow would be very useful for long-term monitoring and quality control because it provides a quick way to inspect the data from the overview. For example, hospital directors can check the waiting time and find bottlenecks in the process [39].

6.3 Comparing Traffic Agencies

6.3.1 Introduction

To illustrate how LifeFlow is not in anyway limited to medical applications, this case study was conducted with the Center for Advanced Transportation Technology Lab (CATT Lab) at the University of Maryland with the help of another computer science PhD student, Mr. John Alexis Guerra Gómez [43].

Vehicle crashes remain the leading cause of death for people between the ages of four and thirty-four. In 2008, approximately 6-million traffic accidents occurred in the United States. This resulted in nearly 40,000 deaths, 2.5 million injuries, and losses estimated at $237 billion. While traditional safety and incident analysis has mostly focused on incident attributes data, such as the location and time of the incident, there are other aspects of incident response that are temporal in nature and are more difficult to analyze. LifeFlow was used to examine a dataset from the National Cooperative Highway Research Program (NCHRP) that includes 203,214

traffic incidents from 8 agencies.

6.3.2 Procedure

I attended initial meetings with researchers from CATT Lab and learned about dataset and problems. Mr. John Alexis Guerra Gómez then facilitated this study while I was away to an internship at Microsoft Research. During that time, I provided feedback and technical support remotely. The total duration of this study was three months.

6.3.3 Data

This dataset has 203,214 records, which contain 801,196 events in total. Each incident record includes a sequence of incident management events:

1. *Incident notification*: when the agency is first notified of the incident
2. *Incident Arrival*: when the emergency team arrives on the scene
3. *Lane Clearance*: when the lanes are opened, but the incident scene may be not completely cleared
4. *Incident cleared, Incident clearance*, and *Return to normal*: all denote the end of incidents. For ease of analysis, all three event types are aggregated into the new event type *Return to normal (aggregated)*.

A typical sequence should start with `Incident Notification` and finish with `Return to normal (aggregated)`, with the possibility of having `Incident Arrival`

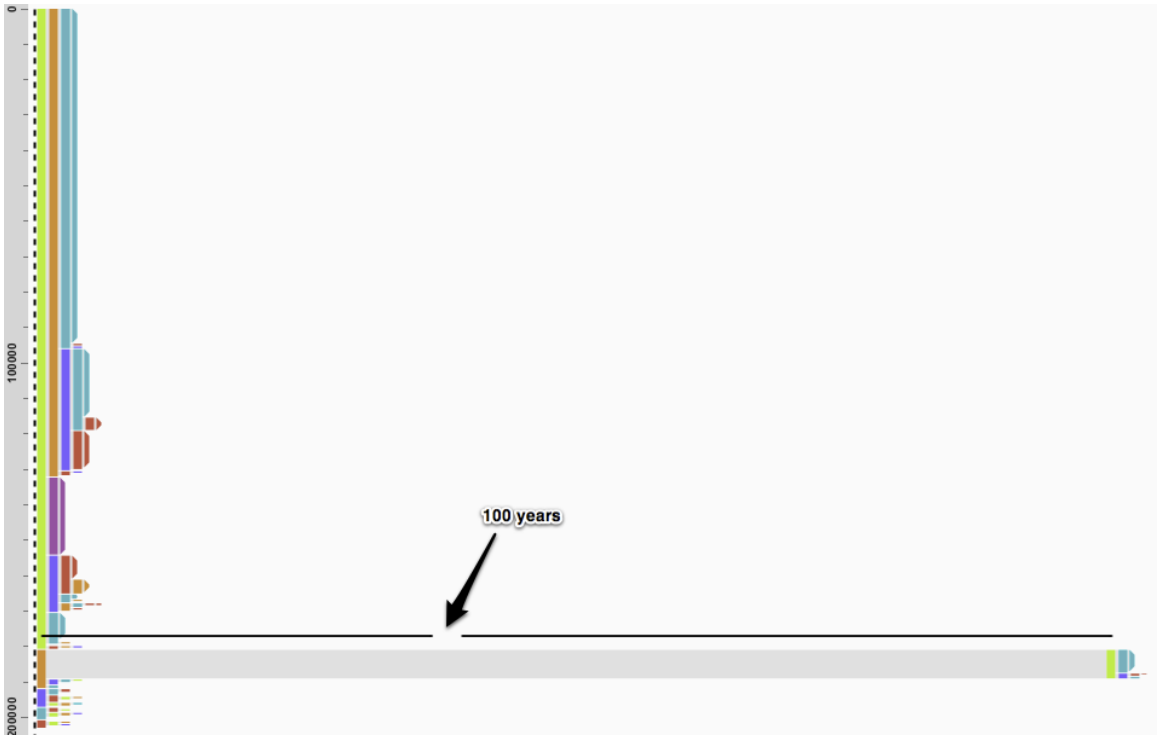

Figure 6.3: This figure shows 203,214 traffic incidents in LifeFlow. There is a long pattern (more than 100 years long) in the bottom that stands out. I was wondering if there was an incident that could last more than hundred years, we probably should not be driving any more. Investigating further, the `Incident Arrival` time of all those incidents were on January 1^{th} 1900, a common initial date in computer systems. This suggested that the system might have used this default date when no date was specified for an incident.

and `Lane Clearance` in between. In addition, the traffic incidents data include two attributes: the `agency` (represented with a letter from A to H for anonymity) and the `type of incident` (e.g. Disabled Vehicle, Fatal Accident, etc.)

6.3.4 Analysis

6.3.4.1 Quantifying data quality issues

After loading the dataset in LifeFlow, it was noticed immediately that Agency B contains 6,712 incidents that was more than 100 years long (Figure 6.3). Further

analysis showed that Agency B reported the `Incident Arrival` of those incidents as January 1^{st} 1900. Since this date is commonly used as the initial date in computer systems, this suggested that the system the Agency used to register this event might have used it as a default value when no date was specified. Considering these incidents as corrupted data, all of them were removed from the dataset. While it was easy to spot this problem, such anomalies can often remain undetected, and skew the results of even the simplest of analyses such as calculating the mean time to clearance. Similarly, 48 incidents from Agency D that are about 10 months long, in which the `Incident Arrival` occurs before the `Incident Notification`, were found and removed.

The next thing noticed from the data was that there were many incidents that lasted exactly 24 hours, which seemed unlikely. Using the Align, Rank and Filter operators, we found that those 24-hour-long incidents had `Incident arrival` events occur in the first hour of the day (e.g. 12:30 a.m. April 10 2009) and `Incident notification` events happened in the last hour of the same day (e.g. 11:50 p.m. April 10 2009). This observation seemed to suggest that there were data entry problems with those incidents, indicating that the operator failed to—or was not able to—record the correct date of an event (e.g. 12:30 a.m. Apr 11, 2009 as opposed to 12:30 a.m. Apr 10, 2009). Similar errors were discovered for paths that are about 12 hours long, in which case the errors seem to be problems choosing between AM and PM in the date. Those anomalies were found quite easily by Mr. Guerra Gómez, the computer scientist developer, who had no experience in transportation data. Finding such errors using traditional tools like SQL or manual analysis can

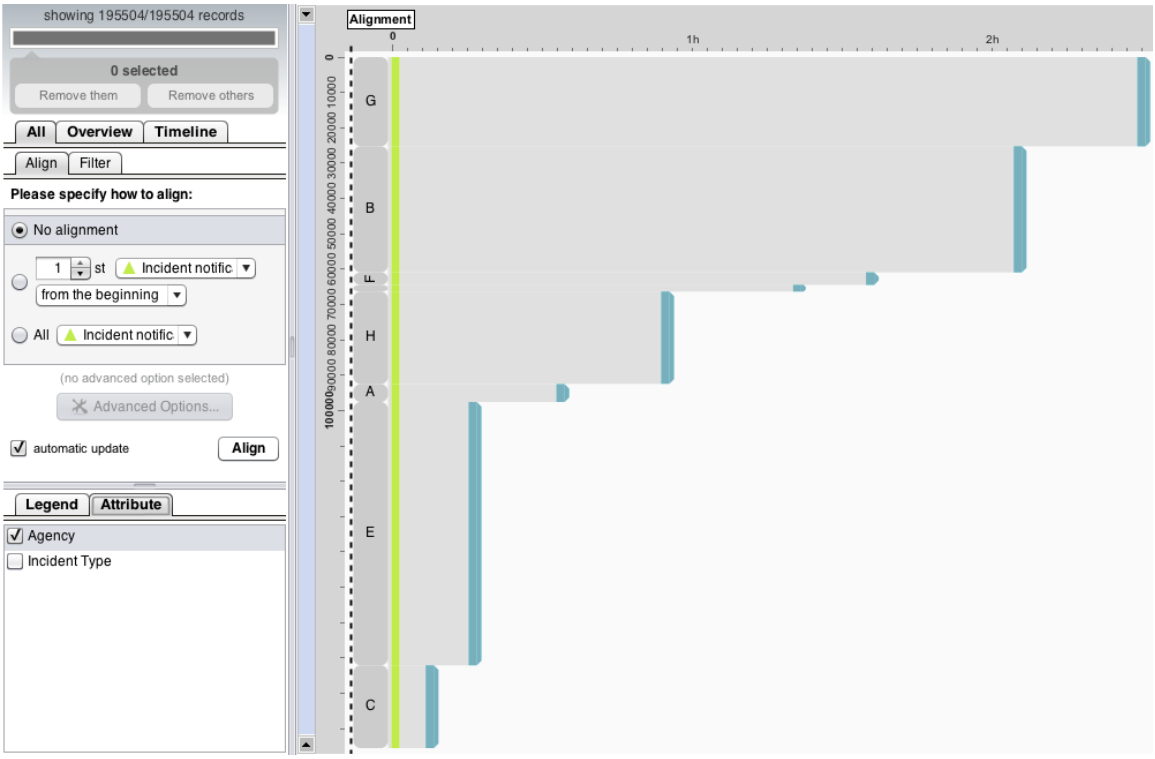

Figure 6.4: LifeFlow with traffic incidents data: The incidents are separated by agencies (A-G). Only `Incident Notification` and `Return to normal (aggregated)` events are shown. Other events are hidden. The agencies are sorted by a simple measure of agency performance (average time from the beginning to the end). Agency C seems to be the fastest to clear its incidents, followed by E, A, H, D, F, B and finally G.

be very difficult and time consuming, and requires experienced analysts who would suspect the existence of such errors.

6.3.4.2 Ranking the agencies' performance

After cleaning the data, we used the time from when the agencies were notified until the final clearance of the incidents as a performance measure. The time when the agency was notified can be indicated by the `Incident Notification` event. In order to compare the agencies performance, we first removed the inconsistent data (incidents that do not start with `Incident Notification`), which could be

performed easily using the equal height overview feature. After the steps above, the visualization of the data can be seen in Figure 6.4. Incidents are grouped by agencies. We showed only two event types: `Incident Notification` (*green*) and `Return to Normal (Aggregated)` (*blue*), so the horizontal length of each agency's path represents the average time from incident notification to final clearance, which reflects the performance measure for that agency. We then sorted the agencies according to the length of their paths, resulting in the fastest agency (shortest path) on the top and the slowest agency (longest path) in the bottom. From Figure 6.4 we could see that Agency C was the fastest agency to clear its incidents, taking about 5 minutes on average, while the slowest one was Agency G with an average of about 2 hours and 27 minutes.

To investigate deeper into Agency C's data, we removed the data from other agencies and looked into the different incident types reported (Figure 6.5). Most of the incidents that Agency C reported are `Disabled Vehicles` which had about 1 minute clearance time on average. Looking at the event distribution, we also found that a large number of the incidents reported `Clearance` immediately after `Incident Notification`. This observation made us wonder if there is any explanation for these immediate clearances, and encouraged further analysis.

In a similar fashion, Agency G, which seemed to be the slowest agency, were investigated. Agency G classified their incidents in only two types: `Non-ATMS Route Incident` and simply `Incident`. The `Incident` incidents had an average length of about 38 minutes, which is very fast compared to the other agencies. However, the `Non-ATMS Route Incident` incidents took on average 5 hours 14 minutes to clear.

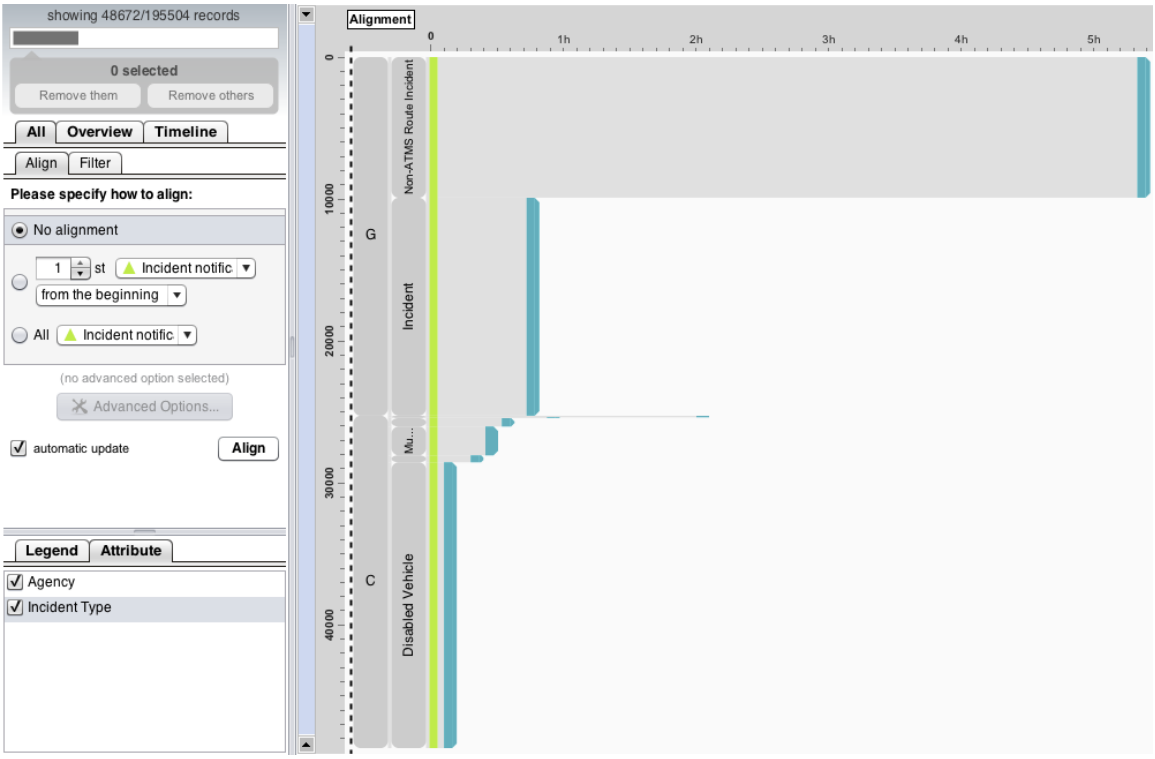

Figure 6.5: LifeFlow with traffic incidents data from Agency C and Agency G: Only `Incident Notification` and `Return to normal (aggregated)` events are shown. The incidents are also grouped by incident types. Most of the incidents that Agency C reported are `Disabled Vehicles` which had about 1 minute clearance time on average.

This made us realize that when using the average time of all incidents from Agency G without considering the incident types, Agency G seemed to be slower than other agencies. While in fact, Agency G performed quite well for incident type `Incident`.

These evidences showed that the agencies reported their incident management timing data in inconsistent ways, e.g., Agency C's incidents with 1 minute clearance time. Beside timing information, the same problem occurred with incident types as the terminology was not consistent across agencies or even local areas. Difference might stem from differences in data entry practices, terminology variations, or missing data. Policies are needed to decide if, and how, to normalize the data and tools will be needed to manage this normalization process and provide audit trails documenting the transformations. At the time of this analysis, these inconsistencies among agencies restricted further meaningful performance comparison.

6.3.5 Discussion

Although this data analysis was limited and preliminary, domain experts from the CATT Lab were conducting a more formal analysis of the data. They reviewed this work and stated that they wished LifeFlow was available earlier on when they started their own analysis. They confirmed the existence of anomalies that were found in the data, and stated that their elimination was non-trivial when using SQL because they had to expect the errors in advance and be careful to exclude them from their analysis. However excluding all the possible erroneous sequences in a SQL query would be very difficult. In the end, they needed to review the

results of SQL queries to ascertain that there were no longer any errors. Without LifeFlow, this kind of review and identification of unexpected sequences would be almost impossible. Finally, they mentioned that LifeFlow would allow them to ask more questions faster, and probably richer questions about the data. LifeFlow was also able to reveal unexpected sequences that may have been overlooked, but the tool also suggested that their prevalence is limited. I believe that using LifeFlow can assist analysts in exploring large datasets, such as the NCHRP traffic incident information, in ways that would be very difficult using traditional tools and might allow analysts to find richer results in less time.

6.4 Analyzing Drug Utilization

6.4.1 Introduction

The U.S. Army Pharmacovigilance center has assembled a data warehouse of medical records and claims representing the clinical care of 12 million soldiers, dependents and retirees over a period of five years. They use that data to promote patient safety in the military and to serve as a federal partner for the FDA's Sentinel program studying drug safety signals across the U.S. population. Mr. Sigfried Gold was a computer scientist at Oracle Corporation working with the PVC. He was helping the PVC epidemiologist and pharmacists explore drug exposure and diagnostic events in longitudinal patients. The dataset consists of 20,000 patients over 7 years with 5-100 medical events per patient.

His initial goal was to look at causal relationships between drug exposure

and adverse events. However, after some preliminary attempts, we decided that causal analysis would require more complex analysis which might go beyond the capabilities of LifeFlow. We learned that LifeFlow is great for looking at multiple sequences when each sequence is one process, but less effective when each sequence consists of events from multiple independent processes because the negligible order of events from different processes will be used to aggregate the sequences and lead to a large number of infrequent irrelevant patterns. It also experiences difficulties when there are a large number of event types, which makes color-coding become more challenging. Causal analysis would require an analysis with a large number of event types and possibly includes events from multiple processes.

As a result, we focused on a simpler case which LifeFlow seemed to be directly applicable by looking at drug utilization: drug prescribing patterns and drug switching, in particular.

6.4.2 Procedure

Since the data in this study were very sensitive and confidential, I was not allowed to look at the data by myself. I helped Mr. Gold start his analysis by generating a fake dataset and demonstrating how to use the software. After he was familiar with the software, I provided technical support as needed and let Mr. Gold and the PVC team perform an analysis on their own. We communicate by emails every 2–3 weeks and had in-person meetings several times at HCIL and PVC. The total duration of this study was six months before an extension project was spin-

offed. In the end, I interviewed Mr. Gold and reported the following results.

6.4.3 Analysis

6.4.3.1 Drug prescribing patterns

The PVC team looked at the drug prescribing data that record when medications were prescribed. With drugs that the PVC team expected to be chronic, LifeFlow visualization pointed out that there are always more than half of the patients who have gaps within their exposures. These could potentially be drug holidays, i.e., when a patient stops taking a medication(s) for a period of time, anywhere from a few days to many months or even years if they feel it is in their best interests.

The following question was immediately raised: If the gaps were really representative of holidays or they were just brief period between two prescription that they have enough pills to keep going? LifeFlow helped the PVC identify the patient cohorts with possible drug holidays and also show potential for taking part in a deeper analysis to answer this question.

6.4.3.2 Drug switching

Another study focused on how patients switch from one drug to another. One interesting characteristic of this dataset is that some event types are interval events, not just point events. For example, an event, such as `Drug A` or `Drug B`, has start and end time, which indicate the exposure period. Two drugs can overlap, increasing the complexity of the analysis. There were also some periods that patients were not

taking any drug.

LifeFlow was designed for point events, which each event has only one time. We came up with a workaround that converted the data from interval events (`Drug A` and `Drug B`) into point events (`Drug A only`, `Drug A & B`, `Drug B only` and `no drug`) before using LifeFlow. This approach was tolerable in the preliminary phase. However, the conversion increased the overhead in data processing time when any change had to be made and users lost flexibilty of LifeFlow that can include/exclude any event type when needed. For example, when analyzing a dataset with drug A, B and C, including only drug A and B and excluding drug C will require one round of data preprocessing and including drug C back will require another. Users ended up spending most of the time preprocessing data, making the analysis more time consuming and less engaging. There were also several useful features for interval that Mr. Gold would like to have. Therefore, we decided to fork the LifeFlow project to create an extension of LifeFlow that can support interval, which is beyond the scope of this dissertation, to support further analysis.

6.4.4 Discussion

This case study demonstrated the ability of LifeFlow to analyze drug prescribing patterns, which led to a question whether patients are having drug holidays.

> "Statistical techniques for dealing with longitudinal data generally focus on changes in continuous variables over time, and the problem of identifying patterns of sequence and temporal spacing in categorical events is

not handled by standard techniques and software. This problem arises a lot in analysis of health care data, and this tool opens up a kind of study that just hasn't been possible before," said Mr. Gold.

Working with Mr. Gold also help me identify LifeFlow's limitations in handling several event types, multiple processes and intervals. "LifeFlow is good for complex temporal relationships among a small number of variables (event types)." Understanding these limitations led to a spin-off project beyond the scope of my dissertation that extended LifeFlow idea to support intervals.

6.5 Hospital Readmissions

6.5.1 Introduction

Hospital readmission shortly after discharge is well known to be a factor in rising healthcare costs and is felt to be a marker of inpatient quality of care. Dr. A. Zach Hettinger, a physician at the MedStar Institute for Innovation (MI2), was interested in the hospital readmissions study. He participated in this LifeFlow case study to visualize emergency department patient visits over time and planned to look for patterns in return visits and across chief complaints, and also to look at high risk populations.

6.5.2 Procedure

I scheduled tentative biweekly meetings for coordination. Our methods for communication were email, phone, Skype and in-person meetings. In the first few

months, I developed an understanding of medical problems while explaining LifeFlow to Dr. Hettinger. During this period, I learned new requirements that are not handled sufficiently in LifeFlow and developed new features to support them. After each new feature was developed, I would demonstrate it and ask for feedback in the following meeting. This phase lasted for several months. We also had to request access to patient information, which caused some delay. Towards the end of this study, I arranged a few Skype sessions with Dr. Hettinger, in which he performed analyses and shared his screen. I collected results from those sessions and report them in Section 6.5.5 and 6.5.6. The total duration of this study was one year.

6.5.3 Before the Study

Prior to this study, the hospital was using canned reports involving 72-hour returns. Dr. Hettinger explained that he was looking at the data using these steps:

1. Pull the data from *Azyxxi*, an EMR system, in a table view which contains the following columns:
    - Patient ID
    - Diagnosis
    - Visit Date #1
    - Attending Physician #1
    - Visit Date #2
    - Attending Physician #2
    - and many more columns
2. Sort by the date that the he saw the patients (Visit Date #1).

3. See if the patients came back. (Visit Date #2 is not blank.)

4. Click on each row to read the diagnosis to learn why they came back plus more information. This method can be slow and sometimes not helpful.

   This method also cannot answer questions such as

- When they see me (Dr. Hettinger), is this their first visit to the hospital? The table view did not show the data before this visit.
- Azyxxi can show if the patient came back once or did not come back, but could not show if they come back for the 2nd, 3rd, ... time.
- How many of them came back after seeing me (Dr. Hettinger)? Cannot see the proportion easily.
- Cannot easily filter by diagnosis type. For example, show patients who were diagnosed with heartburn only.

  With LifeFlow, the questions above can be answered easily.

6.5.4 Early Progress

In the first few meetings, we discussed the data format that LifeFlow can accept and used a simulated dataset for demonstration. The feedback from these meetings led to the development of *DataKitchen* [1], and new LifeFlow interactions and other improvements:

- Display clear separation between hospital visits (Section 3.7).

[1] A data conversion tool developed by Hsueh-Chien Cheng, another PhD student

- Allow users to filter alignment point by event attributes. For example, attending physicians and diagnoses are attributes of the event `Arrival`.
  - Find `Arrival` events when patients visited a specified physician (such as Dr. Hettinger) and align to see what happened after arrival.
  - Find only arrival events when patients visited with a particular diagnosis (such as chest pain) and see what happen after arrival.
- Selection from distribution (Section 3.7)
- Add/Modify custom attributes (Section 3.7)
- Display attribute summary (Section 3.7)
- New tooltip design

6.5.5 Analysis: Personal Exploration

Dr. Hettinger envisioned using LifeFlow for two applications. The first one is for a physician to review his/her own performance. The second one is using it for administrative purposes (Section 6.5.6).

Dr. Hettinger used LifeFlow to select only his patients using the advanced alignment. He set the alignment point to be a visit to the hospital when the attribute "attending physician" was him. Then he looked for patients who returned within 72 hours after being discharged. He did not find one, which reflects good performance. To continue the analysis, he selected a few patients who returned soon after their discharges and compared the diagnoses before they were discharged and when they

came back to see if these two visits were related. If they came back with the same complaints, it could indicate that the physician missed something during the patient's prior visit. LifeFlow provided the physician with more information than the canned reported that he was previously using.

6.5.6 Analysis: Administration

Another aspect that LifeFlow can be used for is to look at the data from a managerial perspective and see the overall performance of the hospital. We used an anonymized dataset which has 92,616 patients visits spread over 60,041 records of patients in this analysis. The duration of this dataset is one year.

6.5.6.1 Revisit

To see the proportion of patients who visit once, twice and so on, we aligned them by the first `ED Registration`. "What I don't understand is why the patterns are not sorted by number of visits?" said Dr. Hettinger. I then explained that the patterns are sorted by number of patients by default, but we could change it. So, we changed the ranking to "Max time to the end" and saw the patterns sorted by number of visits (Figure 6.6). The tooltip displays more information when the cursor is placed over the bar. For example, in Figure 6.7, 1501 patients (2.5%) visited four times.

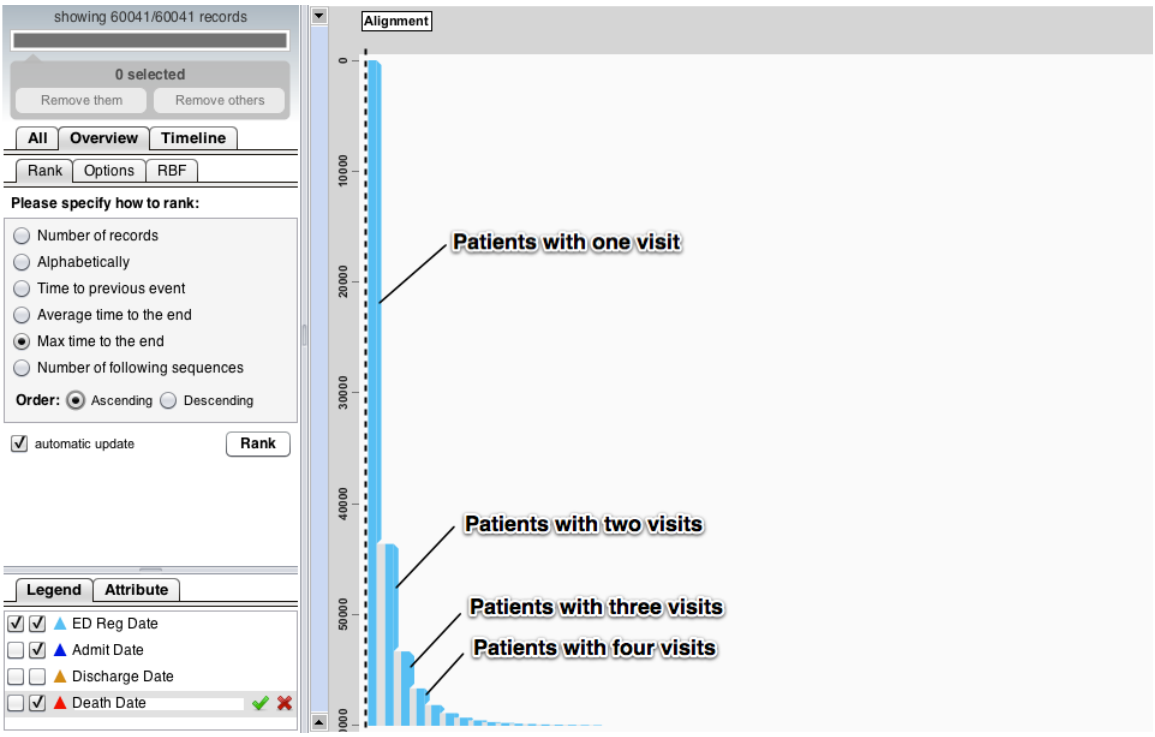

Figure 6.6: Patient records aligned by the **first** `ED Reg date` (*light blue*): The bar's height represents the number of patients. The visualization shows the proportion of revisits.

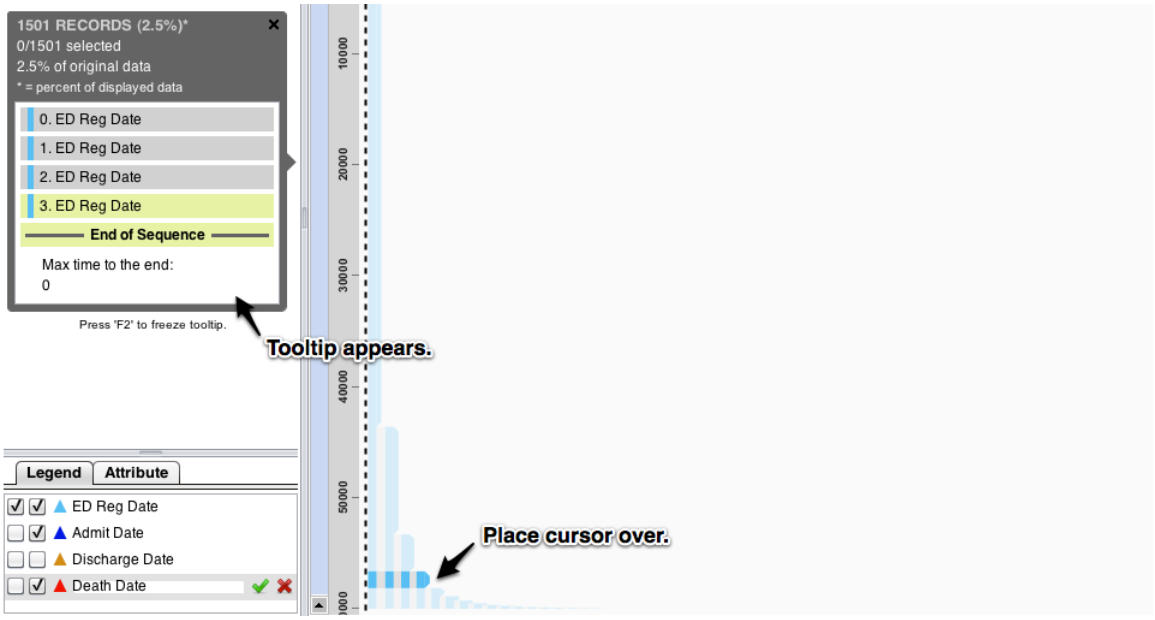

Figure 6.7: Tooltip uncovers more information: Place cursor over a sequence in Figure 6.6 to see more information from tooltip. For this sequence with four visits, there were 1501 patients, which is 2.5% of all patients.

6.5.6.2 Revisit by recurring patients

Another task was to see the number of all visits by patients with at least n visits. This could tell how many visits that the recurring patients were accounted for. We excluded all events except `ED Reg (registration) date` and aligned by **all** `ED Reg date` (Figure 6.8. The patient records were duplicated depending on their number of visits. This time the bar's height represents the number of visits instead of the number of patients. The first bar after the alignment point represents the number of visits by all patients who had at least one visit. The second bar represents the number of visits by all patients who had at least two visits. The tooltip allows access to more useful information (Figure 6.9). "We can quickly make a nice table out of this," said Dr. Hettinger. "Patients with at least six visits, which is 2% of patients, account for 4.3% of visits. Patients with at least five visits, which is 3% of patients, account for 6.6% of visits. Patients with at least four visits, which is 5.6% of patients, account for 10.2% of visits and so forth. These are statistics that people enjoy looking at and can be compared between hospitals."

In Dr. Hettinger's opinion, getting the number of visits by the top 5% or more frequent patients is more interesting than knowing who the top 5% are. "If you wanna say I just want the top 5%, you are gonna get some random number like the top 5% have 6.2 visits. That is kinda useless number. It is actually a much better way to ask the other way round that we just went through down here" said Dr. Hettinger.

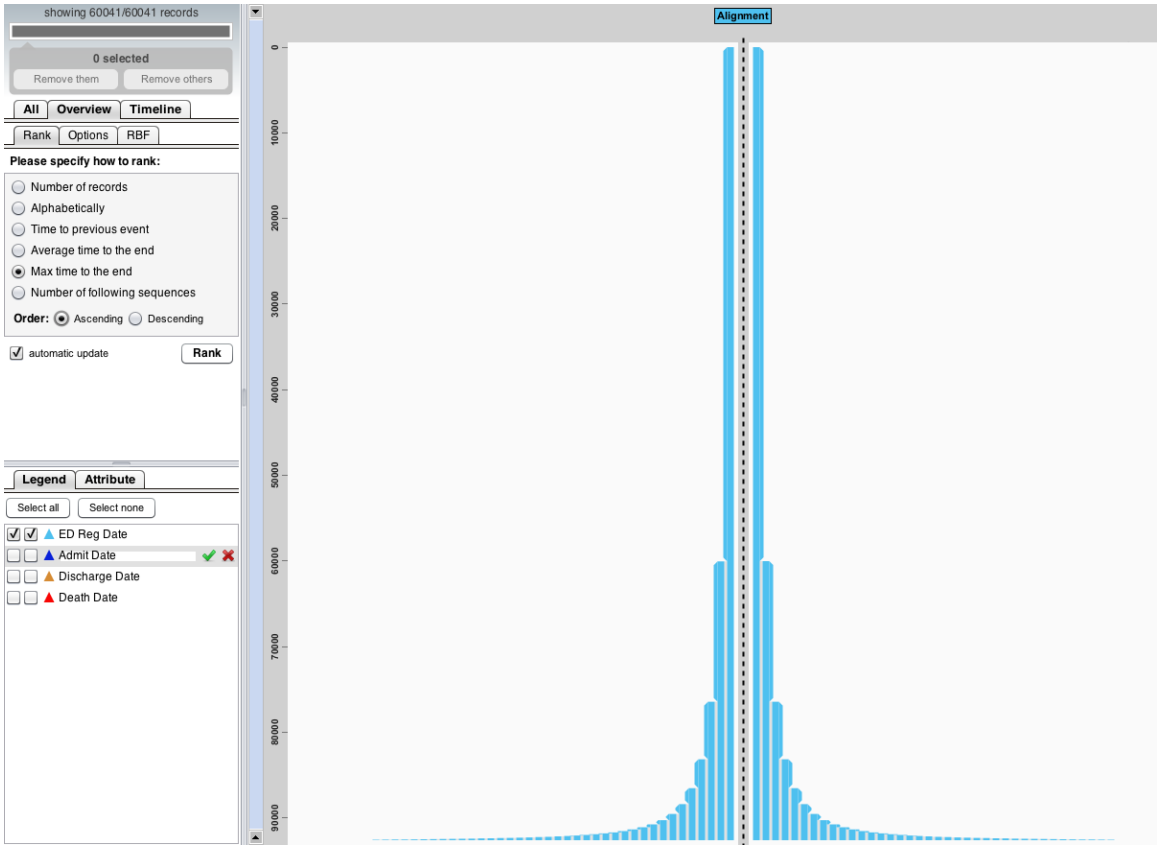

Figure 6.8: Patient records aligned by the **all** `ED Reg date` (*light blue*): The bar's height represents the number of visits. The first bar after the alignment point represents the number of visits by all patients who had at least one visit. The second bar represents the number of visits by all patients who had at least two visits.

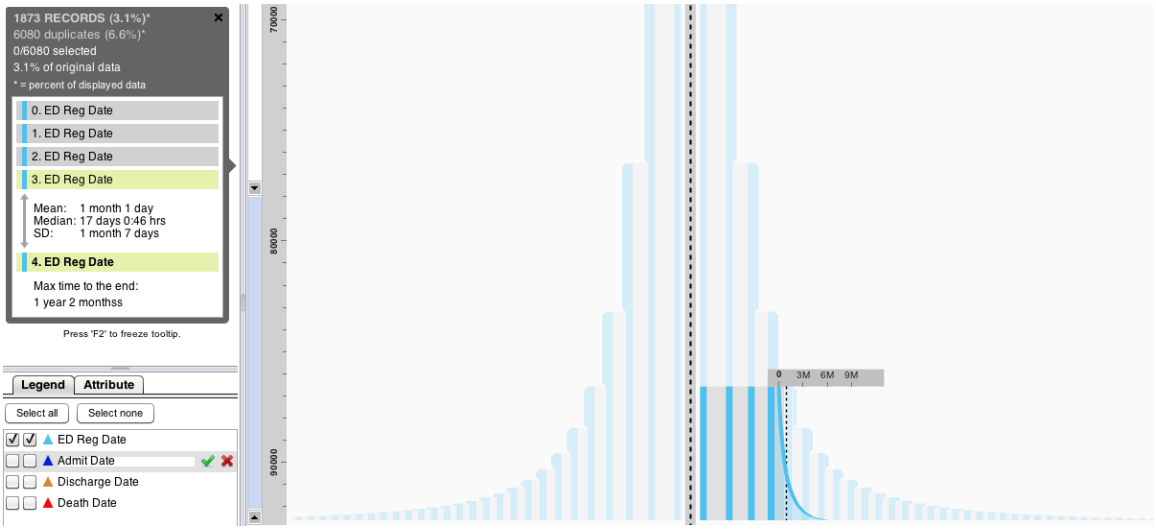

Figure 6.9: Access more useful information using tooltip– Patients with at least five visits, which is 3.1% of patients, accounted for 6.6% of visits.

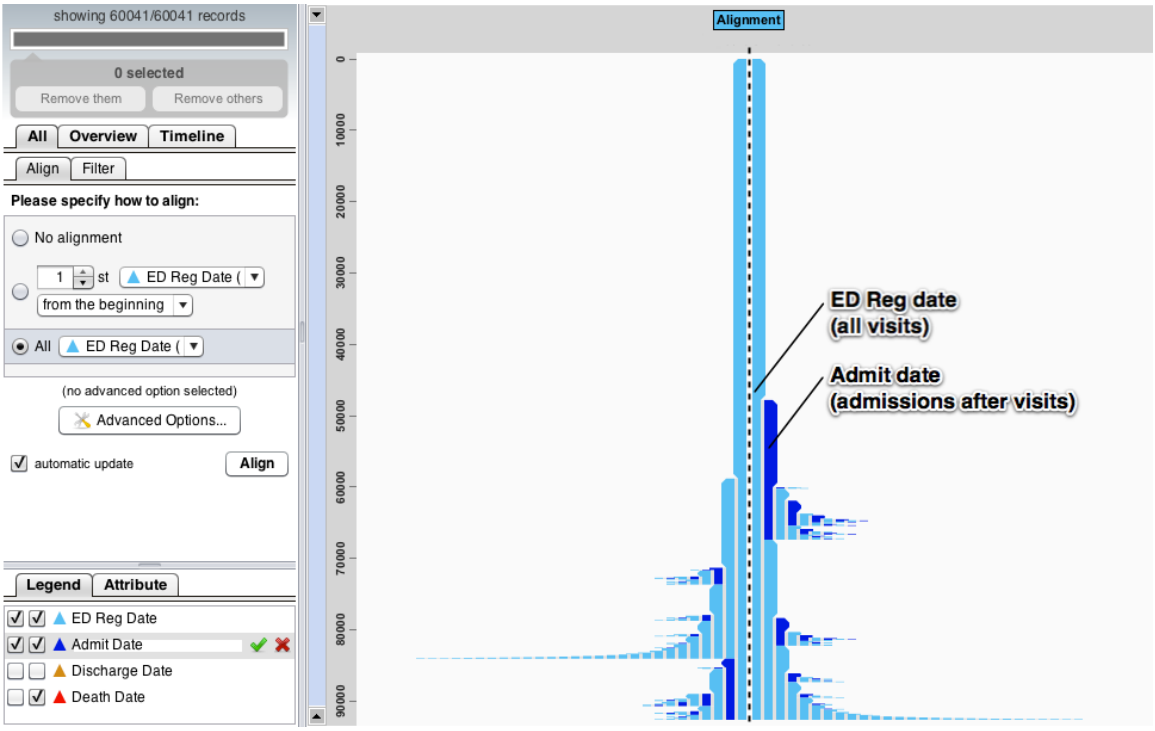

Figure 6.10: Patient records aligned by **all** `ED Reg date` (*light blue*): The total height represents total number of visits. The `ED Reg date` bar (*light blue*) on the right of the alignment represents all visits. The `Admit date` (*blue*) bar on its right shows 19,576 admissions out of a total of 92,616 visits.

6.5.6.3 Admission

Next, we looked at the proportion of people that came to the ED and were admitted We still aligned the data by **all** `ED Reg date` but included `Admit date` (Figure 6.10). The total height represents the total number of visits. The `ED Reg date` (*light blue*) bar on the right of the alignment represents all visits. The `Admit date` (*blue*) bar on its right shows admissions after visits. There were 19,576 admissions out of a total of 92,616 ED visits.

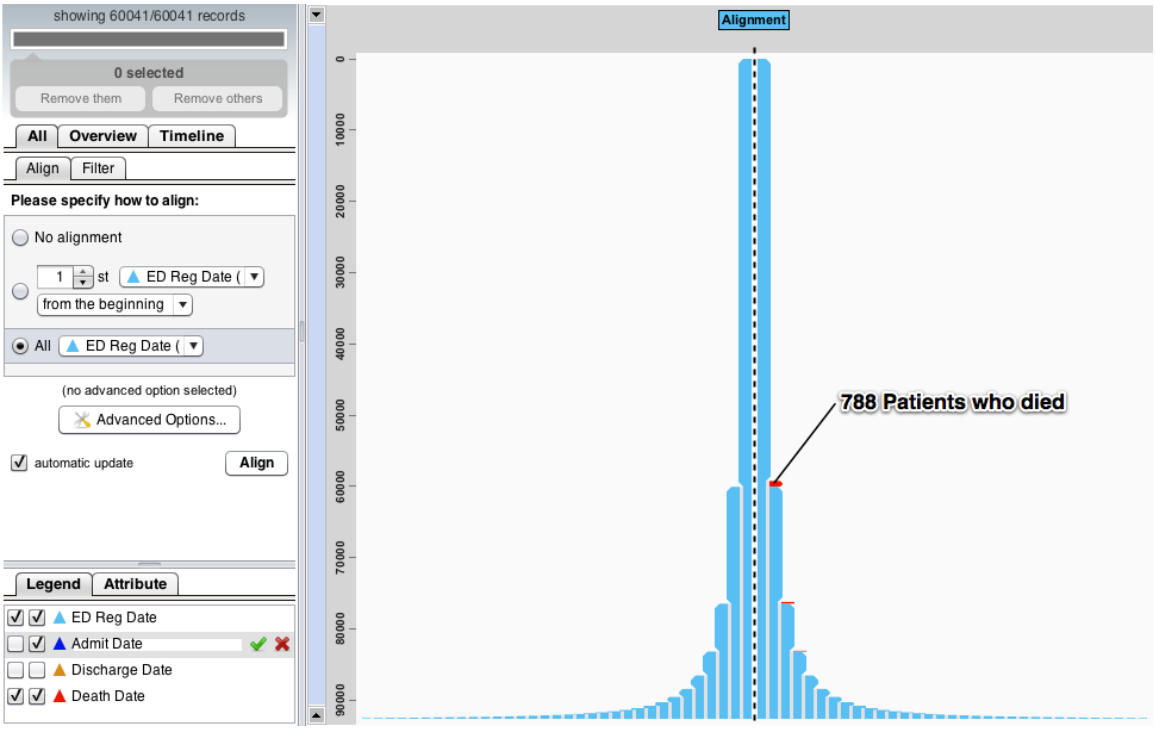

Figure 6.11: Patients who died: Patient records are aligned by **all** `ED Reg date` (*light blue*). The total height represents total number of visits. From the total of 92,616 visits to the ED, there were 788 deaths (*red*).

6.5.6.4 Mortality

Using the same alignment by **all** `ED Reg date`, the `Admit date` events were excluded but `Death date` events were included. The visualization shows that, from the total 92,616 visits to the ED, 788 patients died. This number includes both patients who died in the ED and patients who died after admission (Figure 6.11).

We selected these 788 patients and labeled them by setting an attribute `Status` to "Die", so the patients could be split into dead and alive patients using attribute `Status`. The alignment was then changed from **all** registrations to the **first** registration. We zoomed into the dead group, where the visualization shows a proportion of patients who died on their 1st, 2nd, 3rd visits and so forth (Figure 6.12).

Figure 6.12: Patient records aligned by **first** `ED Reg date` (*light blue*): The total height represents number of patients.

Next, we added the `Admit date` events back to the visualization (Figure 6.13) and found that the majority of the patients died after admission, not in the ED as expected. "You kinda think these people who are coming in and die are gonna die in the emergency department, but actually looks like large majority of them were admitted," said Dr. Hettinger.

We could also see the proportions of patients who were admitted on their second visits and died, or patients who were admitted twice and died. While the major patterns can be detected easily, the rare patterns are more complex and more difficult to interpret. Dr. Hettinger referred to these patterns as "painful to watch". So, we changed the alignment to `Death date` to help us focus on what happened before they died. Of the people that died, 76% died in the hospital (admitted then died). The rest never were admitted and died in the ED. If the data are more

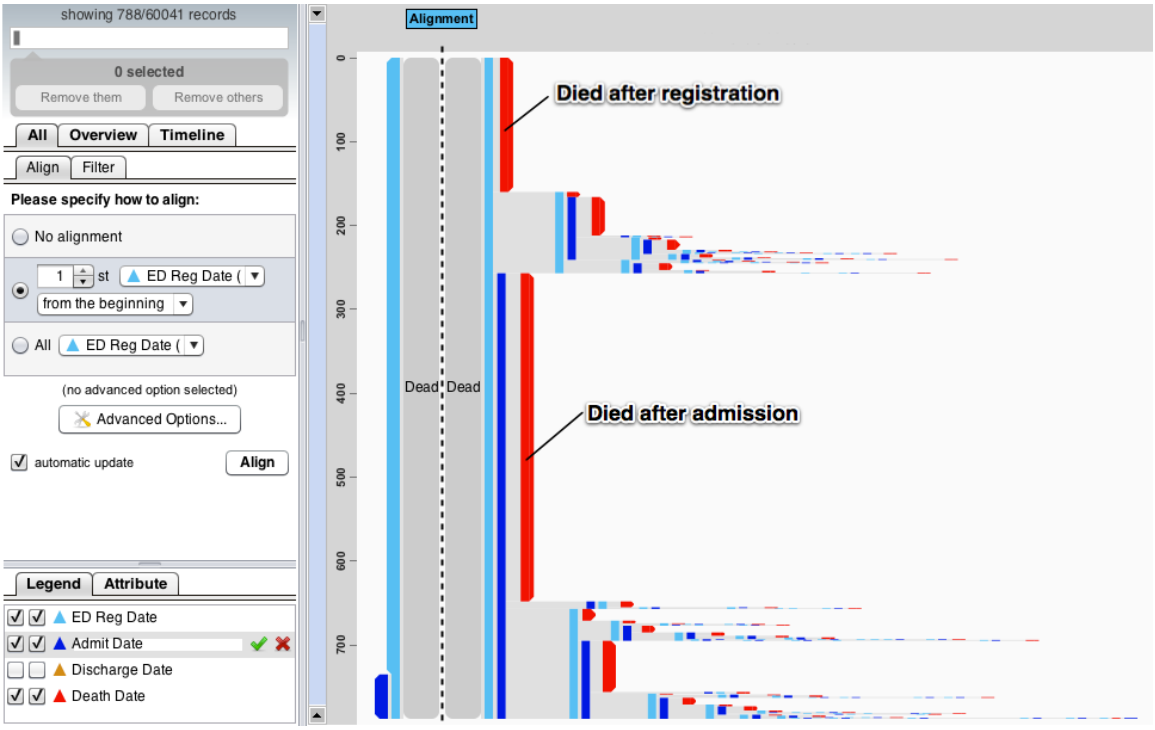

Figure 6.13: The majority of the patients died (*red*) after admission (*blue*) more than those who died while in the ED (*light blue*).

reliable and the timestamps are accurate, we could look at patients who died within 30 minutes of showing up in the hospital, for example, patients who came in with cardiac arrest. There were also some patients who came in alive, but something happened to them while they were in the ED and they died. These are usually rare cases. If we have accurate timestamps, we could look at the time distribution. However, there are many problems with these timestamps. Some patients were reported dead without registration. Some were admitted but the timestamps are not in the correct order.

6.5.6.5 Revived patients and John Doe

We also noticed that some patients were registered after they died. Is that because they came in dead and the registration did not get to them, or did they really come in alive but the timestamps were not consistent? In Dr. Hettinger's opinion, these were the patients who came into the ED and died.

There were some patients who came into the ED and died. Some came, were admitted and died. Then there were some patients with only one timestamp, which means that they died with no prior registration. These cases are called "John Doe". They probably were found dead outside.

6.5.6.6 Identify an interesting pattern and then search for it

Using LifeFlow, Dr. Hettinger was able to drill down to find interesting cases. One patient visited the ED, was discharged, then came back a few months later, was admitted and discharged from the hospital, and shortly thereafter came back and died. He was diagnosed with kidney failure, dehydration, an undefined kidney problem and kidney disease. This case represents very sick patients that were discharged, came back and then died in the ED. The last diagnoses for this patient were kidney disease and cardiac arrest. Another patient came in with a headache, was discharged, came back four months later, then had cardiac arrest and died. Another patient was diagnosed with adema, shortness of breath and high cholesterol. After this person was discharged, he/she came back four months later and died. "See, this pattern right here. It would be awesome if you can reproduce the Similan search,"

said Dr. Hettinger. So, we switched to the search tab to specify query "`ED Reg date` → `Discharge date` → `ED Reg date` → `Death date`".

At first, no record was returned as an exact match results (Figure 6.14 left). However, records in other results look like the pattern that we were looking for. This caused some frustration, so we removed `Discharge date` from the query and submitted the query again. This time we received the expected results (Figure 6.14 right). The results were blank at first because the `ED Reg date` and `Discharge date` in each record were exactly at the same time. This indicated that an additional data cleaning would be required to correct these two dates. We found the pattern that we were looking for in the current data and Dr. Hettinger was very happy with it. "These are exactly the patients that would be interesting to look at," he said. "Another thing that you can do is that... From the risk management standpoint, these are cases that are certainly concerning. I think most deaths ended up getting reviewed in the ED, but I don't know if they always go back and look and see what was their last visits? They might have missed something, but certainly it could be interesting learning points as well to pull the cases and use it to teach people, so that if there's something worthwhile to try to figure out to make it worthwhile to bring it to people's attention. That's great. That's excellent that you are able to do that."

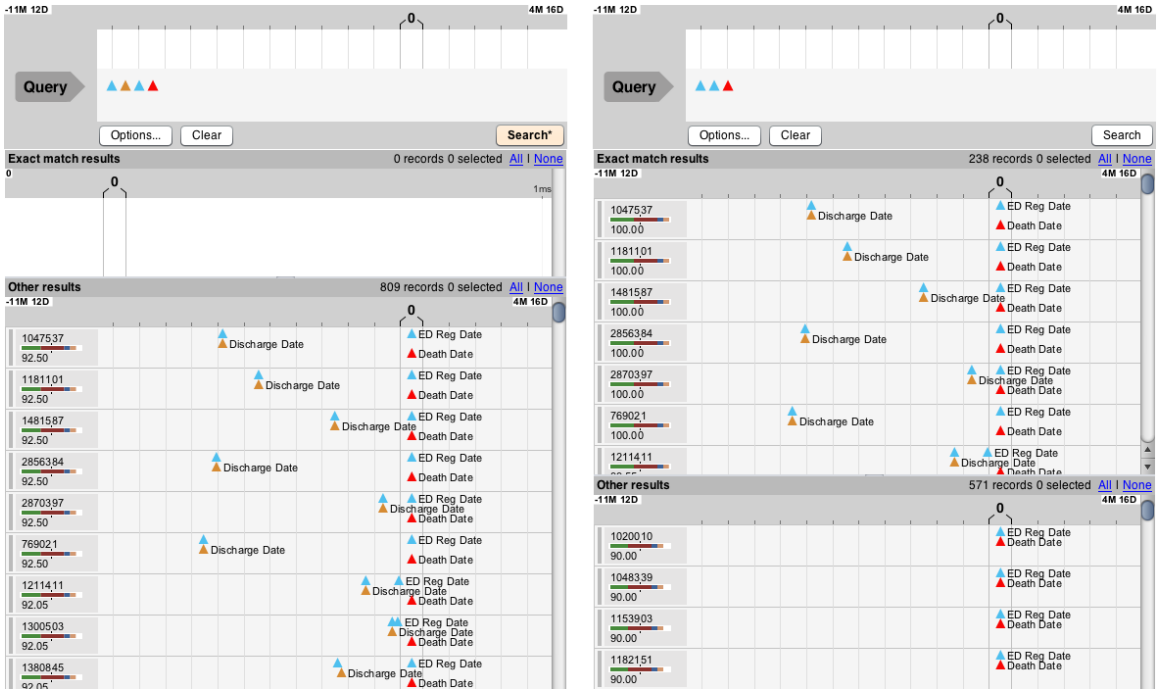

Figure 6.14: (left) Search for patients who visited, were discharged, came back and died. No record was returned as an exact match results. However, records in other results look like the pattern. This was because the `ED Reg date` and `Discharge date` in each record were exactly at the same time. (right) Refine the query to search for patients who visited, came back and died. 237 patients were identified.

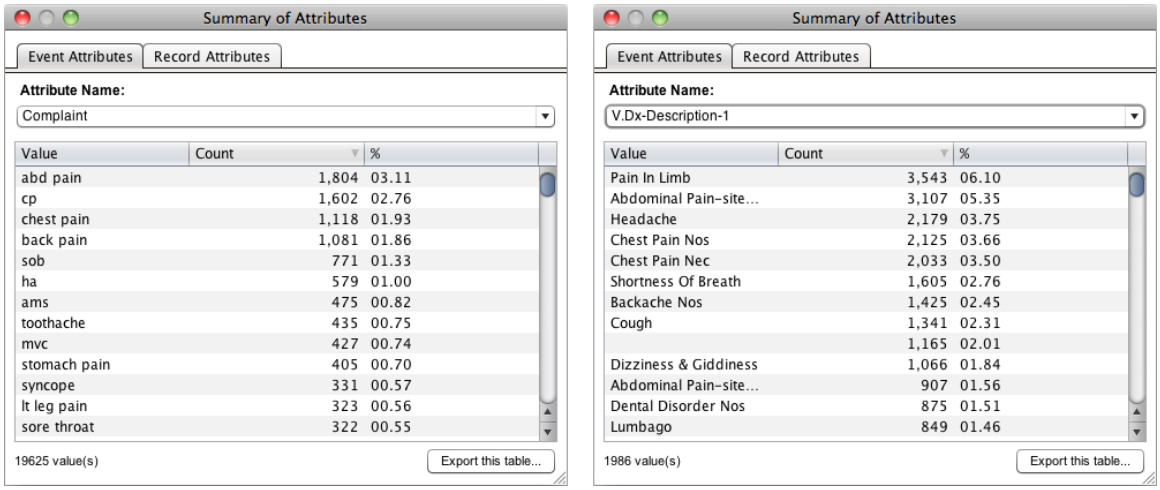

Figure 6.15: Summary of Complaints (left) and Diagnoses (right) of the Patients' First Visits

6.5.6.7 Diagnoses

Another thing we spent time looking at was the diagnosis. To see the common complaints from the first visits, we right-clicked on the first `ED Reg date` bar, selected "Show attributes summary" and selected the attribute "complaints" (Figure 6.15 left). In the table that came up, we saw various kinds of pain: abdominal pain, chest pain, back pain, etc. "This is fantastic. This kind of summary," said Dr. Hettinger. These complaints are from the patients. The triage nurses asked what bring you in today and they answered something. The nurses then had to decide what to put there. I noticed some patients with "end-stage back pain", so I innocently asked if they died after that. "It could be," replied Dr. Hettinger. "Or they're just being sarcastic."

Then we changed the attribute to "V.Dx-Description-1" (Figure 6.15 right), which were the ICD-9 codes that the physicians wrote. However, these descriptions are very specific. For example, *Lumbago* is a type of back pain. *Backache Nos* stands

for “Backache (not otherwise specified)”. These codes were also spread into multiple columns (V.Dx-Description-1, V.Dx-Description-2, ...), so it was not possible to show accurate counts in here, but at least it gave a sense of what the common diagnoses were. More preprocessing will be required to summarize these ICD-9 codes.

6.5.6.8 Frequent visitors

This time we used LifeFlow to drill down to patients with frequent visits (more than 40 visits in one year).

There was one patient who was not admitted in the early visits but after many visits he/she was eventually admitted. We checked the complaints and diagnoses for each visit and found that the complaints and diagnoses in the early visits were quite random. However, the first time he/she was admitted, the diagnosis was “overdosed pain medication”. The second time was also “overdosed pain medication”. This patient was admitted and discharged. The third time was “suicidal ideation”. Again, the patient was admitted and discharged. The fourth time was another “suicidal ideation”. This time, the patient was admitted to the psychiatric department.

In the bigger picture, we examined the LifeLines2 view on the right side and characterized two types of patients (Figure 6.16).

1. *Patients with clusters of frequent visits*: Frequent visits followed by a long gap then another series of frequent visits. These patients may have some recurring disease, such as asthma or a mental disorder. The gaps represented when they

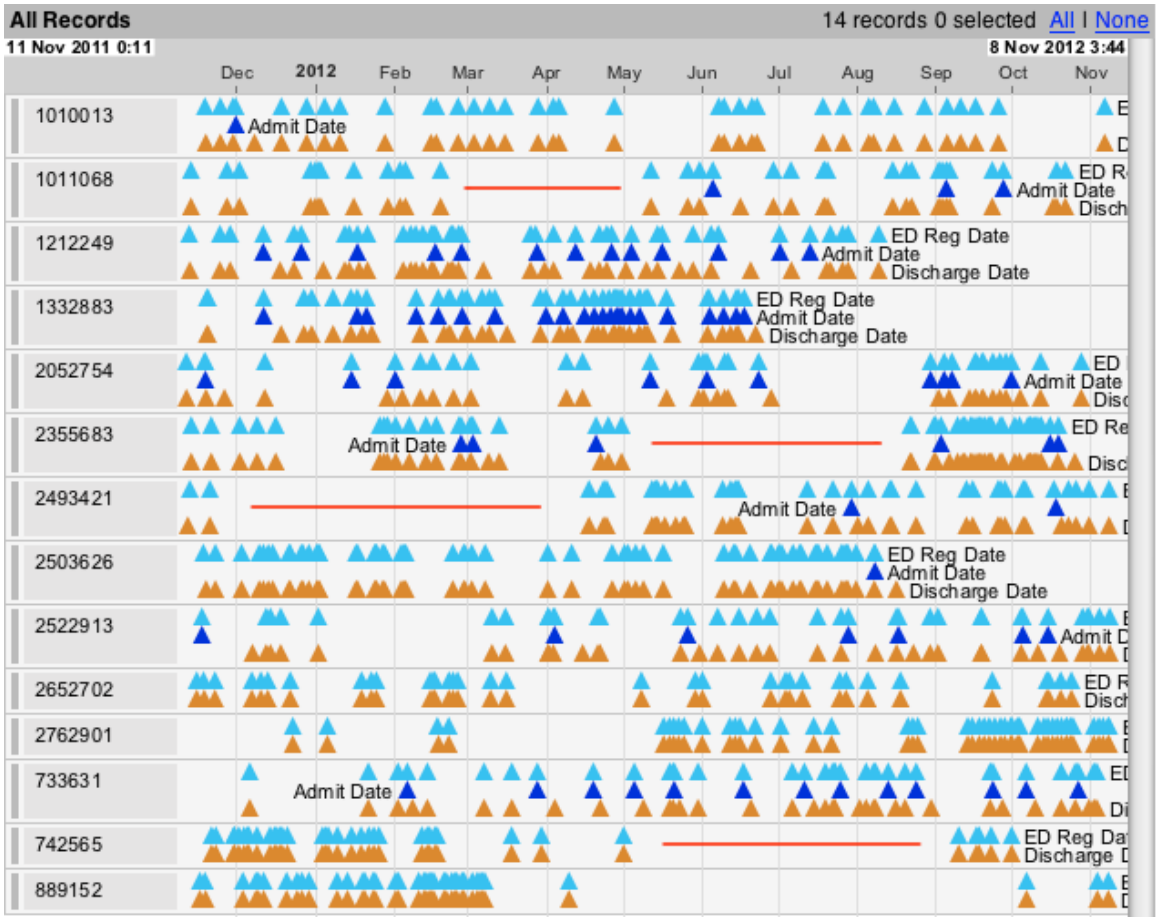

Figure 6.16: Patients with more than 30 visits: (a) Patients with many visits distributed regularly throughout the year, for example, 1212249 and 1332883 (b) Patients with frequent visits followed by a long gap (marked by red horizontal lines) then another series of frequent visits, for example, 1011068 and 2355683

got better. Another possible explanation is that they went to another hospital.

2. *Patients with regular visits*: Many visits distributed regularly throughout the year.

In Dr. Hettinger's opinion, these two types of patients are like the Anscombe's Quartet [12] of medical records. They have similar numbers of visits, but are very different. The difference may be left unnoticed when looking at the statistics but can be easily detected with visualization.

6.5.7 Conclusions and Discussion

This is the longest case study that I have conducted. Dr. Hettinger had participated since the very beginning of the development and been involved in many design iterations. His medical scenarios inspired many parts of the visualization and user interfaces.

The work in this dissertation was found useful for many hospital readmissions use cases. Many of which are understanding the overview, for example, checking admission rate, revisit rate, mortality rate, etc., for which LifeFlow can provide much more information than the standard canned report. Using zoom and filters, users can drill down to particular patients and see their details on demand, for example, examining patients with more than 40 visits. Users may identify an interesting pattern and then start searching for other patients with similar patterns. This case study demonstrates a wide range of exploratory activities that the users can perform on event sequences. At the end of the study, Dr. Hettinger shared his

thoughts about LifeFlow in his own words:

> "I have enjoyed working with LifeFlow (as well as you and your team). I have not used LifeFlow directly in any clinical applications, but am still exploring the best way to use it. However, we have seen interesting patterns within LifeFlow, including patterns of use among frequent visitors to the ED that may serve as targets for directed intervention to help patient's establish care with a primary care doctor and care for chronic conditions. We also looked at patterns of return visits to the ED and specifically focused on patient's that died. These cases could later become important cases for education as well as risk management concern that might not have otherwise been caught if they did not get visualized through the standard processes that usually screen for patients that return within 72 hours. So I believe it will be very useful in the future. I have spoken with various people at conferences and other meetings who are interested in the application of LifeFlow with their datasets and I believe it will provide important insight into many different settings.
>
> I believe there will be two major barriers to the adoption of LifeFlow for looking at datasets for the average user. One is getting access to the data set and formatting it in the necessary manner. Data Kitchen in an excellent tool, and with the use of recipes can automate the task, but certainly requires a fair amount of trial and error and familiarity to be used efficiently. Secondly, the controls and features of LifeFlow provide

a large amount of customization of the data, but again require training and familiarity to be used well. Even after not using the software for a few weeks I have to re-orient myself to the application and frequently "rediscover" features or where things are located. Even today I found myself looking for a zoom out feature, to have you inform me that the feature was already in place by double clicking the zoom bars. Obvious once you told me, but not apparent from the UI. Not sure the answer to the problem, but future versions could attempt to make the most common features more accessible to the novice, with access to the variety of controls as necessary. This is in no way to diminish what you have been able to accomplish, and it has been a pleasure to work with you on the project."

Working on this case study makes me realize the importance of facilitating the preprocessing. By making it easier to preprocess the data, users have less hesitation and overhead to adopt the approach. We also often find data cleaning problems during the analysis and need to go back to the preprocessing step. In addition, many design issues were identified, such as freezing the tooltip [2] or interpreting the alignment [3]. Some issues were easy to address while some issues presented more challenging problems that could lead to future research, such as including attributes in the similarity search. Dr. Hettinger's "rediscover" experience demonstrated that

[2]When the tooltip is very long, it shows a scrollbar, but users cannot move cursor to the scrollbar because the tooltip will disappear. First, I added a locking mechanism. By pressing F2, users can lock the tooltip and move cursor to the scrollbar. However, this was found awkward and unintuitive. After receiving the complaints from Dr. Hettinger, I removed the lock and let users use arrow keys to control the scrollbar of the tooltip.

[3]Users often forget that sequences on the left and right of the alignment are not connected.

MILCs are also beneficial in testing intuitiveness of the interface repeatedly. He could remember how to use the main features correctly, but sometimes forgot about less common features, which could be due to the long duration of this study, issue in the design, or a combination of both. A challenge due to the length of the study (≈1 year) is that users might have difficulty keeping track of added and removed features. Some minor changes could be easily forgotten because users were not using it everyday and there were many changes happened along the way. In terms of design, some features were found to be "not apparent from the UI" and need to be more obvious in the future.

6.6 How do people read children books online?

6.6.1 Introduction

The International Children's Digital Library (ICDL) (Figure 6.17) is an online library that contains 4627 children's books in 61 different languages (as of November, 2011) with over three million unique visitors since its launch in November, 2002. Its users come from 228 different countries. One very important question that Anne Rose, the website administrator, wanted to know is how do people read children's books from the ICDL website?

This encouraged us to look at the Apache web logs from the ICDL website to analyze how people read children's books online.

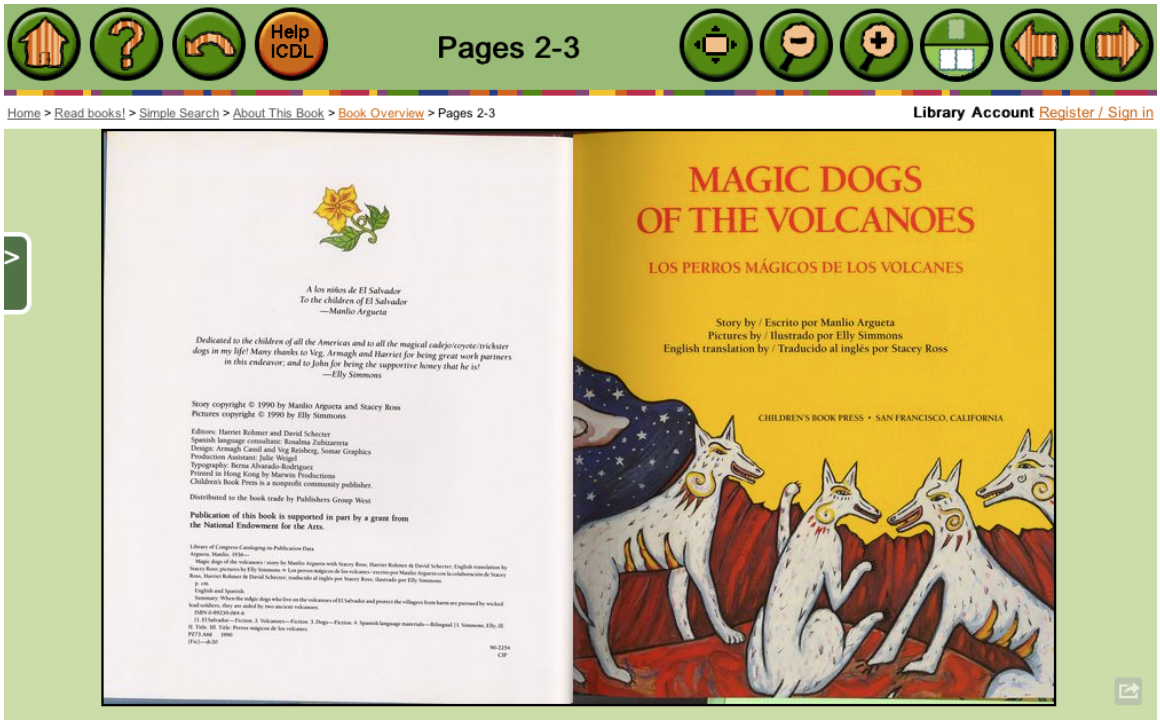

Figure 6.17: International Children's Digital Library www.childrenslibrary.org

6.6.2 Procedure

I requested access to the server and retrieved apache web logs for ICDL website. In the early meetings, I learned about the information contained in the web logs and things we hoped to learn from this dataset. After data conversion, I performed analyses and requested for meetings when I had questions or noticed interesting patterns in the data. The total duration of this study was six weeks.

6.6.3 Data

This sample dataset was taken from the period July 01–07, 2011. The original size of the dataset was about 1GB. We filtered the log entries to select only http requests to access pages of books: `http://www.childrenslibrary.org/icdl/BookPage`.

Each request contains a query string behind the url, such as `bookid=amrdima_00310002 &pnum1=6 &pnum2=7 &twoPage=true &lang=English &ilang=English` We parsed the page number and book id from this query string then group the http requests by IP address and book id into records. Each record represents how each IP address (user) read one book, or a *book session*.

IP address + book id	Page	Time
133.37.60.191_radjese_00380046	2	2011-06-30 05:01:06
133.37.60.191_radjese_00380046	1	2011-06-30 05:01:14
133.37.60.191_radjese_00380046	2	2011-06-30 05:01:20
133.37.60.191_radjese_00380046	4	2011-06-30 05:01:25
133.37.60.191_radjese_00380046	6	2011-06-30 05:01:27

Only the first 30 pages are included in the dataset. The processed dataset has 7,022 records and 57,709 events in total.

6.6.4 Analysis

6.6.4.1 Setup

We encoded the page number using color gradient from *blue* to *red*. Because people can also read the books in two modes: single-page (1,2,3, ...) and double-page (1,2,4,6, ...) [4], we hid the odd pages.

[4] Page 1 is the cover.

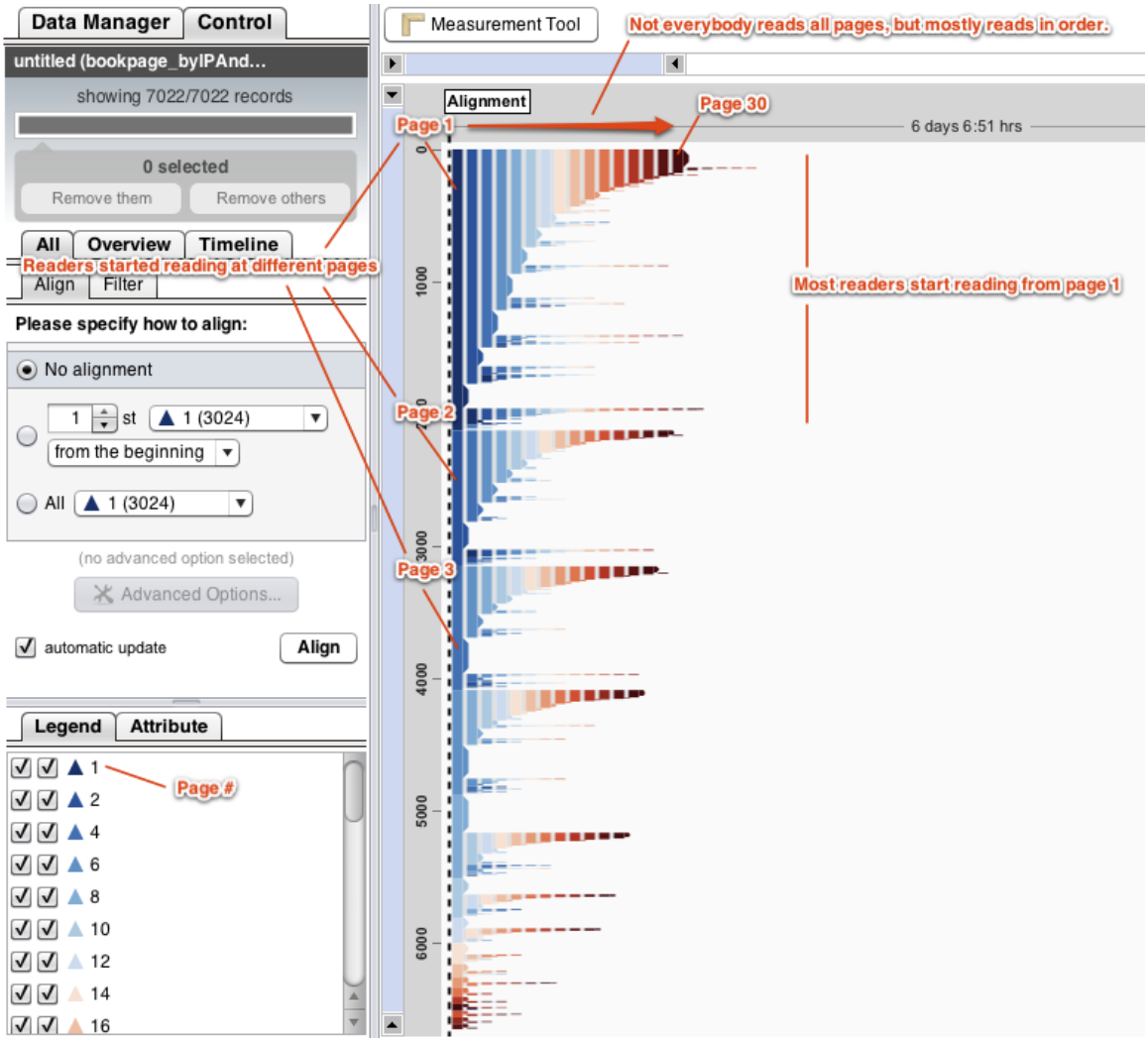

Figure 6.18: The page numbers are color-coded using color gradient from *blue* to *red*. We found that people started their reading on different pages. Some started from the first page, while others jumped into the later pages, probably skipping the empty pages in the beginning. The height of the bars shows that people started on the earlier pages more than the later pages.

6.6.4.2 First observation

The visualization in Figure 6.18 shows that people started their reading on different pages. Some started from the first page but some jumped into the later pages, probably skipping the empty pages in the beginning. The height of the bars shows that people started on the earlier pages more than the later pages. The color bars show a smooth gradient meaning that people mostly read in order. (Remember that the color coding is a gradient from blue to red.)

6.6.4.3 How do people read from the second page?

We aligned by the second page to see the reading pattern after visiting the second page (Figure 6.19). The decreasing height of the bars, as time went by, shows that people gradually dropped out while reading. The little long lines that split from the main trend in each step tell us that some readers flipped back to the previous page before they continued reading. The color changed becoming a slightly darker color (previous page) then changed back to the brighter colors (continue reading). We also noticed some long patterns before the second page.

6.6.4.4 Reading backwards

We zoomed into the mysterious sequence from the previous screen and saw people reading backwards. (The color changes from *red* to *blue* instead of *blue* to *red*.) We were curious so we selected them to see more detail (Figure 6.20).

In the detail view, the visualization shows one reader started reading from

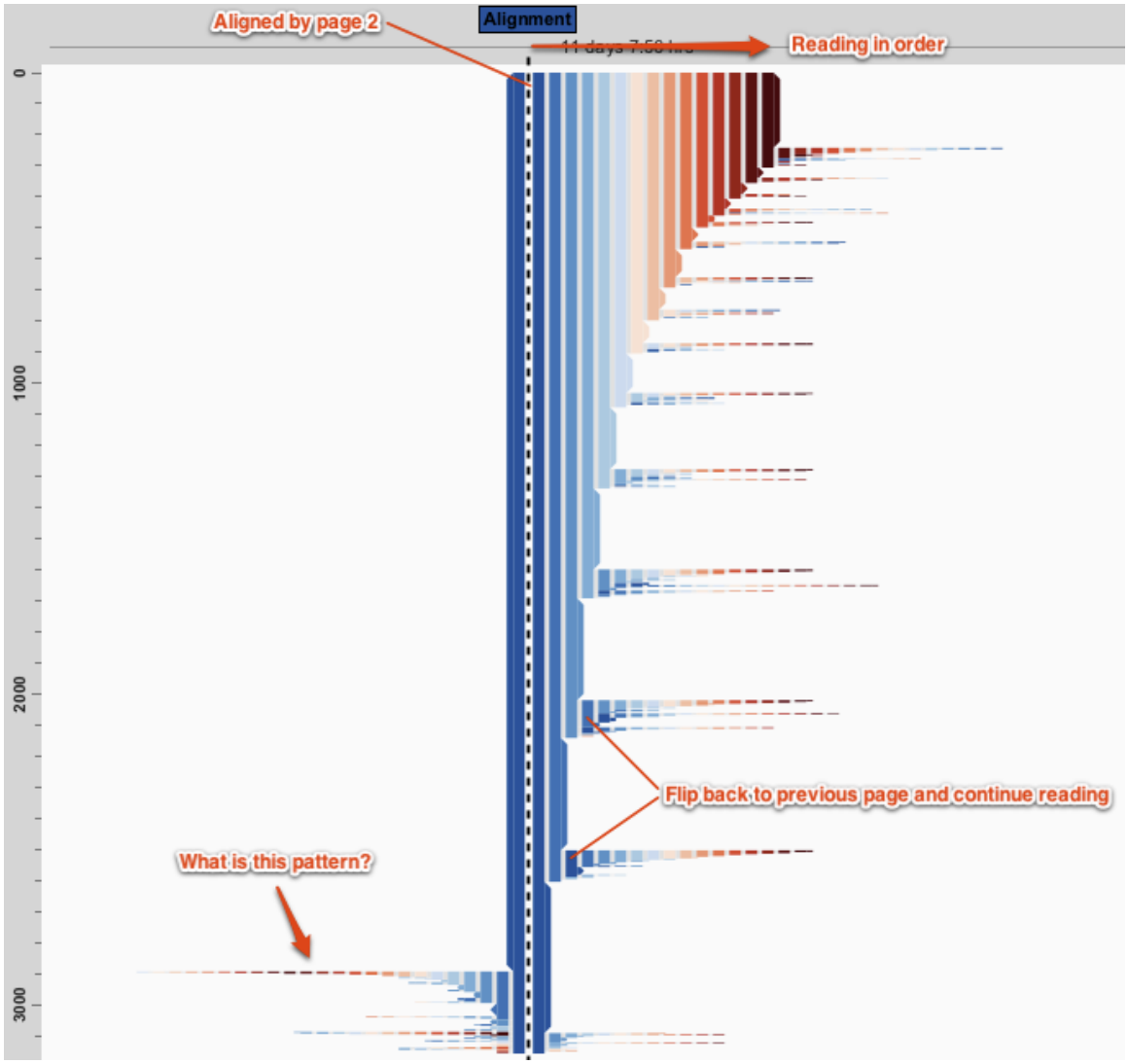

Figure 6.19: After aligned by the second page: People read in order from (*blue* to *red*). Some flipped back one page and continued reading (small lines). There are also some long patterns before the second page.

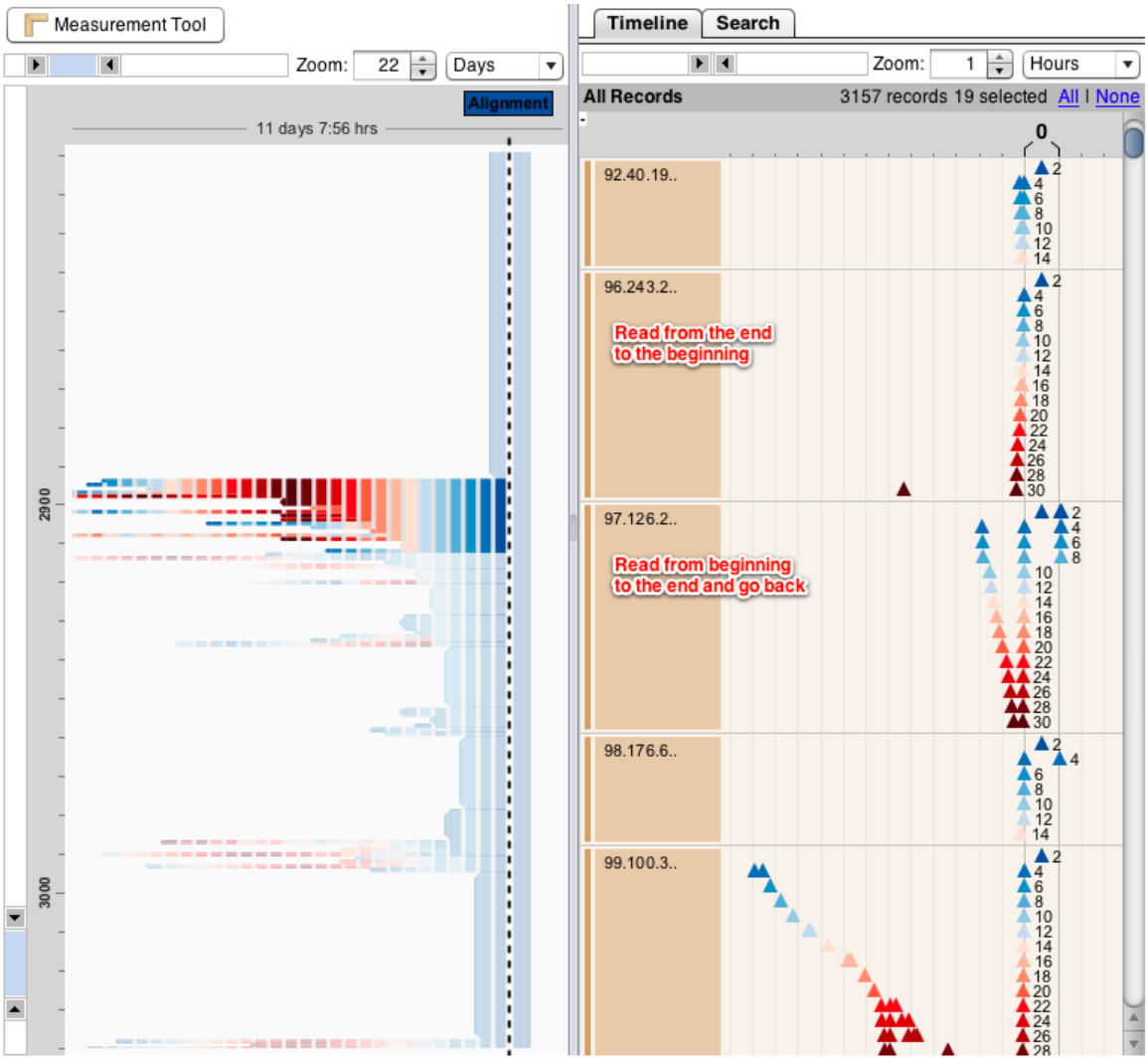

Figure 6.20: The selection shows book sessions in which readers accessed the pages in the backward direction.

page 30 and flipped back all the way to page 2. Another reader read from page 4 to the end and flipped back all the way. And many more cases.

We checked the attributes of these books. They are in English and Spanish (left-to-right languages). So there should not be any confusion about reading direction.

To investigate further, we used the measurement tool to measure the time from page 2 to page 30 when users accessed the pages in the forward direction. The mean and median times were 25:02 and 4:45 minutes, respectively.

We also measured the time from page 30 to page 2 when users accessed the pages in backward direction. The mean and median time were 33 and 24 seconds, respectively, which is approximately one second per page. It appears as though they were not reading backwards, just flipping the pages.

6.6.5 Conclusions and Discussion

We dug deeper into several records with backward access one-by-one and hypothesized a few possible scenarios.

1. *Finished reading a part of the book*: For example, one reader read from page 10 to 18, which is exactly one chapter of the book, then flipped back to page 1 and did not have any interaction after that.

2. *Shelf books*: ICDL has a membership system, which a member can store books on his/her virtual shelf for reading later. The system also remembers the last page read by the member. However, we assumed that some members might

open the book and find it at page 30, so they tried to flip it back to the first page and start reading again from the beginning.

3. *Parent preview*: Some users read a few pages then flipped back to the beginning and started reading from the beginning. We assumed that this is a behavior of parents selecting books for their children. They flipped a few pages to check that it is suitable then went back to the beginning to read it.

From these three possible scenarios, the users seemed to depend heavily on the page-by-page (previous/next button) navigation. To change from page 30 back to page 1, they went to the previous page 30 times instead of using other controls that could let them jump to page 1 faster. This could be because the users could use their arrow keys to go to previous/next page, which was convenient for them, so they kept pressing the arrow keys, or the users did not know how to jump back to the first page directly.

However, finding the real explanation of why many people access the online children's books backwards or whether the navigation needs to be improved, will require a further user study, which is beyond the scope of this case study. Therefore, we would like to take a step back and summarize what we learned from the ICDL web logs so far using LifeFlow.

LifeFlow shows potential for helping the ICDL administrator understand how people are accessing the online books. It highlights the users' reading behaviors from the majority that read in order, the readers that skipped the blank early pages going straight to the content, the readers that flipped back and continued reading, to the

readers that flipped backwards all the way back to the first page. By understanding these behaviors better, the administrator can improve the website to suit the readers better.

6.7 Studying User Activities in Adaptive Exploratory Search Systems

6.7.1 Introduction

Adaptive exploratory search systems (ESS) can provide efficient user-centered personalized search results by incorporating interactive user interfaces. The core component of the adaptive systems is called a user model, where the users search tasks, contexts, and interests are stored, so that the systems can adapt to those specific tasks or contexts. In conventional adaptive search approaches, the user models are hidden from the users. They are black boxes and the users cannot estimate easily what is going on inside. Therefore, they cannot expect what personalized results will be returned from the black box and have very limited capability to correct the systems unexpected erroneous behaviors. Several approaches tried to solve this problem by transparently revealing the user model contents and providing the ability to directly control the user models.

Dr. Jae-wook Ahn, post-doc researcher at the HCIL, developed two adaptive ESS called *TaskSieve* [2] (Figure 6.21) and *Adaptive VIBE* [1] (Figure 6.22) as parts of his PhD dissertation work. TaskSieve uses typical ranked lists but it has a component for adjusting the importance of user model and user queries. Users can

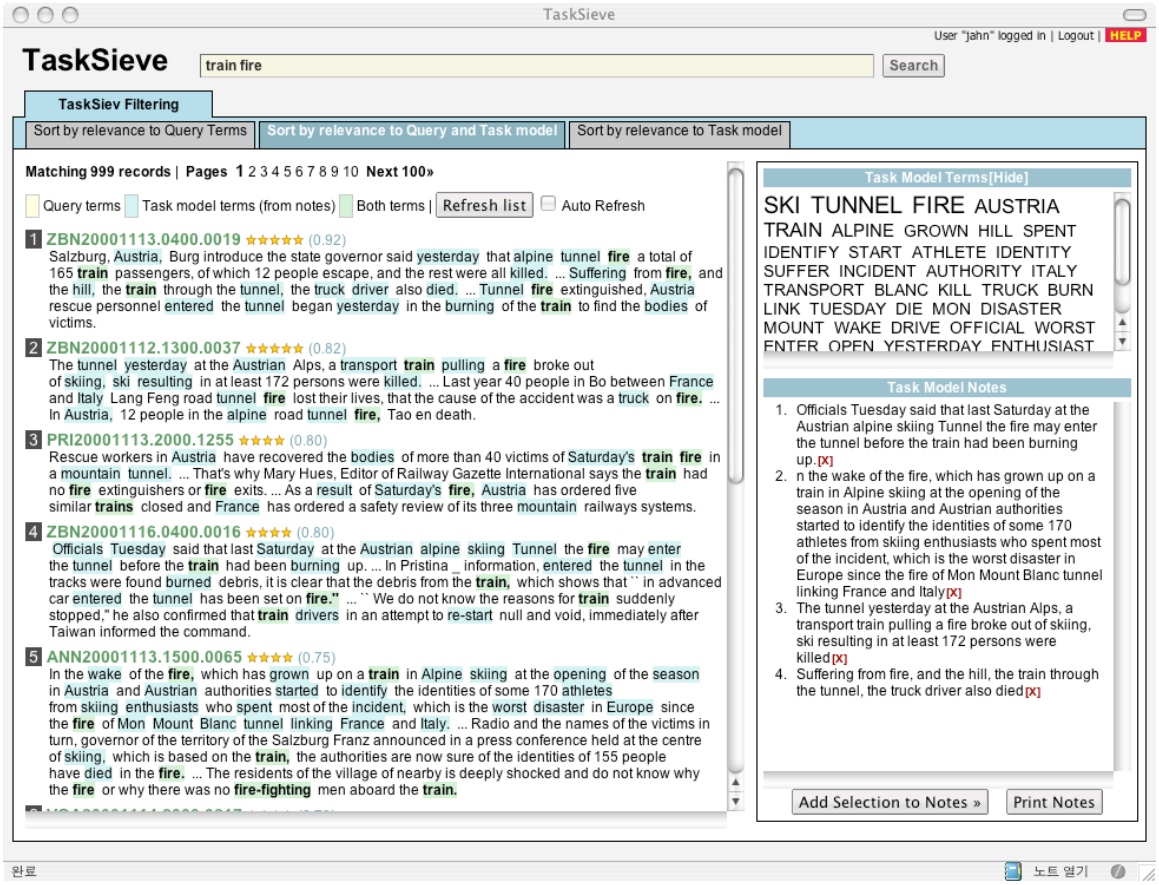

Figure 6.21: TaskSieve

explore different search results by switching three sets of weight configurations of user models versus user queries (1:0, 0.5:0.5, and 0:1) while monitoring their user model contents. Adaptive VIBE extends the open user model using a reference point based interactive visualization called VIBE (Visual Information Browsing Environment). It defines the user query and the user model (as the list of keywords) as reference points (Point of Interest or POIs) and interactively places the search result documents according to their similarities to each POI. Documents are placed closer to more similar POIs and users drag the POIs in order to visually explore the search space. Both systems have log facilities that record user activities. These

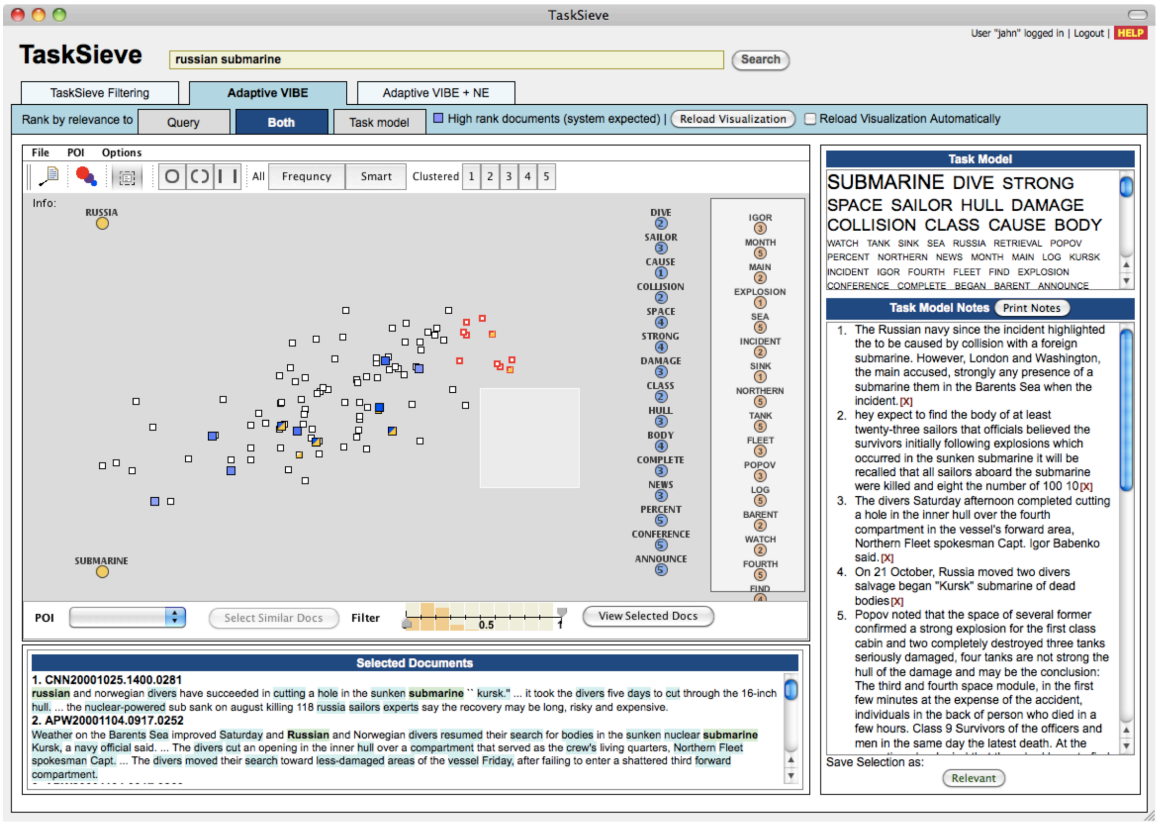

Figure 6.22: Adaptive VIBE

actions are organized by higher-level categories in Table 6.2 and 6.3. TaskSieve has six categories of actions while Adaptive VIBE has the same six categories and two additional.

Dr. Ahn was interested in analyzing user activity sequences from the log files of the two systems. The log data was extracted from a user study including 33 users in 60 sessions (20 minutes per session). For each session, the systems were loaded with news articles from the Topic Detection and Tracking (TDT4) test collection and the participants of the user study were asked to finish search tasks. Our goal in this study was to discover two user behavior patterns from the log data.

1. *Switch of activities*: Typical look up search systems will have simple activity

Category	Action
Login/out	login, logout
Search	search
Overview	search response, navigate page
Examine doc	open doc
Update UM (User Model)	save note, remove note
Manipulate UM	change UM & query weight

Table 6.2: TaskSieve User Actions

Category	Action
Login/out	login, logout
Search	search
Overview	reset visualization
Examine doc	view by mouse-over, select doc, open doc
Update UM (User Model)	save note, remove note
Manipulate UM	change UM and query weight, select UM POI, move UM POI
POI activities	reset POI, select POI, select query POI, move POI, move query POI
Find subset	select similar documents from a POI, marquee selection, range slider filtering, show similar documents or POIs, view selected documents, view auto-selected documents

Table 6.3: Adaptive VIBE User Actions

switch patterns such as query → examine list → examine documents. We expected to see different patterns from the open user model based ESS. Users were supported with more features to explore the search space and view/control their mental models. We were interested to discover how the users really exploited those features.

2. *User activity patterns*: It would be useful to have a list of user activity patterns regarding how users actually used the features while trying to solve specific sub-tasks. This knowledge will help improving the future ESS systems with open user models.

6.7.2 Procedure

Dr. Ahn and I met a few times in the first two months of this study. I demonstrated the software and explained the data format. After Dr. Ahn was familiar with the LifeFlow software, he converted his data and analyze them on his own. He contacted me when he needed technical support or found interesting patterns. In the end, we wrote a paper together to report the results [3]. The total duration of this study was five months.

6.7.3 Data

We generated datasets from activity logs of 60 user study sessions for the two analyses above:

1. List of all actions (per user session) for analyzing switch of activities: One

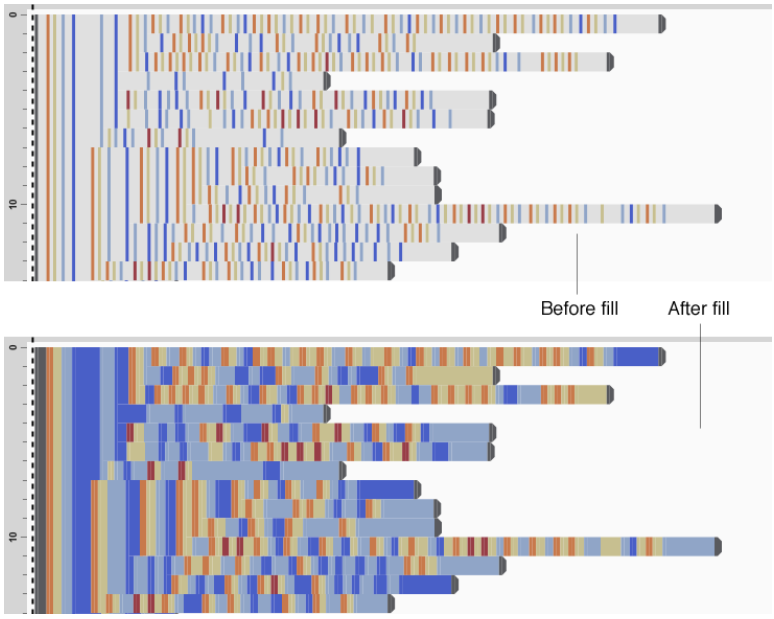

Figure 6.23: Before (above) and after (below) filling gaps with colors of previous events

dataset for TaskSieve which has 1,716 events and another dataset for AdaptiveVIBE which has 10,646 events.

2. Pairs (i.e. bigrams) of actions for analyzing user activity patterns: This dataset has 19,142 events.

6.7.4 Analysis

6.7.4.1 Switch of activities

We used the high-level action categories (Table 6.2 and 6.3) instead of individual actions in order to reduce the number of event types and visual clutters. Next, we merged repeated consecutive events into one event using a "merge repeated events" option in LifeFlow. However, the sequences are still long and visualizing the time gap continues to be difficult. So, we enabled another option in LifeFlow to

fill the gaps between events with colors of previous events, instead of light gray, to help us notice the long activities easier (Figure 6.23).

Events are colored as follows: `Login/out` is not important, so we colored it *gray* for less attention. `Search` is encoded by *orange*. `Overview` (show an overview of search results) is a follow-up event of `Search`, so it is encoded in *yellow* to show connection with `Search` (*orange*). `Examine doc` and `Update UM` are encoded in *pale blue* and *blue*. `Manipulate UM` is an interesting event type so it is encoded in *red* to make it stands out. Two additional event types from Adaptive VIBE, `POI activities` and `Find subset`, are encoded in *light green* and *green*, respectively.

Figure 6.24 compares the overviews of the search activity sequences of TaskSieve and Adaptive VIBE. Remember that the width of the colored blocks corresponds to the duration of the action.

The two most frequent activities that occupied the largest portion of the screens were `Examine Doc` (*pale blue*) and `Update UM` (*blue*). It had been expected because the users would spend a significant amount of time reading the snippets or fulltexts of the documents (`Examine Doc`) and edit their notebooks. In the two systems, a notebook editing action automatically leads to updating the user model (`Update UM`). What was more interesting was the density of the exploratory activities. In the visualization-based system (Adaptive VIBE), the distribution of different actions was denser than in the text-based system (TaskSieve). The participants switched more frequently by spending less time (smaller width colored blocks) per action, with the two additional event types (`POI activities` (*light green*) and `Find subset` (*green*)) mixed in the sequence. This observation reflects that they

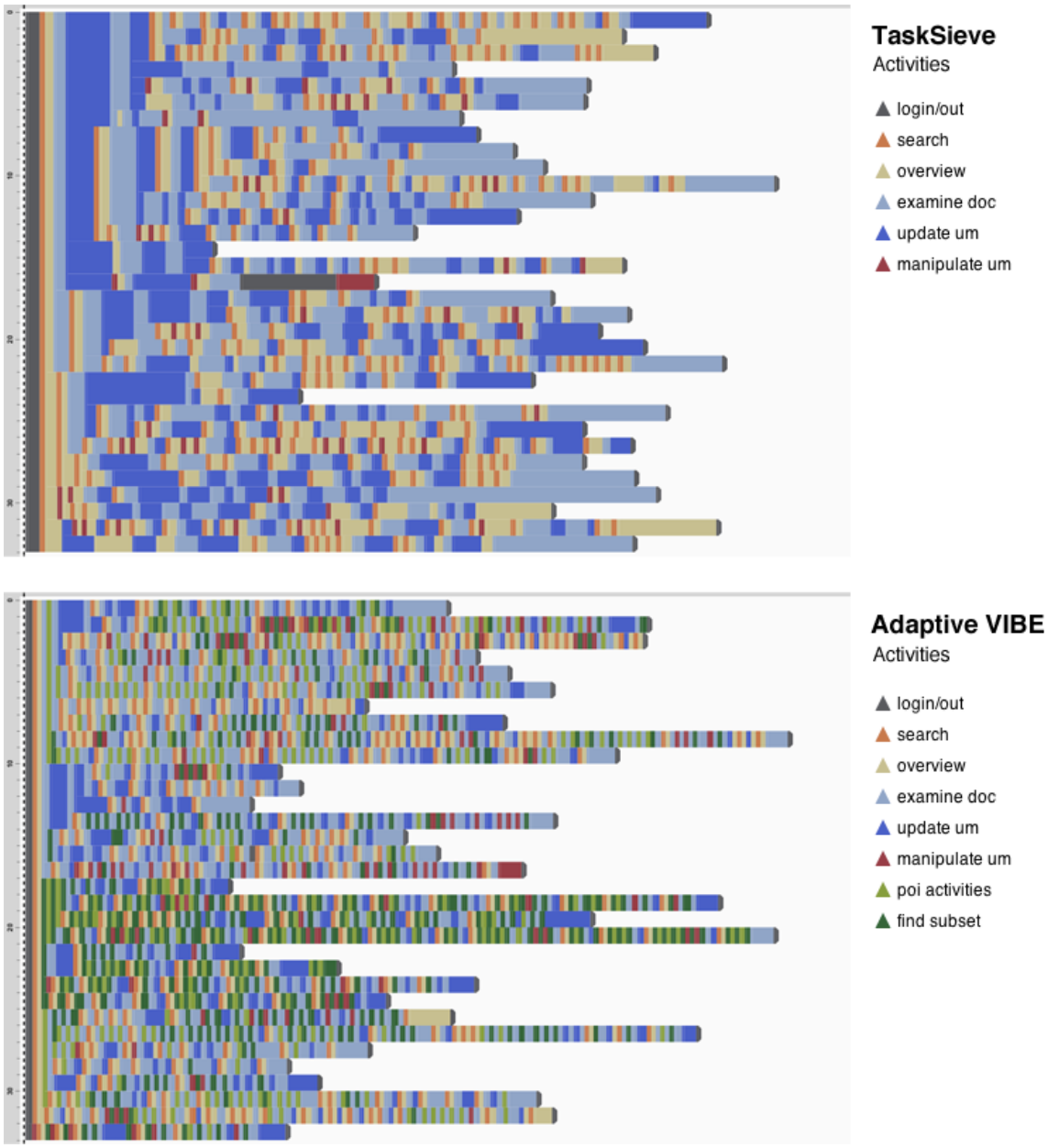

Figure 6.24: Overview of User Behaviors in TaskSieve (above) and Adaptive VIBE (below): In the visualization-based system (Adaptive VIBE), the distribution of different actions were denser than in the text-based system (TaskSieve). The participants switched more frequently by spending less time (smaller width colored blocks) per action, with the two additional event types (`POI activities` (*light green*) and `Find subset` (*green*)) mixed in the sequences.

fully exploited the richer feature set provided by Adaptive VIBE. At the same time, the high action switches included a lot of exploratory actions. We could assume that they were able to take advantage of the flexibility of the visualization system and performed more exploratory behaviors.

The second observation is the user model manipulation. Users could directly manipulate the user model weights or drag the user model keywords (POIs), in order to better reorganize the visualization of retrieved documents, learn about and control the effects of their user models. In Adaptive VIBE, these activities (*red*) were more frequent and prevalent, which suggests that the users more actively used the visual user model manipulation feature of the system.

6.7.4.2 User activity patterns Part I: Frequent patterns

The second part of the analysis was to find out the exact user behavior patterns. It will help to design more efficient adaptive exploratory search systems if we could learn how the users really used them. We limited this analysis to Adaptive VIBE only, because it provided more diverse adaptive exploration features than TaskSieve. Each record in this dataset is a bigram taken from a full sequence of user's activities in the previous section. For example, a sequence `Login/out` → `Search` → `Overview` → `Examine doc` becomes three bigrams: (1) `Login/out` → `Search`, (2) `Search` → `Overview` and (3) `Overview` → `Examine doc`.

When these bigrams are visualized in LifeFlow, it becomes a distribution of user actions of length one and two (Figure 6.25). Each bar's height represents

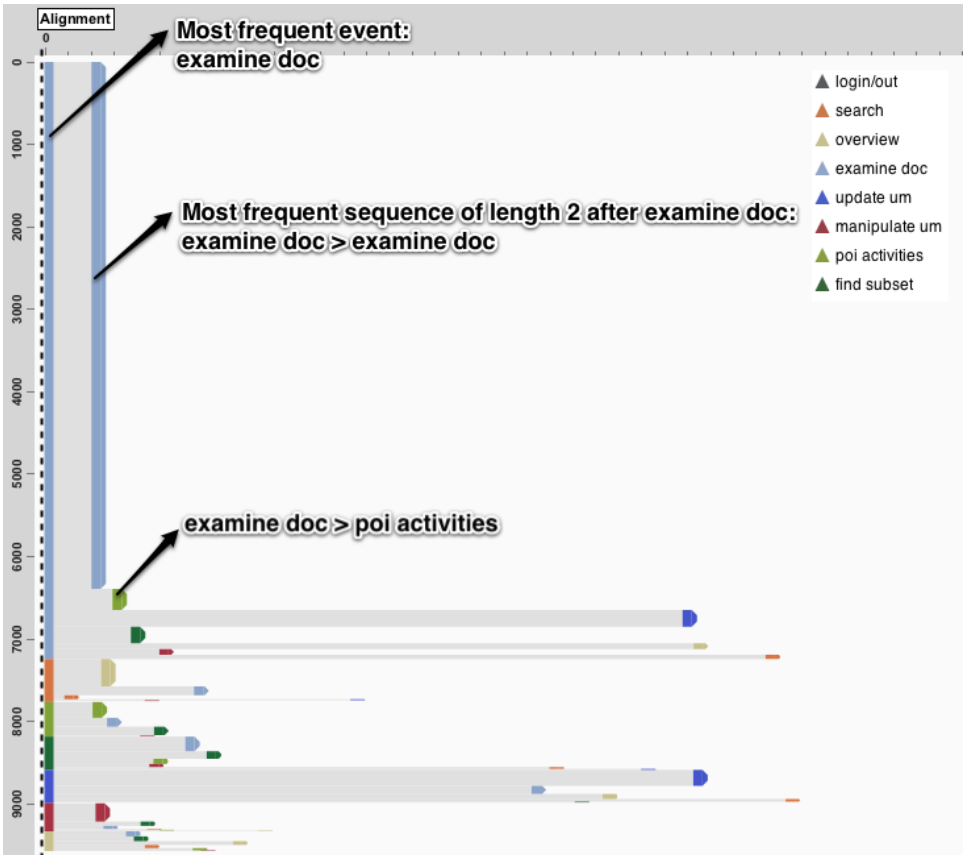

Figure 6.25: Bigrams of user activities visualized in LifeFlow

frequency of the action patterns and the horizontal gaps between bars represent the average time. In this analysis, we did not fill the gap with the colors of previous events. Reading the visualization from left to right, we could first find out the most frequent actions by looking at the tallest first level bars and then the following frequent actions. For example, the most frequent action is `Examine doc` (*pale blue*). For the actions that follow `Examine doc`, the most frequent action was `Examine doc` again while the second and the third were `POI Activities` and `Update UM`, which means that the users moved/selected POIs (not user model POIs in this case) or updated their user model contents by adding or removing texts to/from the notebook. This hierarchy of the LifeFlow tree could make the structured analysis of the bigrams easier than simple frequency counting, by following down the branches of dominant activities in the first level.

Table 6.4 summarizes the list of frequent bigrams discovered from this analysis. Overall, the frequencies of the first actions of the bigrams were almost equivalent except `Examine doc`. We thus included all event types in the first level and then counted the top three event types that follow. The findings from the table are as follows:

1. Many exploratory actions—examine the overview, control the visualization to change/re-interpret the big picture (including *user model update/manipulation*), zoom and find subset—were much more prevalent compared to simple look-up search actions.

2. There are many sequences of repeating events—`Examine doc` → `Examine`

Table 6.4: List of dominant user activity bigrams

1st activity	2nd activity	Frequency
Examine doc	Examine doc	6,392
Examine doc	POI activities	254
Examine doc	Update UM	204
Search	Overview	330
Search	Examine doc	108
Search	Search	48
POI activities	POI activities	192
POI activities	Examine doc	107
POI activities	Find subset	102
Find subset	Examine doc	178
Find subset	Find subset	93
Find subset	POI activities	62
Update UM	Update UM	193
Update UM	Examine doc	97
Update UM	Overview	62
Manipulate UM	**Manipulate UM**	**222**
Manipulate UM	**Find subset**	**53**
Manipulate UM	**Examine doc**	**37**
Overview	Examine doc	63
Overview	Find subset	58
Overview	Overview	46

`doc`, `POI activities` → `POI activities`, `Update UM` → `Update UM` and `Manipulate UM` → `Manipulate UM`.

3. User model exploration was used almost as frequently as other actions.

4. This table can be used as a list of possible user actions for similar adaptive ESS.

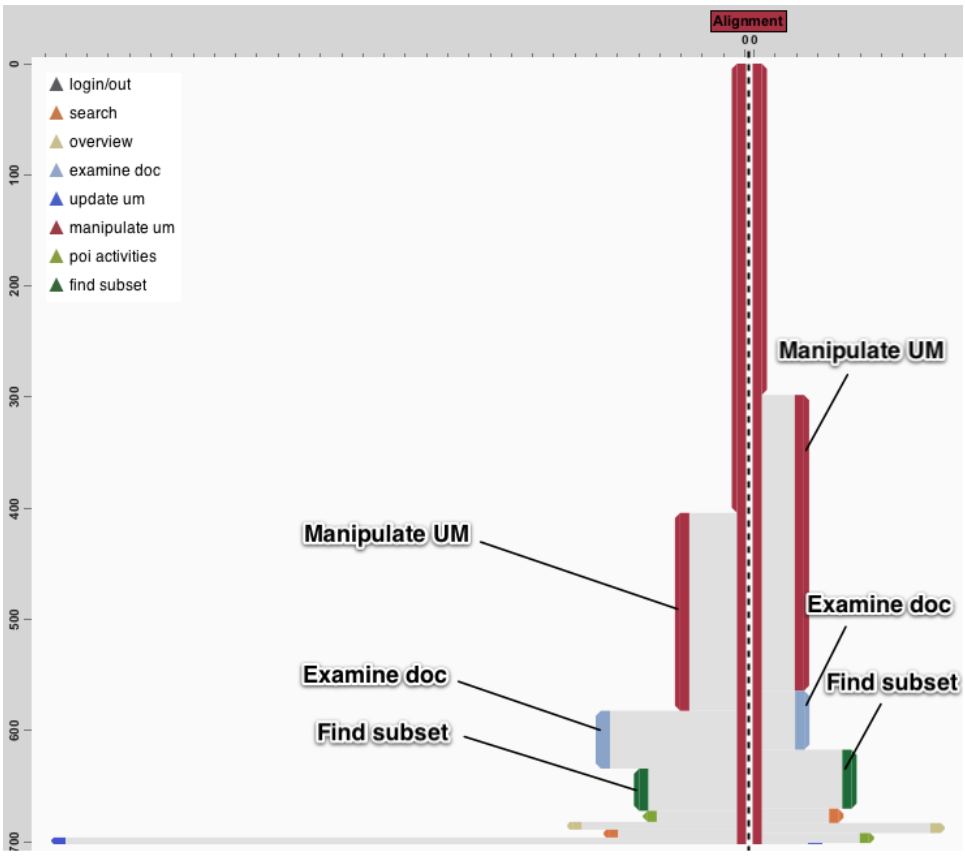

Figure 6.26: Bigrams of user activities after aligned by `Manipulate UM`: The three most frequent actions *before* `Manipulate UM` were still the most frequent actions *after* `Manipulate UM`. Seems like the user model manipulations were in the chain of the three actions repeatedly and the user model manipulation task needs to be considered as a set with those *friend* actions.

6.7.4.3 User activity patterns Part II: User model exploration

The next step was to look deeper into the user activities regarding how the users explored and controlled their open user models, which were the core module of the adaptive ESS. Adaptive VIBE implemented the visual user models as keyword POIs and allowed users to drag them around the screen, select them for supporting further actions (e.g. selecting documents similar to the selected POI), disable some of them temporarily, etc. From the analysis above, these user model manipulation activities were conducted as significantly as the others. The three most frequent actions *just after* `Manipulate UM` were `Manipulate UM` (*red*), `Examine Doc` (*pale blue*), and *Find subset* (*green*) (Table 6.4). This suggests that the user model manipulation actions were done repeatedly and the users could narrow down their search targets as results of user model exploration.

Therefore, we aligned the sequences by all `Manipulate UM`. In Figure 6.26, the right side after the vertical dashed line is the pairs where `Manipulate UM` was preceding the second action. The left side is vice versa. We could compare the left and right side and observe that the three most frequent actions *before* `Manipulate UM` were still the most frequent actions *after* `Manipulate UM`. It led us to conclude that the user model manipulations were in the chain of the three actions repeatedly and the user model manipulation task needs to be considered as a set with those *friend* actions. By looking at the overview again (Figure 6.24 below), we could confirm that occurrences of `Manipulate UM` (*red*), `Examine doc` (*pale blue*) and `Find subset` (*green*) are adjacent to each other.

6.7.5 Conclusions and Discussion

This case study provided a preliminary analysis of user behavior patterns in adaptive exploration search systems. We analyzed the log data of a user study that included various user actions using LifeFlow. From the analysis, we could find that the user exploration actions were switching more frequently using the visualization-based adaptive ESS system. At the same time, the user model exploration features were used as frequently as other features and we could find patterns regarding how the user model explorations were combined with other exploratory behaviors. We also generated a list of user action patterns of the adaptive ESS systems for supporting future system design of the same kind. This could be extended to include deeper analysis of other activities closely related to user model exploration, such as finding subset, query exploration, and user model updates. Finally, I would like to include Dr. Ahn's comment on LifeFlow in his own words:

> "LifeFlow is an efficient and powerful tool to analyze complex log data. I used to store my log data in a database and used SQL to analyze them. Even though I am familiar with formulating sophisticated SQL commands, they have innate limitations compared to visualizations such as LifeFlow. For example, I can get the activity switch statistics using SQL by calculating the average switch counts. However, they are mostly aggregated scores and it cannot provide the overview of the distribution of the activity switches, which LifeFlow can do successfully.
>
> At the same time, the activity switch analysis is prone to a criticism that

if the frequent switches only reflect the random and meaningless actions of the users (probably due to the frustration from a new tool). Using LifeFlow, we can check the visual length of actions (longer actions will represent a low chance of random switches) and the regular pattern of sequences in order to address such concerns.

For identifying the user-model related activity patterns and generating a meaningful list of those activities, I believe visual analysis using LifeFlow is one of the most efficient methods. In particular, it was very helpful to examine the actions before and after the interested actions. Using this feature, I was able to easily identify the frequent chain of activities that the users performed during the experiment."

6.8 Tracking Cement Trucks

6.8.1 Introduction

Freewill FX Co., Ltd., is a company based in Bangkok, Thailand that designs and produces wireless sensors and mobile tracking devices. Freewill FX's mission is to help people lead better lives and to help businesses become more effective by using mobile and wireless technologies. One of the products that the company offers is *TERMINUS: Advanced Fleet Management System*. Knowing just the locations of your fleet does not provide tangible business benefits. Thus, TERMINUS focuses more on adapting fleet management systems to business processes and optimizing fleet utilization to generate measurable business benefits.

After hearing about LifeFlow, Mr. Chanin Chanma, product specialist, and Mr. Sorawish Dhanapanichakul, product manager of TERMINUS, were interested in looking at one of the datasets that they had in LifeFlow. One of their clients had ordered devices for tracking cement trucks. The sensory data from these devices were automatically collected into an excel spreadsheet.

6.8.2 Procedure

Mr. Chanma and I and discussed about the possibility of using LifeFlow to analyze his data. Following the discussion, he sent me a spreadsheet that contains sample data. I then converted the data using DataKitchen and contacted him back with a few screenshots. We then arranged an online meeting via Skype with him and Mr. Dhanapanichakul to look at the data together. After the online meeting, I summarized findings from the meeting and requested for additional feedback via email. The time between each meeting was roughly one week and the total duration of this study was six weeks.

6.8.3 Data

The dataset consists of 821 trips and 8,091 events. Each trip consists of multiple events tracked from the beginning to the end of the trip: `Enter plant`, `Start load cement`, `End load cement`, `Leave plant`, `Arrive site`, `Start fill cement`, `End fill cement`, `Leave site` and `Back to plant`. IDs of the plant, site and vehicle were also documented for each trip.

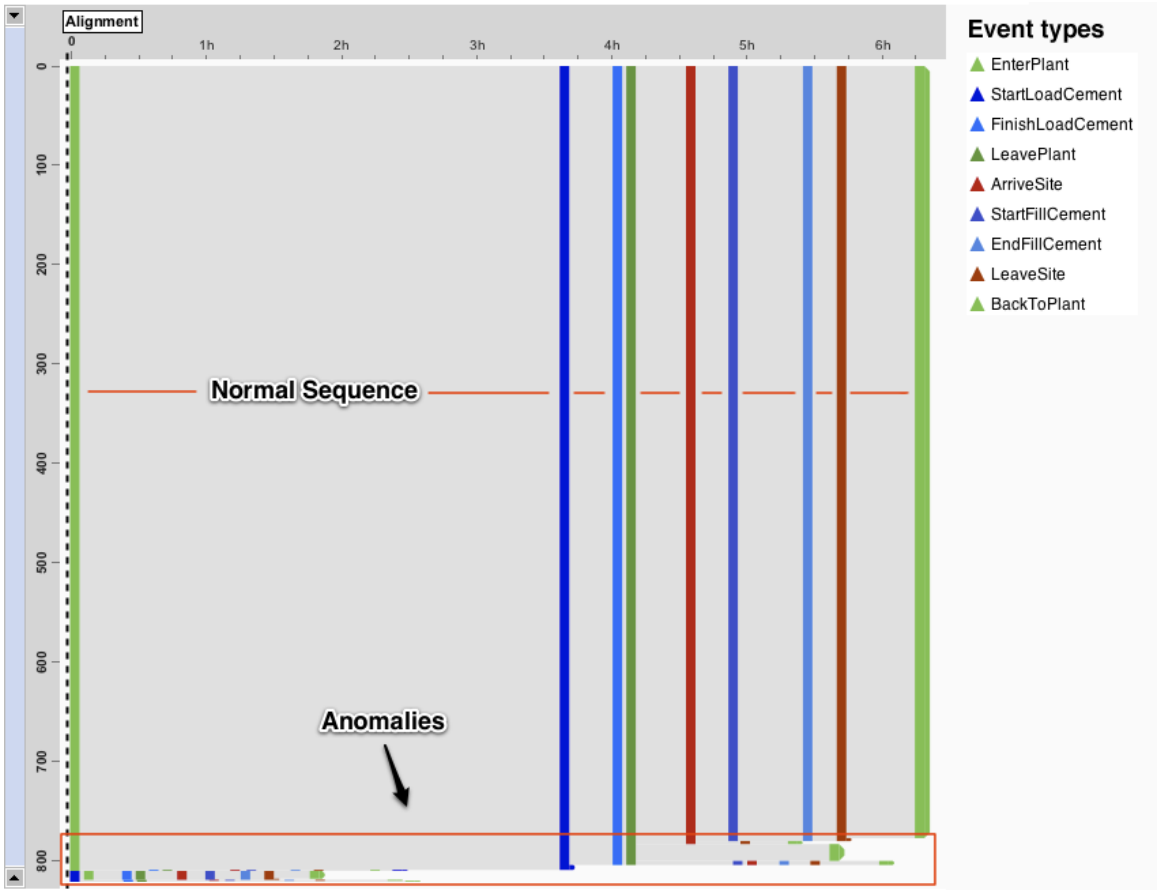

Figure 6.27: LifeFlow showing 821 trips: The majority of the trips had normal sequences. However, there were some anomalies in the bottom.

The raw data was in MS Excel Spreadsheet format. A preprocessing was required to translate the column headers into English. (They were in Thai.) After that, the spreadsheet was converted into LifeFlow format using DataKitchen.

6.8.4 Analysis

6.8.4.1 Overview

After loading the dataset with 821 trips, we could see that, most of the time, the trips occurred step-by-step as expected (Figure 6.27). Then we could see the

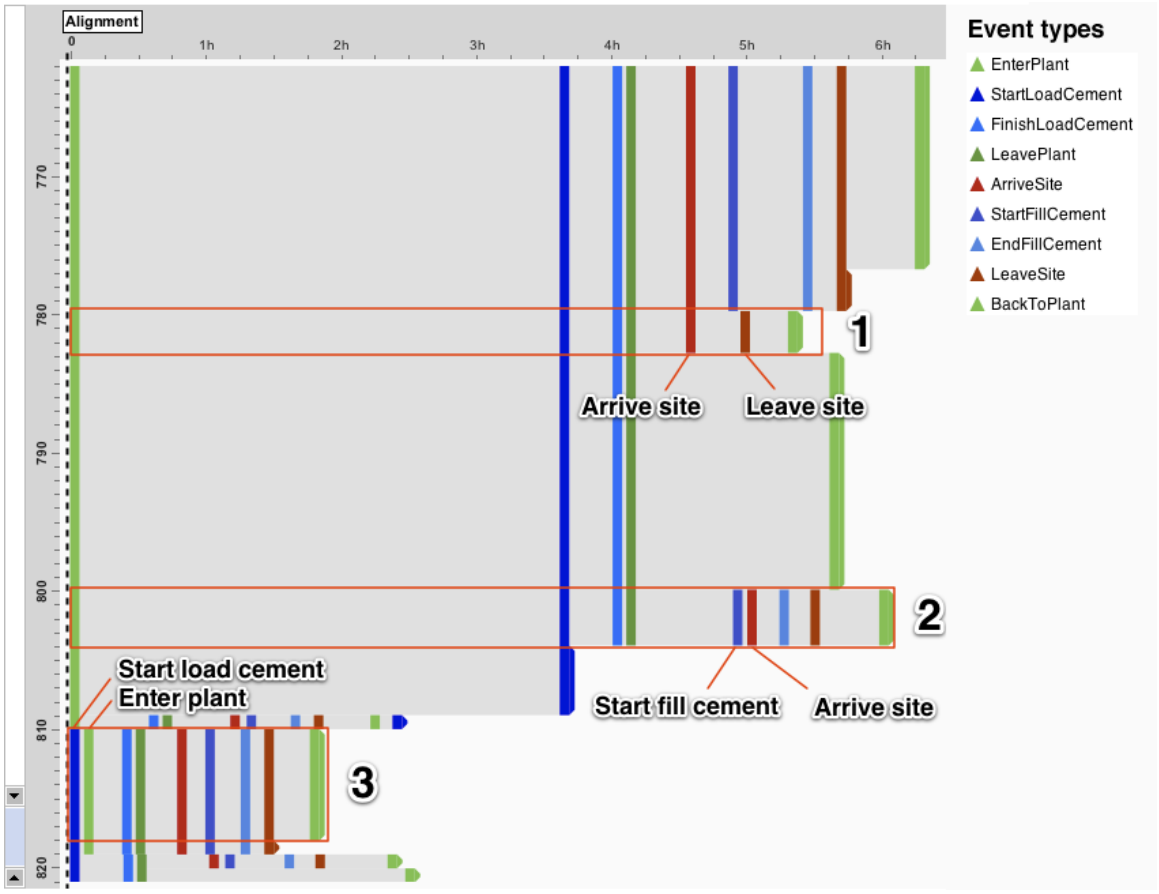

Figure 6.28: Anomalies: 1) Some trucks arrived to the sites but did not fill cement. 2) Some trips were reported to have begun filling cement before arriving to the sites. 3) Some trips were reported to have loaded cement before entering plants.

time distribution of each step by placing the cursor over and select some trips that took a longer time than usual from the distribution.

6.8.4.2 Anomalies

In addition to the normal sequences, we also noticed some rare sequences on the bottom of the screen, so we zoomed into those sequences (Figure 6.28). These anomalies could be placed into two categories:

1. *Incompleted trips*: Some trucks did not go to the sites or went to the sites but

did not fill the cement. A possible explanation was the trip might have been cancelled.

2. *Erroneous sequences*: Some trips were reported with `Start fill cement` before `Arrive site`, or `Start load cement` before `Enter plant`, which is illogical. Because the mobile sensors trigger events when reaching specified locations, this indicated inaccurate locations of some plants and sites.

6.8.4.3 Monitoring plants' performance

Mr. Dhanapanichakul observed that trips from some plants usually take more time, so we tried to confirm his observation using LifeFlow. However, these trucks sometimes stay overnight at plants or sites. Comparing the overall time from `Enter plant` to `Return to plant` will include overnight time, resulting in a biased comparison. Therefore, three event types; `Enter plant`, `Leave site` and `Return to plant`; were excluded to eliminate the overnight time and provide a more accurate comparison. Figure 6.29 shows the visualization after excluding the three event types and grouping by `Plant ID`. The ranking was changed to "Average time to the end" instead of the default "Number of records", so the plants were sorted by their trip time. The slowest plant was "C313", which is located far from the city. The average time from leaving this plant (*green*) to arrival at sites (*red*) was 45 minutes, which was much longer than other plants. From our discussion with the engineers at Freewill FX, this plant is responsible for a very wide area coverage and located in a district with very heavy traffic congestion.

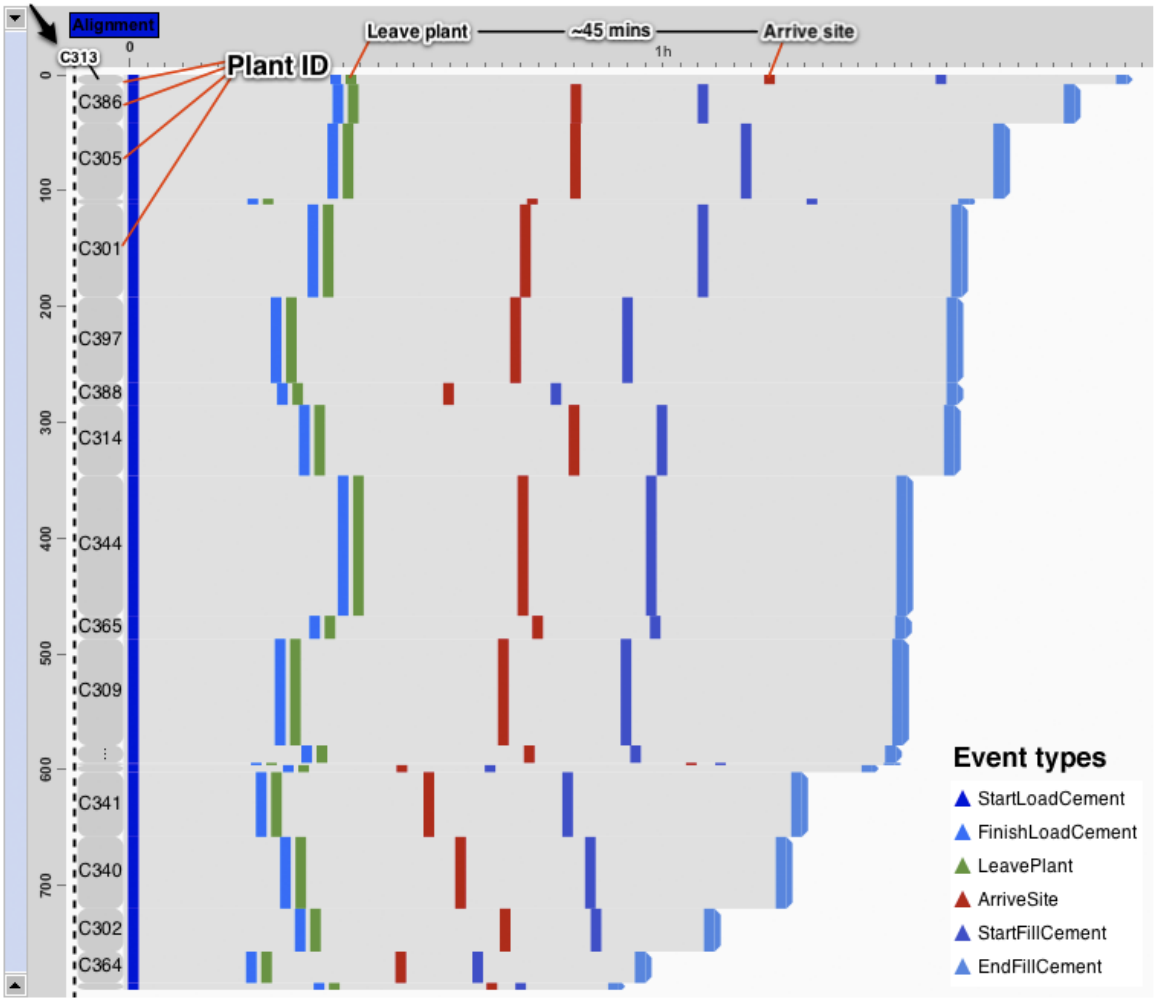

Figure 6.29: Trips are grouped by `Plant ID`. Three event types; `Enter plant`, `Leave site` and `Return to plant`; were excluded to eliminate the overnight time and provide a more accurate comparison. Trips from plant "C313" took on average 45 minutes from leaving plants (*green*) to arrival at sites (*red*), which was much longer than other plants. This is because of its wide area coverage and regular heavy traffic near its location.

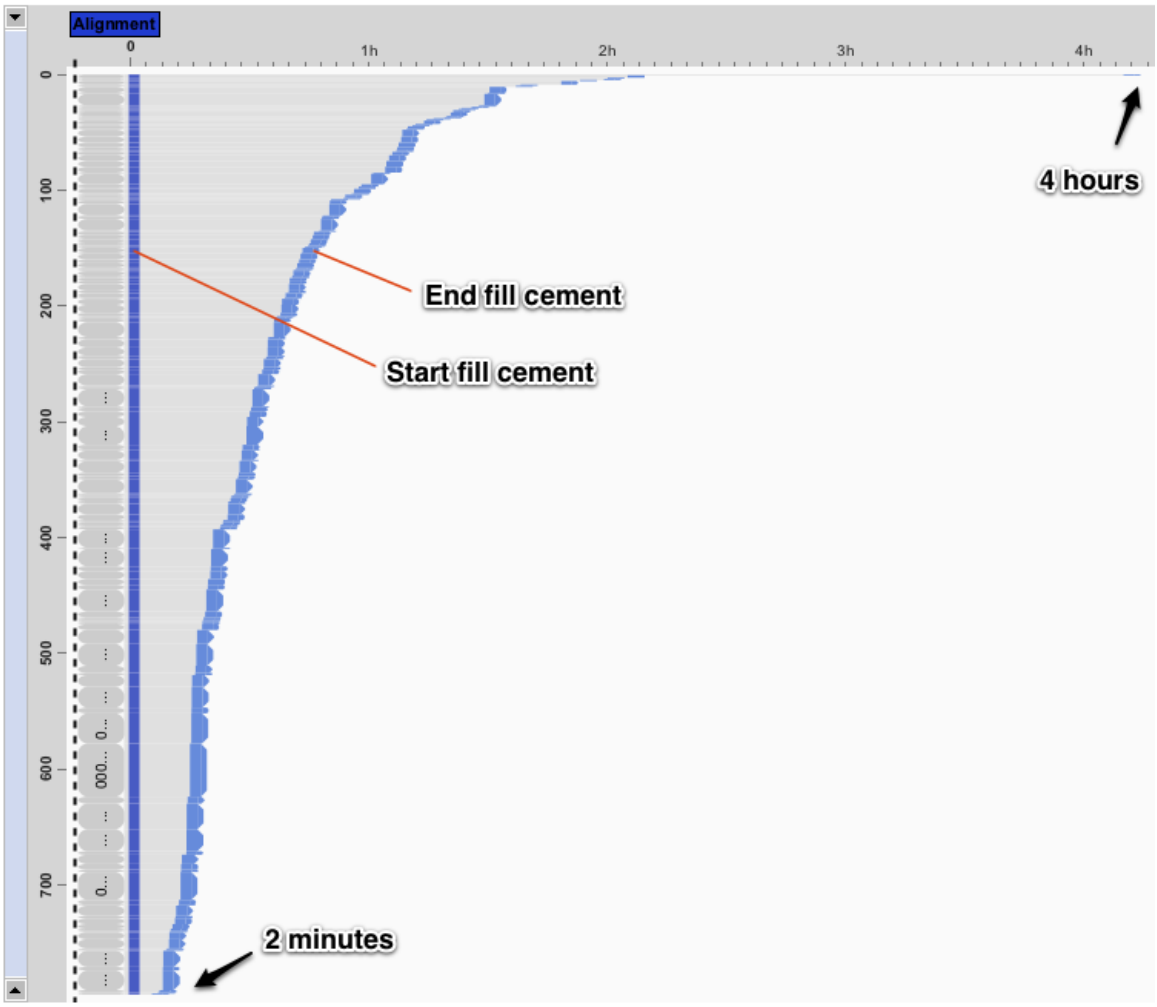

Figure 6.30: All events except `Start fill cement` (*blue*) and `End fill cement` (*light blue*) are hidden. Next, we aligned by `Start fill cement`, grouped by attribute `Site ID`, and ranked by "Average time to the end". We can see cement filling duration at each site ranging from two minutes to four hours.

6.8.4.4 Classifying customers from delay at sites

The cement company was interested in analyzing the cement filling duration at each site. This could help the manager decide how to treat their customers according to the delay incurred. For this analysis, we start by hiding all events except `Start fill cement` (*blue*) and `End fill cement` (*light blue*). Next, we aligned by `Start fill cement`, grouped by attribute `Site ID`, and ranked by "Average time to the

end". We could see duration ranging from two minutes to four hours on average (Figure 6.30). Interesting cases were low-performance sites that took quite a long time to complete the operation, e.g, four hours. Zooming into the top portion of the visualization helped us identify those sites.

Similar analysis can be performed to analyze these scenarios:

1. Group by `Plant ID` and analyze cement loading duration (time gap between `Start load cement` and `Finish load cement`).

2. Group by `Vehicle ID` and analyze travel time (time gap between `Leave plant` and `Arrive site`).

6.8.4.5 Search

The search could be useful in finding particular patterns. For example:

- Finding trips that had cement loading time longer than a specified time period, e.g., 40 minutes.

- Finding trips that had cement filling duration longer a specified time period, e.g., three hours.

- Finding trips that had travel time from plants to sites longer than a specified time period, e.g., 30 minutes.

6.8.5 Conclusions and Discussion

LifeFlow was applied to fleet management in this case study. Because the cement delivery process is structured and straightforward with little variations, LifeFlow can easily identify anomalous patterns and detect tracking errors. The ability to include and exclude events rapidly allows quick explorations of subsets of the data to answer particular questions, such as cement filling time or travel time from plants to sites. Grouping event sequences by attributes and ranking support easy comparison between subgroups, especially when all groups have the same sequence.

Mr. Chanma said that LifeFlow helps them see the data and understand the big picture. However, the learning curve can be challenging in the beginning. A proper training is required to help users understand the visualization and learn the interactions. The Freewill FX team envisioned the possibilities of using LifeFlow for another scenario when additional data are available. Everyday, the cement company has to decide how many cars should be sent to a particular plant at the same time. For example, should they send a fleet of six trucks or three fleets of two trucks each? Too large fleets can lead to long queues at destinations. To begin answering this question, we can group the trips by index of the truck in a fleet (1, 2, 3, ...), and see how long the following trucks in each fleet (2, 3, 4, ...) took to fill cement.

6.9 Soccer Data Analysis

6.9.1 Introduction

Sports are activities that captivate a wide range of audiences and attract major media attention. For example, the World Cup 2010 was broadcast in 214 countries and 715.1 million people—that is one in ten people alive at the time—watched the final. Key events and statistics from these competitive matches were carefully collected into sports databases. These databases are invaluable for coaches to learn their teams' performance and adjust their coaching strategies. Sports scientists are studying these data intensively [56] while sports fans also enjoy exploring fun facts from sports statistics.

In this study, I explore how LifeFlow can be used for analyzing sports data. The dataset is collected from Manchester United Football Club (Man U), an English professional soccer club, based in Old Trafford, Greater Manchester, that plays in the English Premier League. Manchester United has won the most trophies in English football, including a record 19 league titles and a record 11 FA Cups. It is one of the wealthiest and most widely supported football teams in the world.

I invited Mr. Daniel Lertpratchya, a PhD student at Georgia Institute of Technology and a devoted Man U fan who has been supporting the team for more than ten years, to participate in this study.

6.9.2 Procedure

I discussed the possibilities of analyzing data with LifeFlow with Mr. Lertpratchya. After he expressed his interest, I sent the software, data and introduction videos to him via email, and encouraged him to explore the data on his own. I also provided technical feedback as necessary. Two weeks after, we arranged an 1.5-hour online meeting via Skype to discuss his findings. I audiotaped this session and later transcribed and summarized all findings into a report. The total duration of this study was five weeks.

6.9.3 Data

I gathered the data from all 61 matches that Man U. played in season 2010-2011. Each match contains the following events:

- *Kick off*: Beginning of the match
- *Score*: Man U scored.
- *Concede*: Opponent scored.
- *Pen missed*: Man U missed a penalty.
- *Yellow*: Man U conceded a yellow card.
- *Red*: Man U conceded a red card.
- *Opp Yellow*: Opponent conceded a yellow card.
- *Opp Red*: Opponent conceded a red card.
- *Final whistle*: End of the match

The dataset also contains the following attributes for each match:

- *Competition*: English Premier League, UEFA Champions League (UCL), FA Cup, Carling Cup, FA Community Shield or friendly
- *Opponent*: The opponent team
- *Result*: win, loss or draw
- *Venue*: home, away or neutral
- *Score*: Total Man U goal(s)
- *Opponent Score*: Total opponent goal(s)

6.9.4 Analysis

6.9.4.1 Finding entertaining matches

The first scenario that we discussed was finding entertaining matches to see replay videos. Mr. Lertpratchya defined his entertaining matches as follows:

1. *Matches that Man U dominated and scored several goals*

   Mr. Lertpratchya used the LifeFlow view to select matches that Man U scored the first three goals in a row (Figure 6.31). The detail view on the right show matches against Birmingham City, Blackburn Rovers, Bursaspor, MLS All Stars, Newcastle United, West Ham United and Wigan Athletic. He was interested in the two highest scoring matches against Blackburn Rovers and Birmingham City when Man U won 7-1 and 5-0, respectively. Using tooltips to learn about goalscorers (event attributes), we found that Dimitar Berbatov scored five goals in the Blackburn game and also a hat-trick (three goals) in the Birmingham game.

Figure 6.31: Matches that Man U scored the first three goals

2. *Matches that both teams scored in "an eye for an eye" fashion*

   Another type of entertaining matches was when one team scored a goal and another team retaliated by scoring in return. We looked for patterns with alternating `Score` (*green*) and `Concede` (*purple*) bars and found the match against Wolverhampton Wanderers (Wolves) in the Carling Cup (Figure 6.32). In this game, Man U scored and was equalized by Wolves twice before Javier Hernandez was substituted and scored the winning goal in the stoppage time.

3. *Matches that Man U conceded early goals and came back to win the game*

   Mr. Lertpratchya grouped the matches by results and focused on the winning matches that start with `Concede` (*purple*). There were matches against Blackpool and West Ham United that Man U conceded two goals and came back

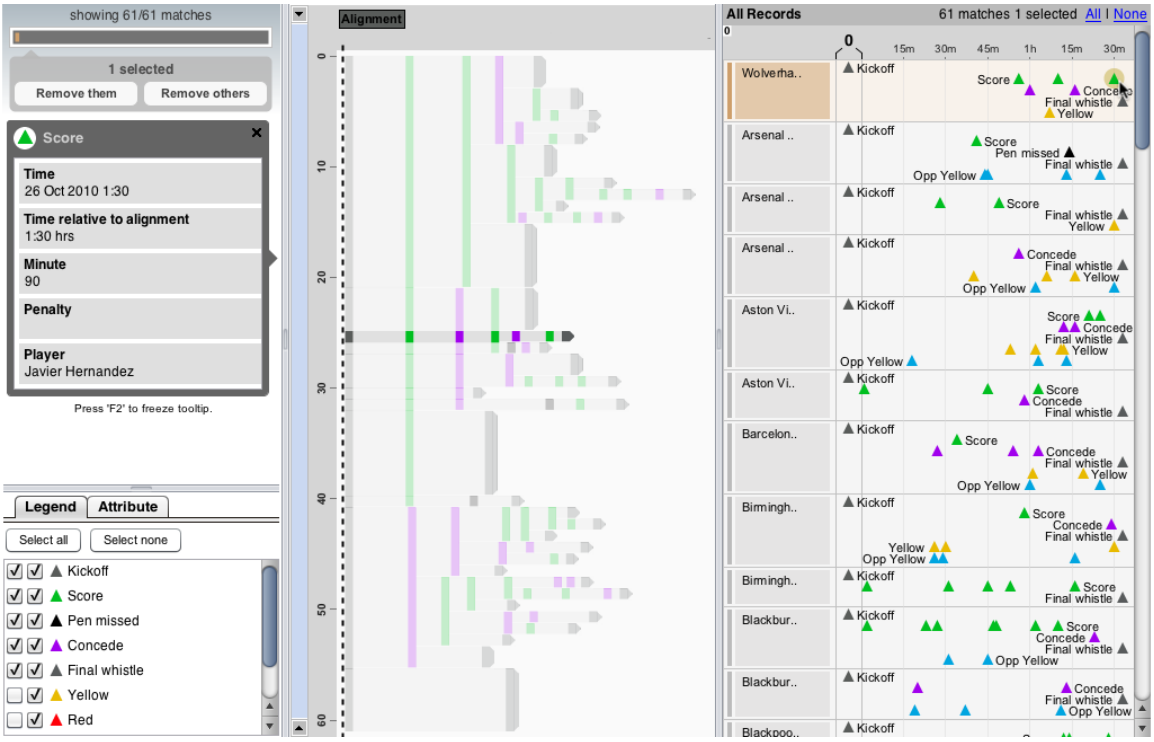

Figure 6.32: Matches that both teams scored in "an eye for an eye" fashion

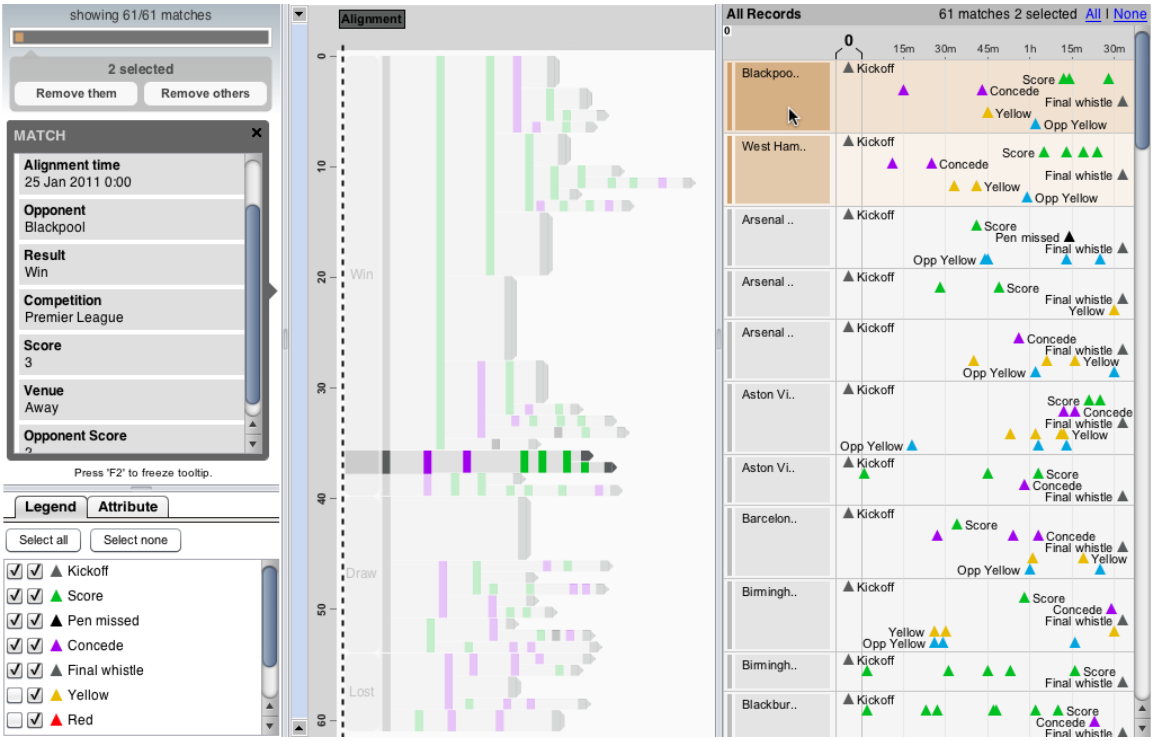

Figure 6.33: Matches that Man U conceded early goals and came back to win the game

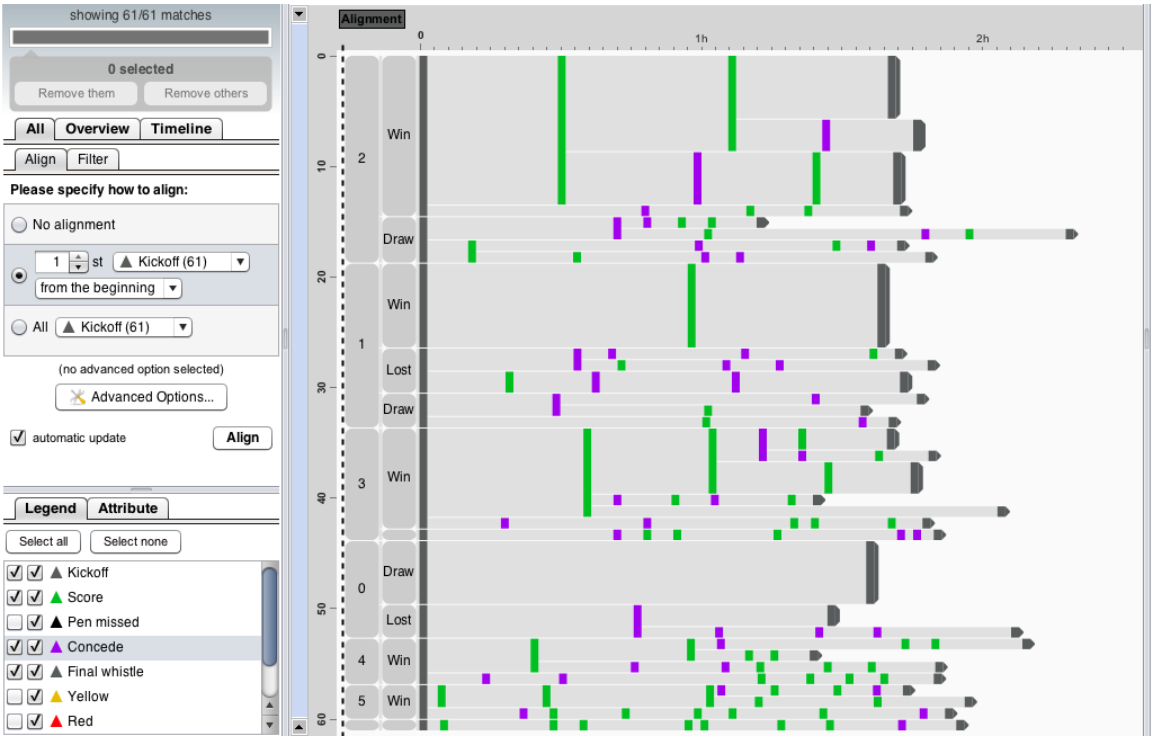

Figure 6.34: Matches grouped by `Score` and `Result`

to win 3-2 and 4-2, respectively (Figure 6.33).

6.9.4.2 Predicting chances of winning

To see the percentage of winning when Man U scored n goals, we grouped matches by `Score` and `Result`. LifeFlow in Figure 6.34 shows all proportions of win, loss and draw when Man U score n goals. The proportion can be used for a rough estimation of winning.

6.9.4.3 Explore statistics

Mr. Lertpratchya and I explored the dataset from sports fans' perspective and found the following interesting facts and observations:

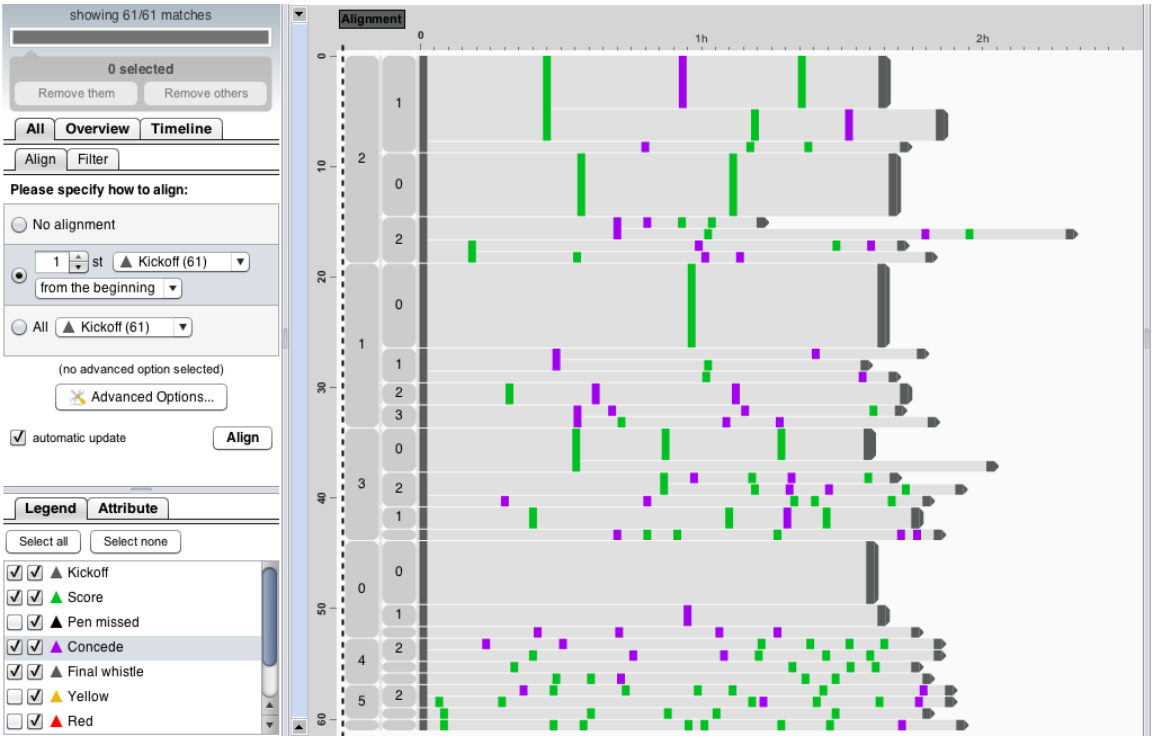

Figure 6.35: Summary of all scorelines

1. **Scoreline**

   We grouped the matches by `Score` and `Opponent Score` (Figure 6.35). From the visualization, we found that:

   - *Man U often scored two goals (19/61 matches).*
   - *There were only nine matches (14.75%) that they did not score.*
   - *2-1 was the most popular scoreline (nine matches), followed by 1-0 (eight matches).*
   - *In four out of a total of eight matches that Man U won 1-0, the winning goals occurred within the last ten minutes.* – We selected all 1-0 matches, removed others and saw the distribution split into two groups: matches with early goals and late goals (Figure 6.36).

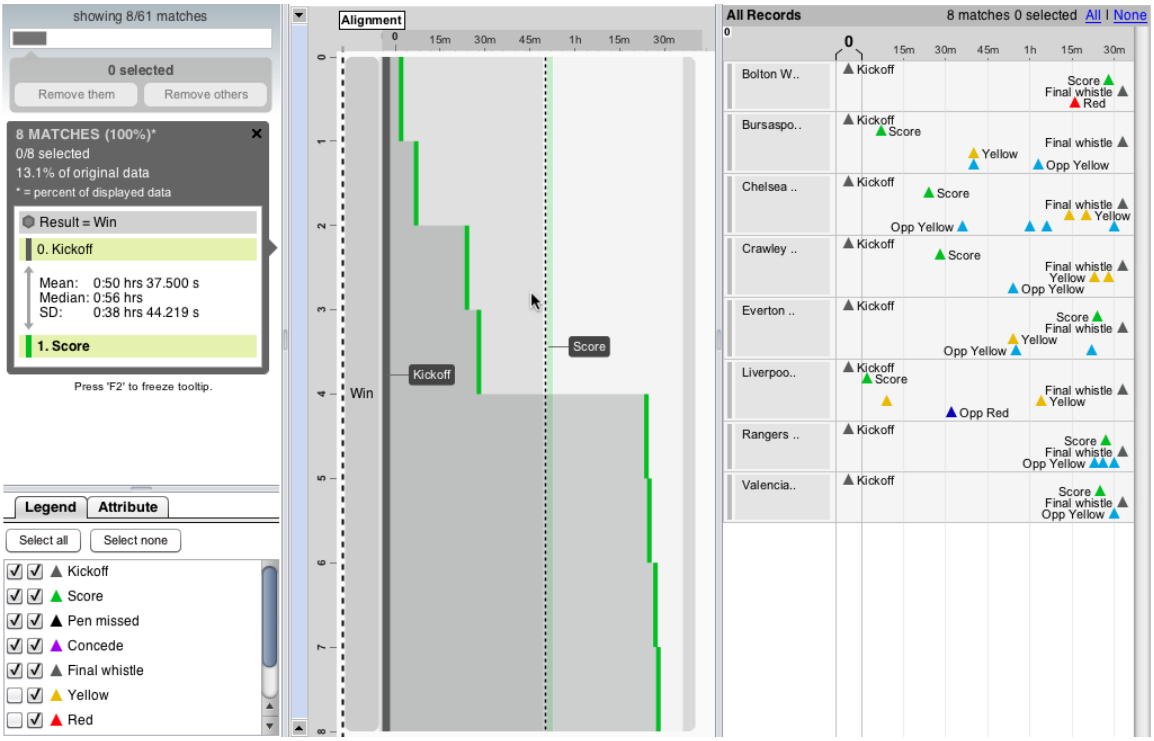

Figure 6.36: A distribution of the winning goals in all 1-0 matches: Matches are split into two groups: matches with early goals and late goals.

2. **Opponent**

   Figure 6.37 shows the matches grouped by *Opponent*.

   - *Man U competed against Chelsea more often than any other teams (five matches), and always scored the first goal.*

3. **Competition**

   Grouping the matches by `Competition`, we saw the difference between their performance in the English Premier League (EPL), a national competition, and the UEFA Champions League (UCL), the biggest tournament in Europe (Figure 6.38).

   - *Man U scored the first goal in the EPL faster than in the UCL*: On av-

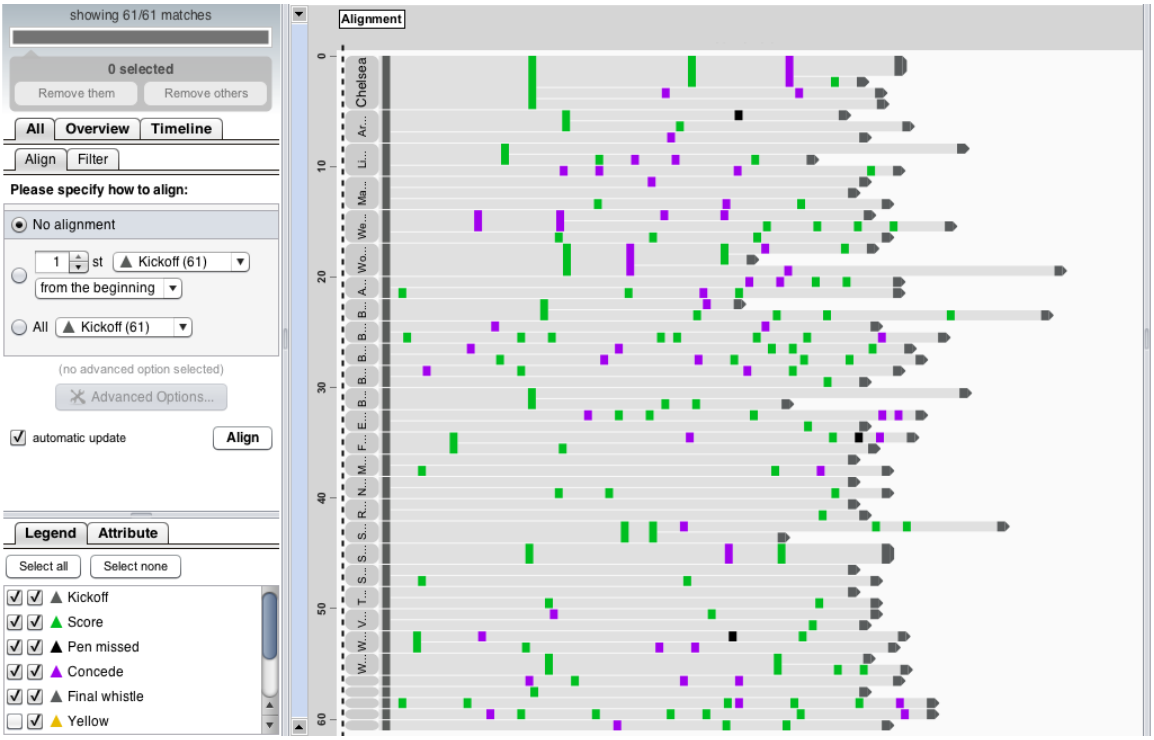

Figure 6.37: Matches grouped by `Opponent`: Man U competed against Chelsea in this season more often than any other teams.

erage the first goals came after 27 and 43 minutes for EPL and UCL, respectively. This is understandable because most teams tended to be more cautious in the UCL.

- *Matches with many goals occurred in the EPL more than in the UCL*: Since the UCL is a more prestigious competition that consists of all the best teams in Europes, the teams' qualities are higher and, therefore, more difficult to score against.

- *In all matches that Man U won in the UCL, they scored the first goals*: On the other hand, if they conceded the first goal, they drew or lost (Figure 6.39).

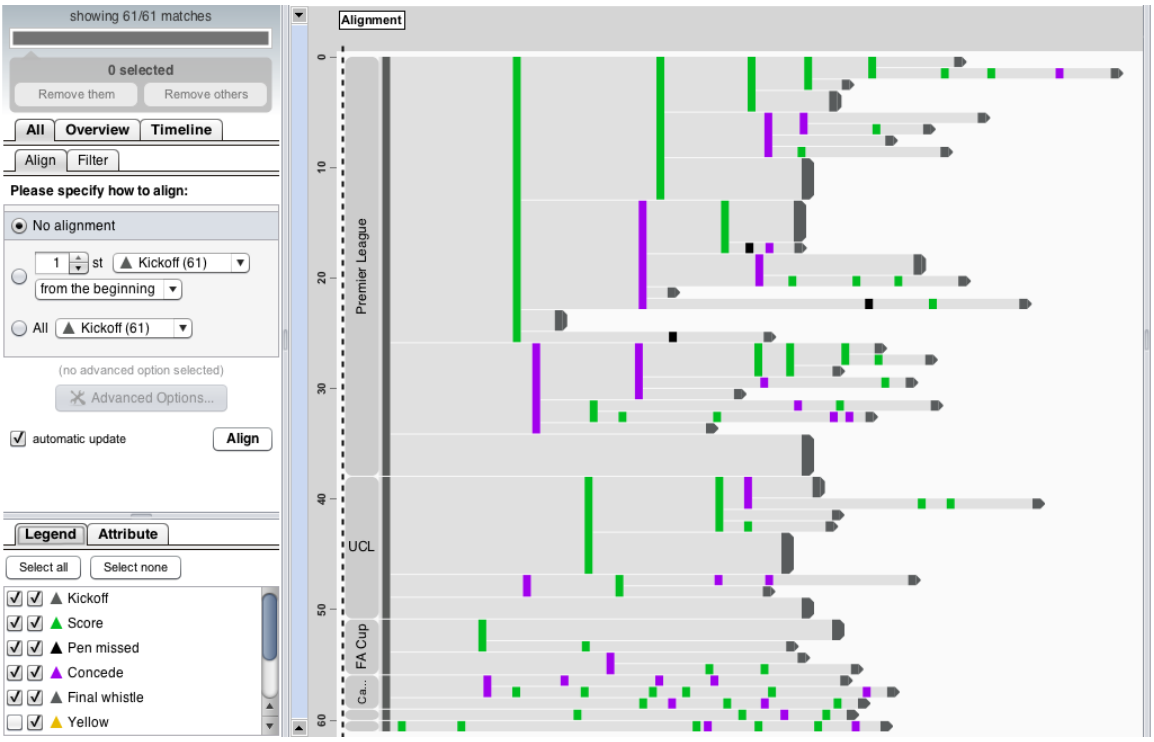

Figure 6.38: Matches grouped by `Competition`: Man U scored (*green*) the first goal in the English Premier League (EPL) faster than in the UEFA Champions League (UCL).

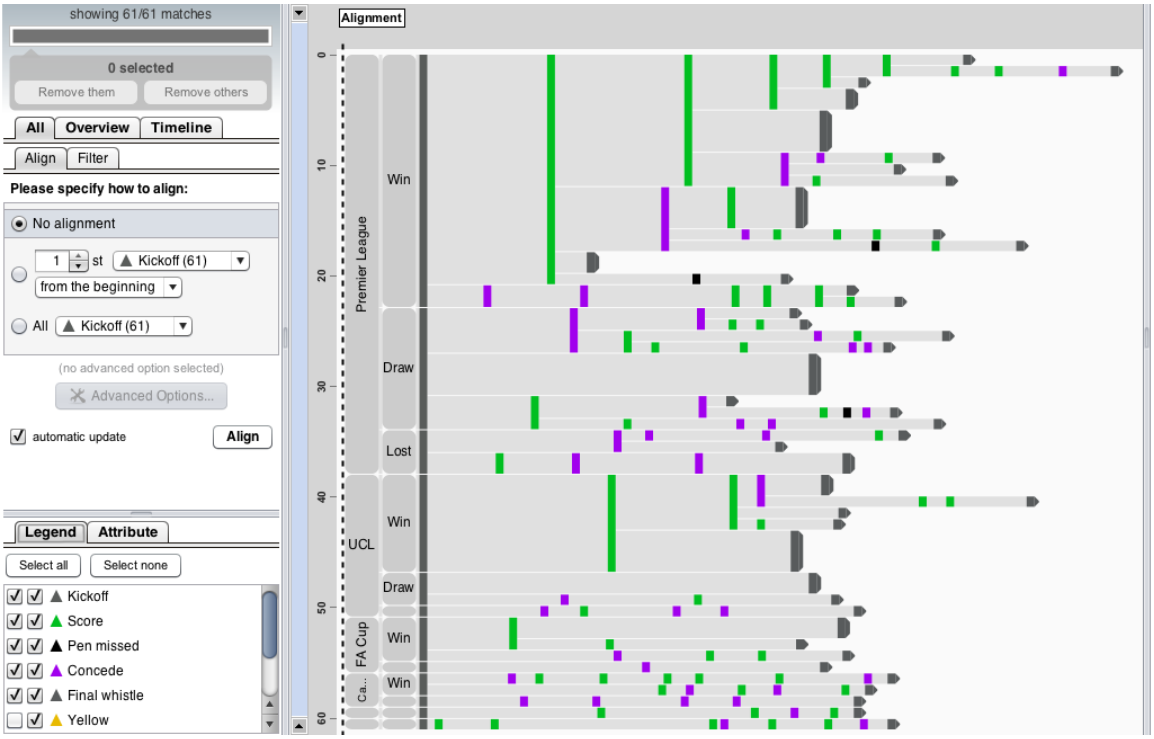

Figure 6.39: Matches grouped by `Competition` and `Result`: Man U scored (*green*) the first goal in all matches that won in the UCL.

4. **Home and Away**

Venues played an important role in the team's performance.

- *Man U is very strong at home but not as strong on the road*: Grouping by `Venue`, the visualization shows that Man U scored first in most of the home matches. Grouping by `Venue` and `Result` (Figure 6.40), we found that they won most of the home matches and scored first in all of the winning home matches. Grouping by `Result` and `Venue` (Figure 6.41) shows that many scoreless (0-0) draws are away matches.

- *For away matches, the number of matches decreased with an increasing number of goals scored, but the same is not true for home matches*: The

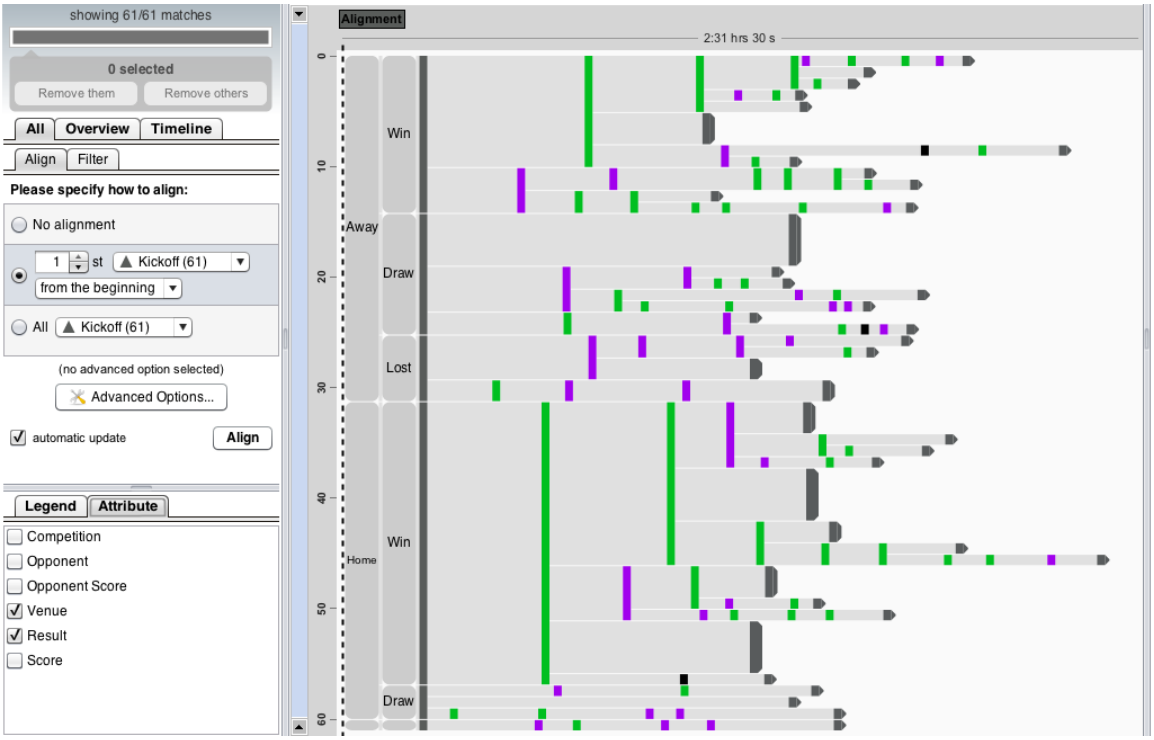

Figure 6.40: Matches grouped by `Venue` and `Result`: Venues played an important role in the team's performance. Man U had a great performance at home but was not as strong on the road.

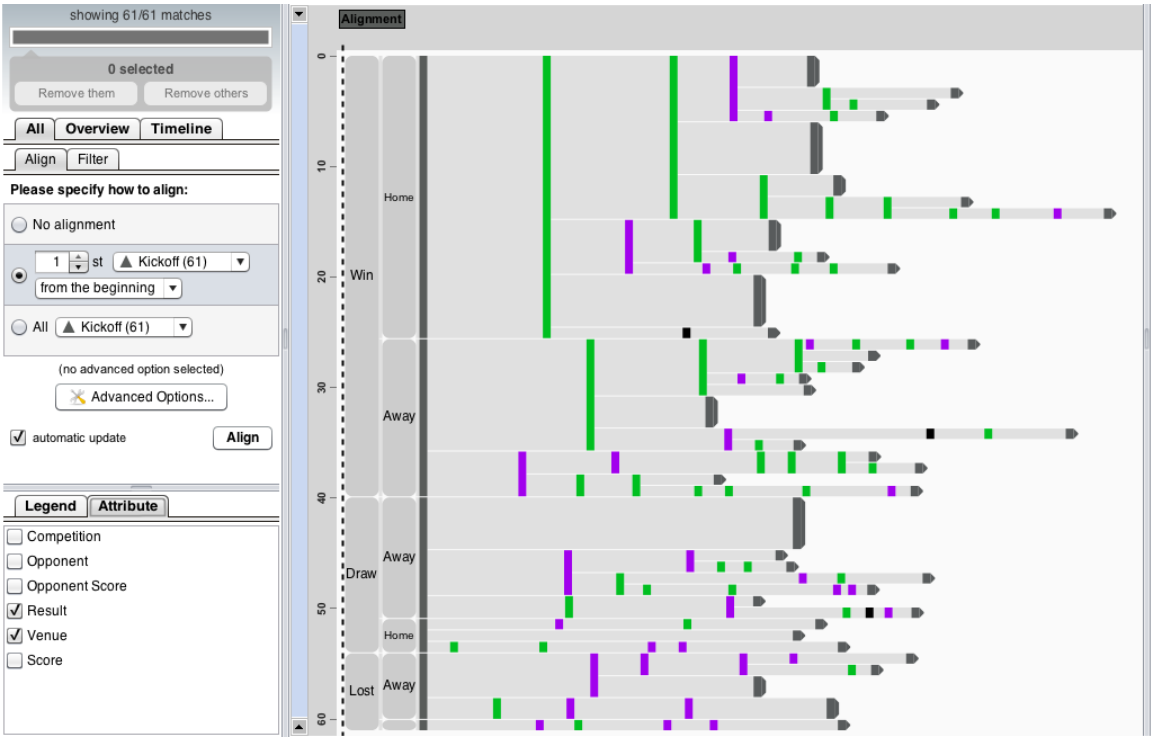

Figure 6.41: Matches grouped by `Result` and `Venue`: Most common score in a draw match is 0-0. Most of these matches are away matches.

number of goals sorted descending by the number of matches with that amount of goals are 0,1,2,3,4 and 5 for away matches and 2,1,3,4,0,5 and 7 for home matches (Figure 6.42).

5. **Goals**

Goals are the most important events in soccer matches.

- *Man U often scored first*: They scored first 41 times while the opponents scored first 14 times. The patterns can be seen in Figure 6.43.
- *Three fastest goals occurred on the first minute*: We aligned by `Kickoff` and hide `Concede`. Using a selection from distribution, we found three matches that Man U scored very early (Figure 6.44). One match was

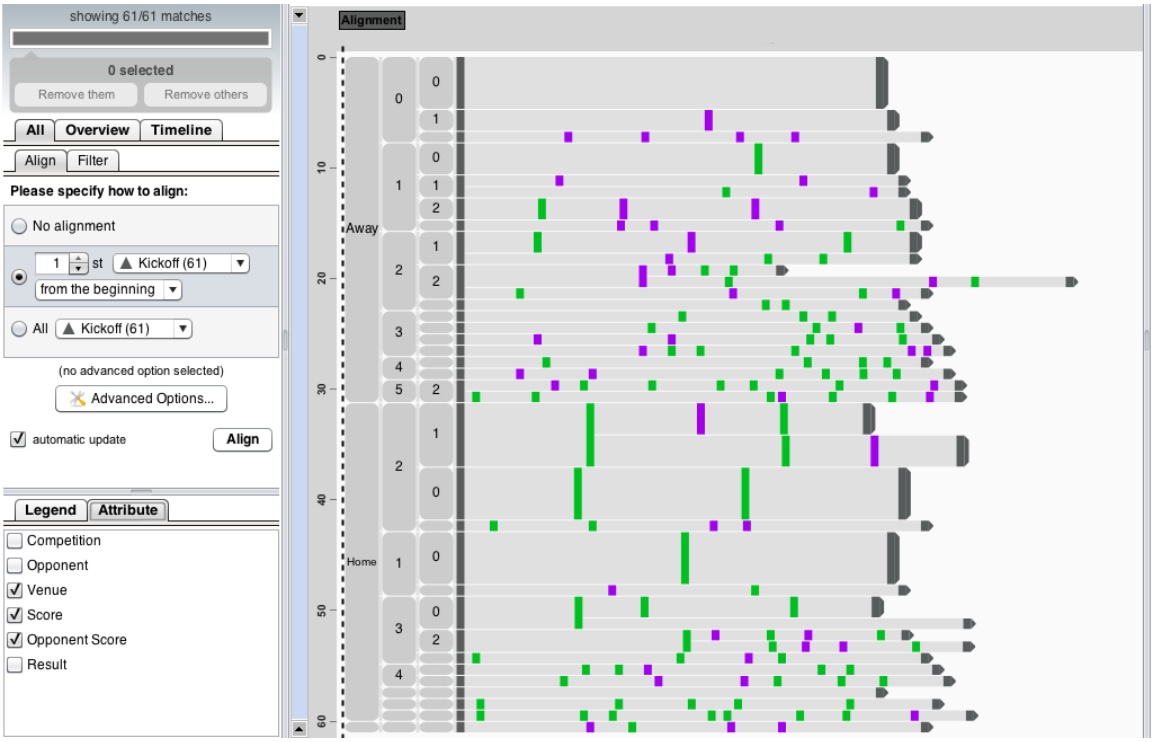

Figure 6.42: Matches grouped by `Venue`, `Score` and `Opponent Score`: For away matches, the number of matches decreased with an increasing number of goals scored, but the trend is not the same for home matches.

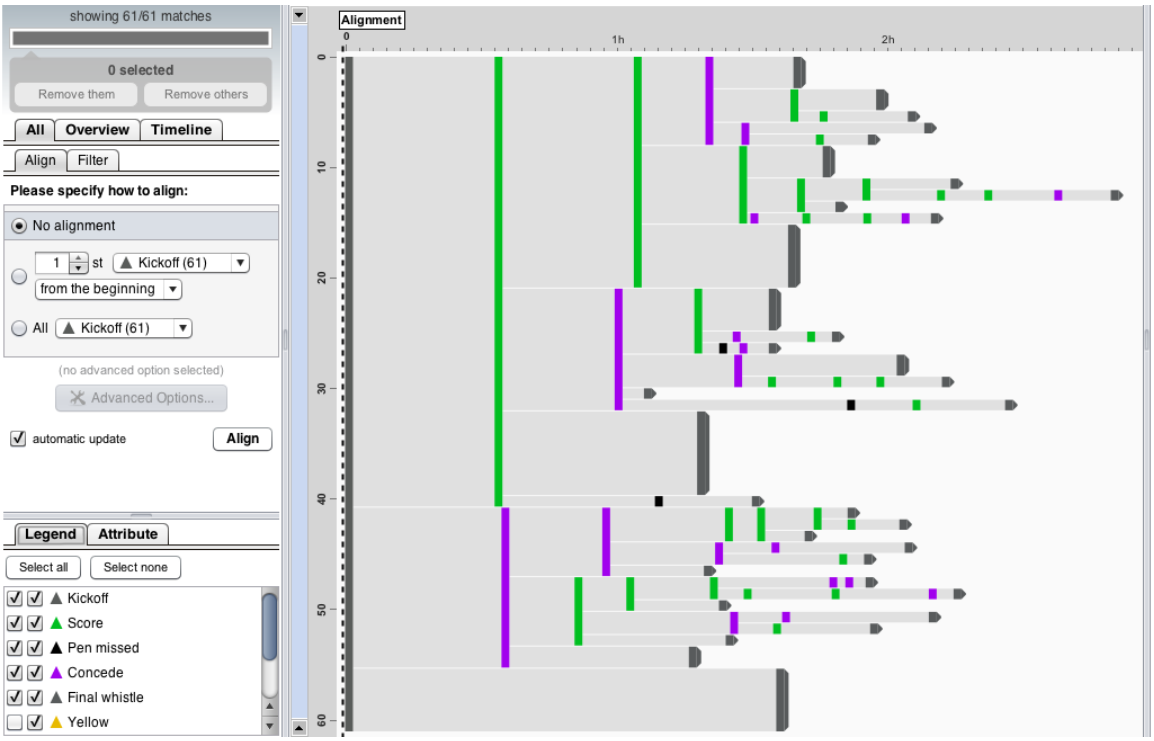

Figure 6.43: All matches that Man U played in season 2010-2011

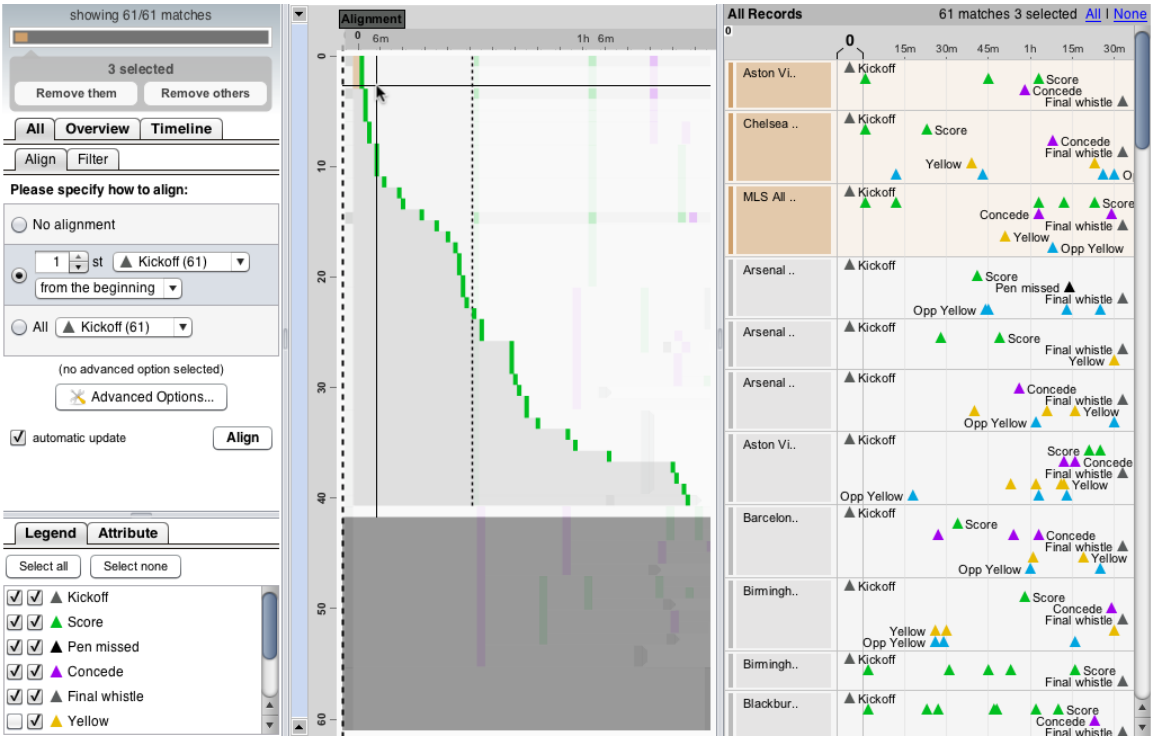

Figure 6.44: Three fastest goals occurred on the first minute: Using a selection from distribution, we could select matches that Man U scored very early.

a friendly match against MLS all stars. Federico Macheda was the goalscorer. The other two matches are in EPL against Aston Villa and Chelsea. Javier Hernandez scored in both of them.

- *Javier Hernandez was the most frequent scorer of the first goal*: We right-clicked on the first green bar (first goal) and brought up a summary of event attribute "Player". The summary table in Figure 6.45 lists frequent scorers of the first goal: Javier Hernandez (ten matches) , Dimitar Berbatov (nine matches), Wayne Rooney (eight matches).

- *Javier Hernandez scored the first goal against Stoke City on the 27th minute twice*: By clicking on "Javier Hernandez" in the summary table, the matches that Hernandez scored were selected (Figure 6.45). We

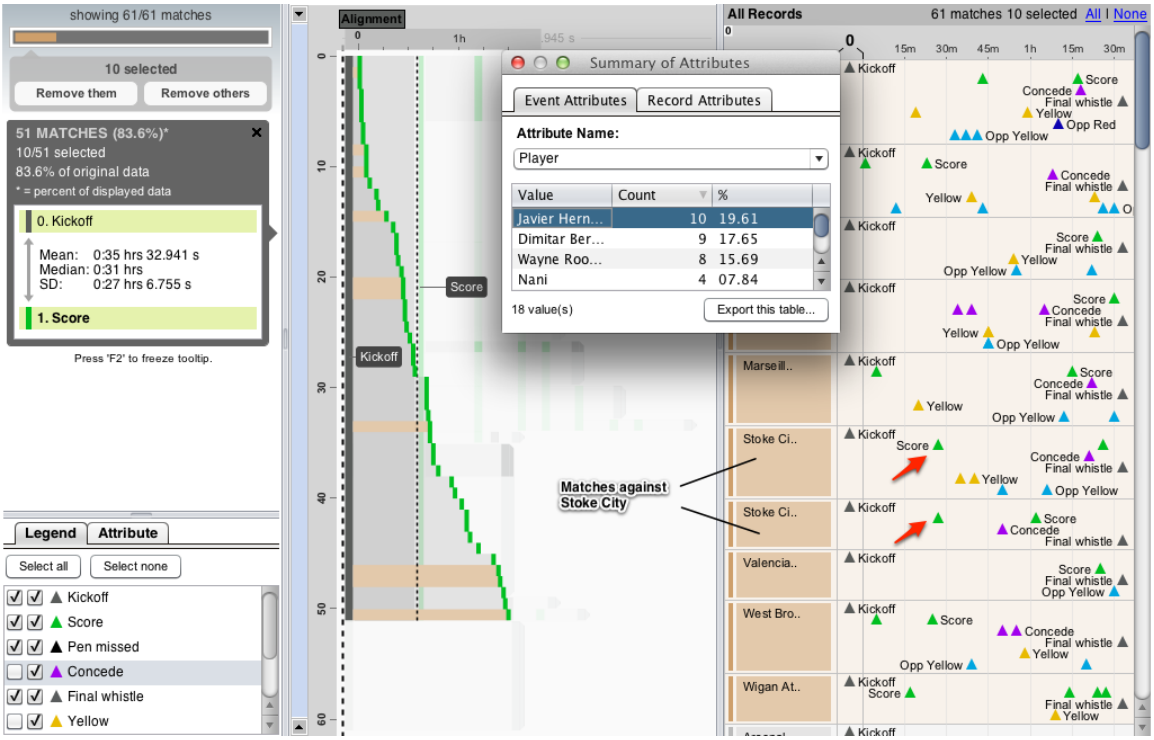

Figure 6.45: Javier Hernandez and first goals: We right-clicked on the first green bar (first goal) and brought up a summary of event attribute "Player". Javier Hernandez was the most frequent scorer of the first goal. He also scored the first goal against Stoke City on the 27th minute twice.

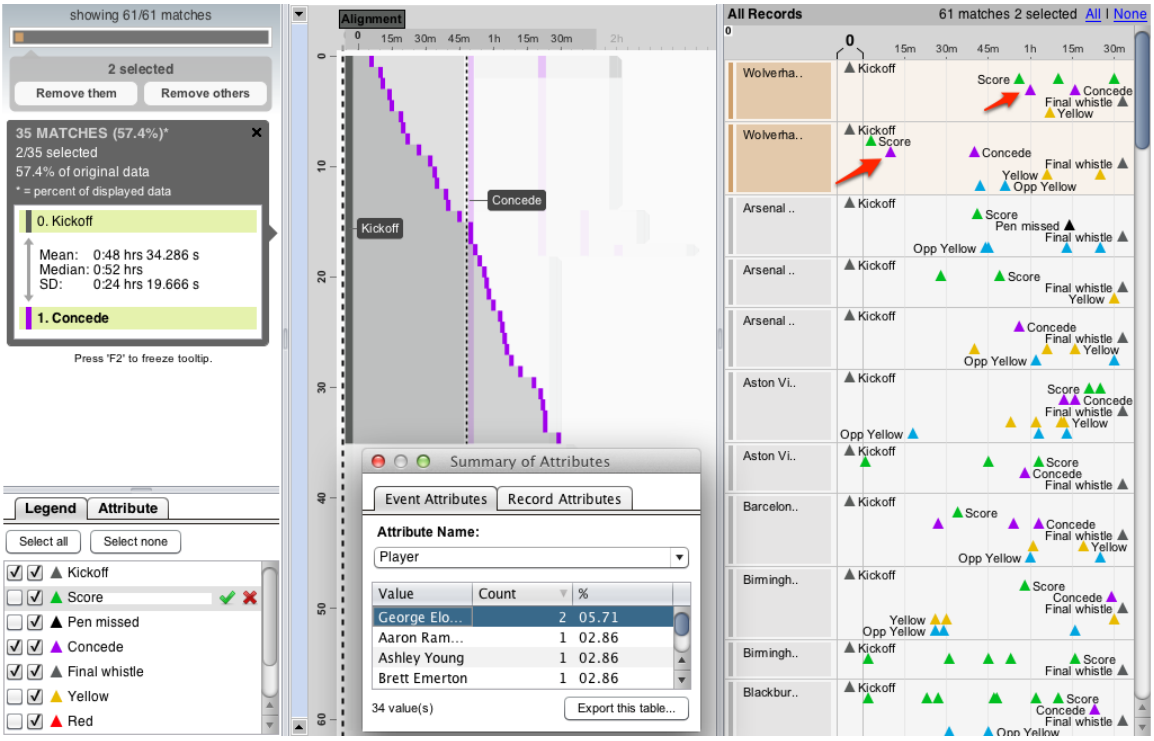

Figure 6.46: George Elokobi, Wolverhampton Wanderer's left defender scored the first goal against Man U twice. Those two goals were the only two goals in his total 76 appearances for Wolves from 2008–2011.

looked at the entire match detail from the LifeLines2 view and noticed that he scored against Stoke City in both fixtures on the exact same minute (27').

- *George Elokobi, Wolverhampton Wanderer's left defender, not any world-class striker, scored the first goal against Man U most often*: We had looked that Man U players who scored the first goal so we were curious about opponent players who scored the first goal. We hid `Score` and showed `Concede` instead, then right-clicked on the first `Concede` (*purple*) bar to see attribute summary (Figure 6.46). Each team meet with Man U only 2-5 times in this season, so most players scored the first goal

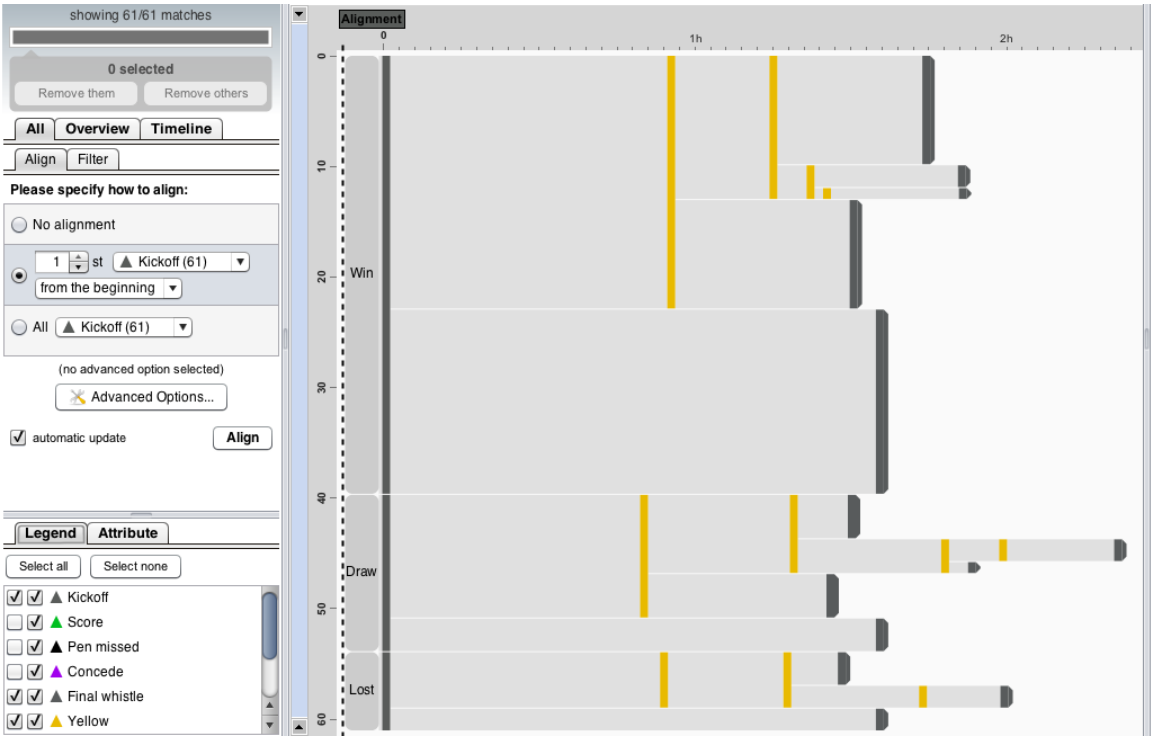

Figure 6.47: Display only yellow cards and group matches by `Result`: Man U received less booking in winning matches.

against ManUtd only once, except Mr. Elokobi, who scored twice. It was surprising because he is an unknown defender from a small team. Those two goals were also the only two goals in his total 76 appearances for Wolves from 2008–2011.

6. **Yellow Cards**

- *Paul Scholes received the first yellow card most frequently*: We hid all events except `Kickoff`, `Yellow` and `Final Whistle`. We right-clicked the first yellow card bar and brought up the attribute summary and found that Paul Scholes, not so surprisingly, received the first yellow cards five times.

- *Man U received less booking in winning matches*: We grouped matches by `Result` and found that 17 out of a total of 40 winning matches were free from yellow cards (Figure 6.47). This supported our assumption that the opponents did not put much pressure to Man U defence, so there were not many fouls. We also thought that the players could be less aggressive and more disciplined when they were winning.

- *In all matches that Man U lost, the opponents received yellow cards.*– This was the opposite situation. We hid all events except `Kickoff`, `Opp Yellow` and `Final Whistle` and grouped matches by `Result`. The opponents received yellow cards in all matches that Man U lost. We made an assumption that Man U were attacking to score back and the opponents had to stop them by committing fouls.

7. **Red Cards**

Figure 6.48 shows only `Red` and `Opp Red` and groups matches by `Result`.

- *When opponents received red cards, Man U always won*: The team took advantages of having an extra player well and turned them into victories.

- *There was not any match that both teams received red cards*: Sometimes this situation happens in a derby or rival match when the atmosphere is fiery, or a stressful match when the stake is high. However, it did not happen during the duration of this dataset.

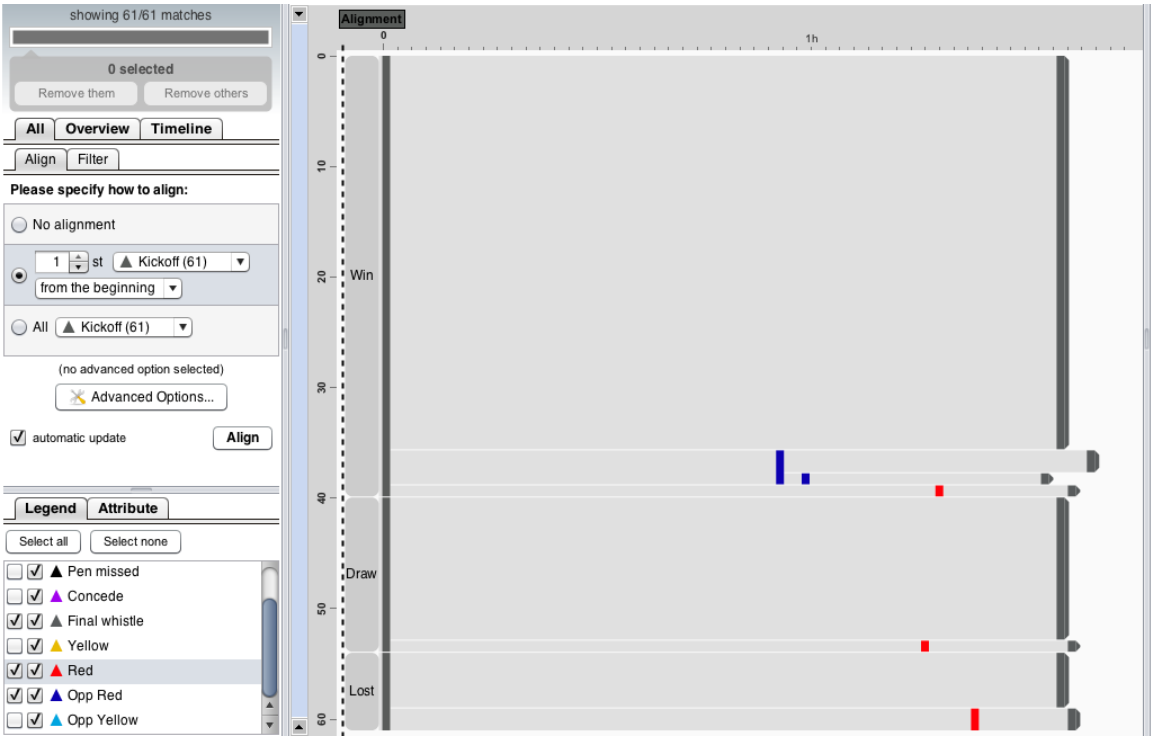

Figure 6.48: Display only red cards for both sides and group matches by `Result`: When opponents received red cards (`blue`), Man U always won. There was one match that Man U won with 10 players.

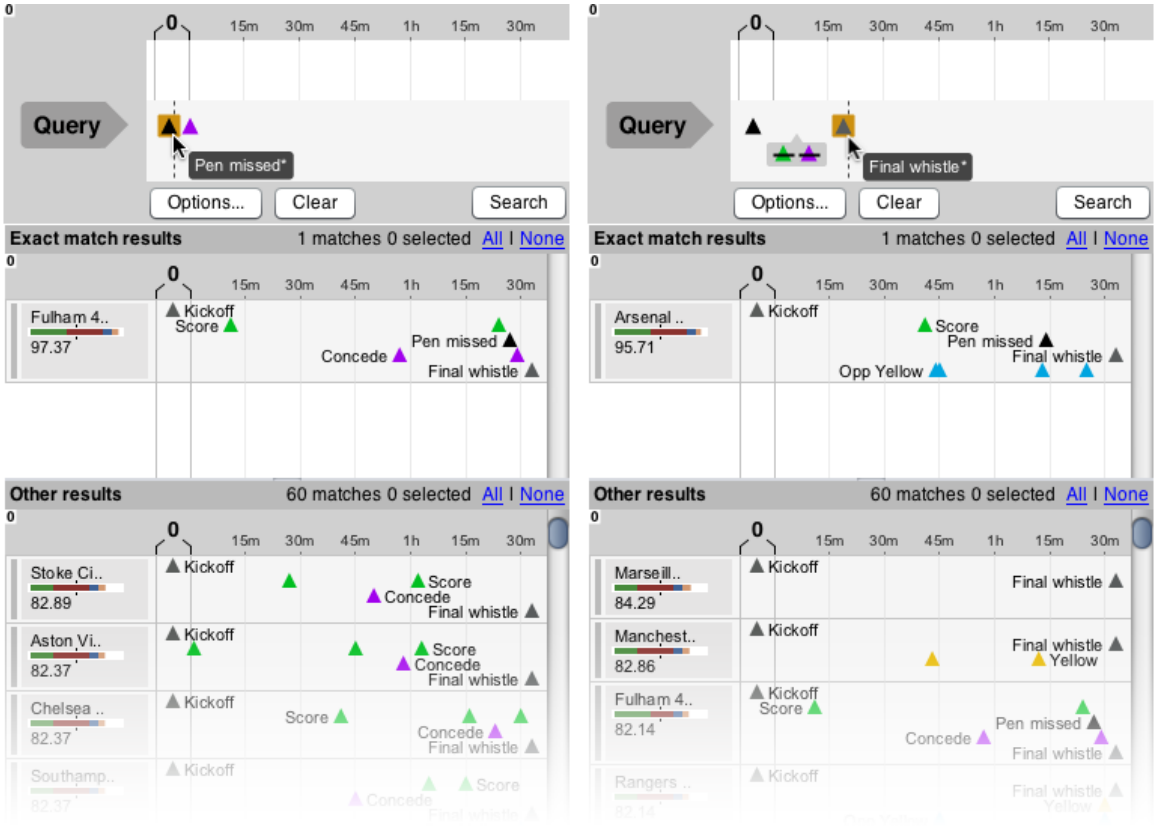

Figure 6.49: (left) *Missed a penalty then conceded a goal*: A disappointment for the fans as the match against Fulham ended with a draw at 2-2. Nani had a chance to make it 3-1, but he missed the penalty kick. (right) *Missed a penalty but nothing happened after that*: Man U was awarded a penalty kick while the team was leading 1-0, but Wayne Rooney could not convert it. However, the opponent could not score an equalizer.

6.9.4.4 Search for specific situations

Near the end of our session, we discussed whether some situations exist in the matches or not. To answer these questions, we used the FTS to specify and search for several situations. Some of the interesting queries are reported below:

1. *Missed a penalty then conceded a goal*: This was an example of situations that could lead to newspaper headlines. We found a match against Fulham, in which Nani missed a penalty while Man U was leading 2-1 (Figure 6.49 left).

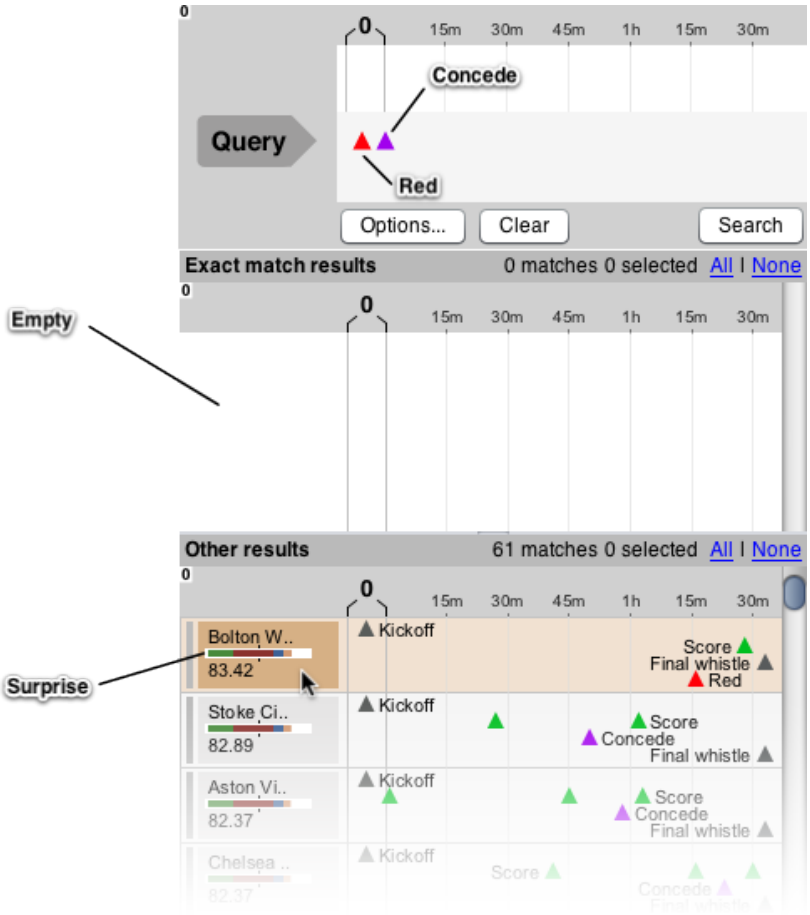

Figure 6.50: Received a red card then scored a goal: A user searched for a situation when Man U received a red card then conceded a goal. However, he could not find any, but found a match against Bolton when Man U scored after receiving a red card instead.

After that Fulham equalized and the match ended at 2-2.

2. *Missed a penalty and nothing happened after that*: This was the opposite of the previous situation when the team made mistake and was not punished. We searched for a pattern `Pen missed` → no `Concede` / no `Score` → `Final whistle` (Figure 6.49 right). There was a match against Arsenal, in which Wayne Rooney missed a penalty while Man U was leading 1-0. Luckily, the match ended at 1-0.

3. *Received a red card then conceded a goal*: This was a possible situation because another team might take advantage of having more players. However, we did not find any match with this pattern (Figure 6.50). The exact match results was empty. To confirm the results, we looked at the other results and there was not any match with red card (*red*) followed by conceding a goal (*purple*). Instead, we surprisingly found a match when Man U received a red card then scored a goal.

6.9.5 Conclusions and Discussion

This dataset, despite its small size (61 records), is interesting because it has several attributes that can be used in combination to produce interesting perspectives. Including/Excluding attributes and changing attribute orders were heavily used features in the analysis. The dataset also contains rich event attributes, such as goalscorers, that highlight the usefulness of attribute summaries.

Mr. Lertpratchya was able to use the software comfortably after watching an

introduction video that I sent him. There were only a few issues that he raised during the study. The first one was about the time axis in LifeFlow. He was wondering why some patterns were shorter than 90 minutes, despite the fact that a soccer match is always 90 minutes long. This is because LifeFlow displays a sum of averages, which can be shorter or longer than the actual time. Another issue was when we were looking at yellow cards in the LifeLines2 view, we knew that the selected yellow card is for a specific player, but the tooltip showed another player that we did not expect. After a moment of frustration, we realized that both players had yellow cards at the same time and the events overlapped. On the positive side, he was satisfied with the software and enjoyed our discussion because it aligned with his interest. He could understand many interesting facts in a short period of time. In the last part of our study where the FTS was used primarily, I found that having the results displayed in two parts give more confidence to the results. When the exact match results are empty, Mr. Lertpratchya examined the other results and confirmed that the situations he was looking for did not exist. This is more beneficial than returning nothing and leave the users wonder whether the query is wrong or the situation really does not exist.

In summary, this case study demonstrates many ways to use LifeFlow with soccer data. It can help video subscribers pick interesting matches from an archive to watch. Mr. Lertpratchya was able to select matches that he felt were entertaining. We also discovered many fun facts in the study. Sports journalists can use LifeFlow to find these fun facts and write about them in the news. Sports fans like Mr. Lertpratchya can enjoy exploring interesting statistics in their free time or use

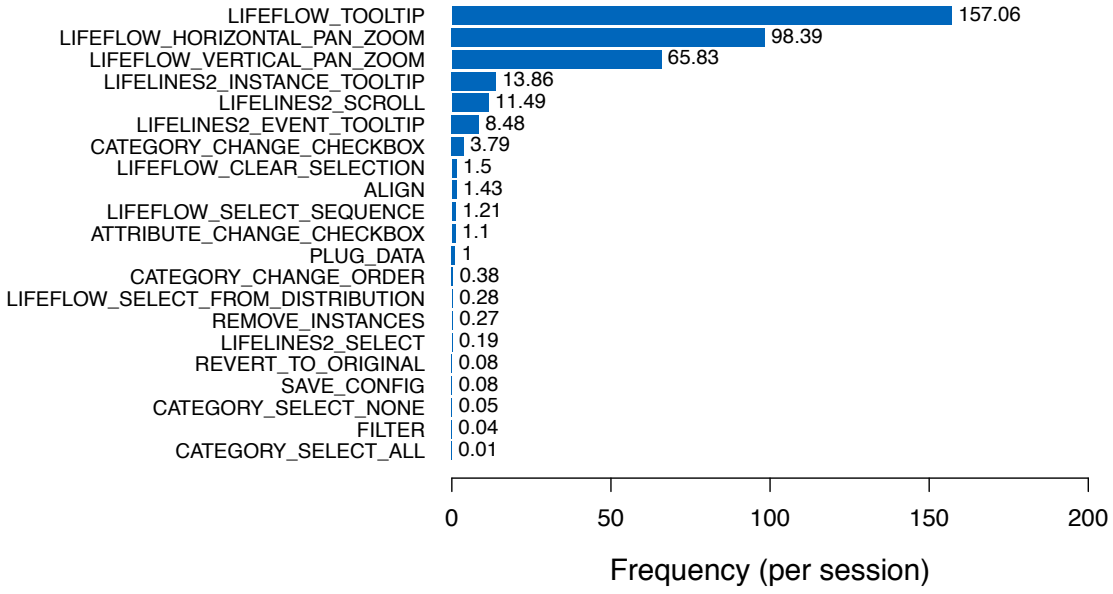

Figure 6.51: Features of LifeFlow used in 238 sessions

them to predict match results. Coaches can analyze their teams' performance with LifeFlow and adjust their tactics. Moreover, LifeFlow has potential to be useful for other sports, which a normal user can foresee. Near the end of our discussion, Mr. Lertpratchya asked about the possibilities of using LifeFlow for football, hockey, and other sports.

6.10 Usage Statistics

I have collected action logs of LifeFlow usage since March 2011. After removing short sessions, which resulted from debugging and testing, 238 sessions were qualified for usage analysis. The frequencies of actions were counted and plotted as a bar chart (Figure 6.51).

As shown in Figure 6.51, the most common action was `LIFEFLOW TOOLTIP`, followed by `LIFEFLOW HORIZONTAL PAN ZOOM` and `LIFEFLOW VERTICAL PAN ZOOM`.

The high frequencies of these three actions indicated that users spent most of the time navigating and exploring the LifeFlow overview visualization. The next three most common actions were LIFELINES2 INSTANCE TOOLTIP, LIFELINES2 SCROLL and LIFELINES2 EVENT TOOLTIP, which implied that users were also examining records in detail using the LifeLines2 view on the right side, but less frequently.

From the frequencies of the three tooltip actions, users seemed to pay more attention towards higher-level information. LIFEFLOW TOOLTIP, a tooltip for LifeFlow sequence, was used more than LIFELINES2 INSTANCE TOOLTIP, a tooltip for each record, and LIFELINES2 INSTANCE TOOLTIP was used more than LIFELINES2 EVENT TOOLTIP, which is a tooltip for each event. After visualizing the tooltip usage pattern in LifeFlow (Figure 6.52), the visualization shows that tooltip for LifeFlow sequence (LIFEFLOW TOOLTIP (*green*)) was often used and followed by tooltip for each record (LIFELINES2 INSTANCE TOOLTIP (*blue*)). After that, users often used LIFEFLOW TOOLTIP (*green*) again, or LIFELINES2 EVENT TOOLTIP (*light blue*). Users rarely used LIFELINES2 EVENT TOOLTIP (*light blue*) immediately after LIFEFLOW TOOLTIP (*green*).

Other than navigational and tooltip actions, other frequent actions were "including and excluding events" (CATEGORY CHANGE CHECKBOX), "including and excluding attributes" (ATTRIBUTE CHANGE CHECKBOX), and "align" (ALIGN).

In terms of selection, users were found selecting top-down from the LifeFlow overview (LIFEFLOW SELECT SEQUENCE and LIFEFLOW SELECT FROM DISTRIBUTION) more than selecting bottom-up from the LifeLines2 view (LIFELINES2 SELECT).

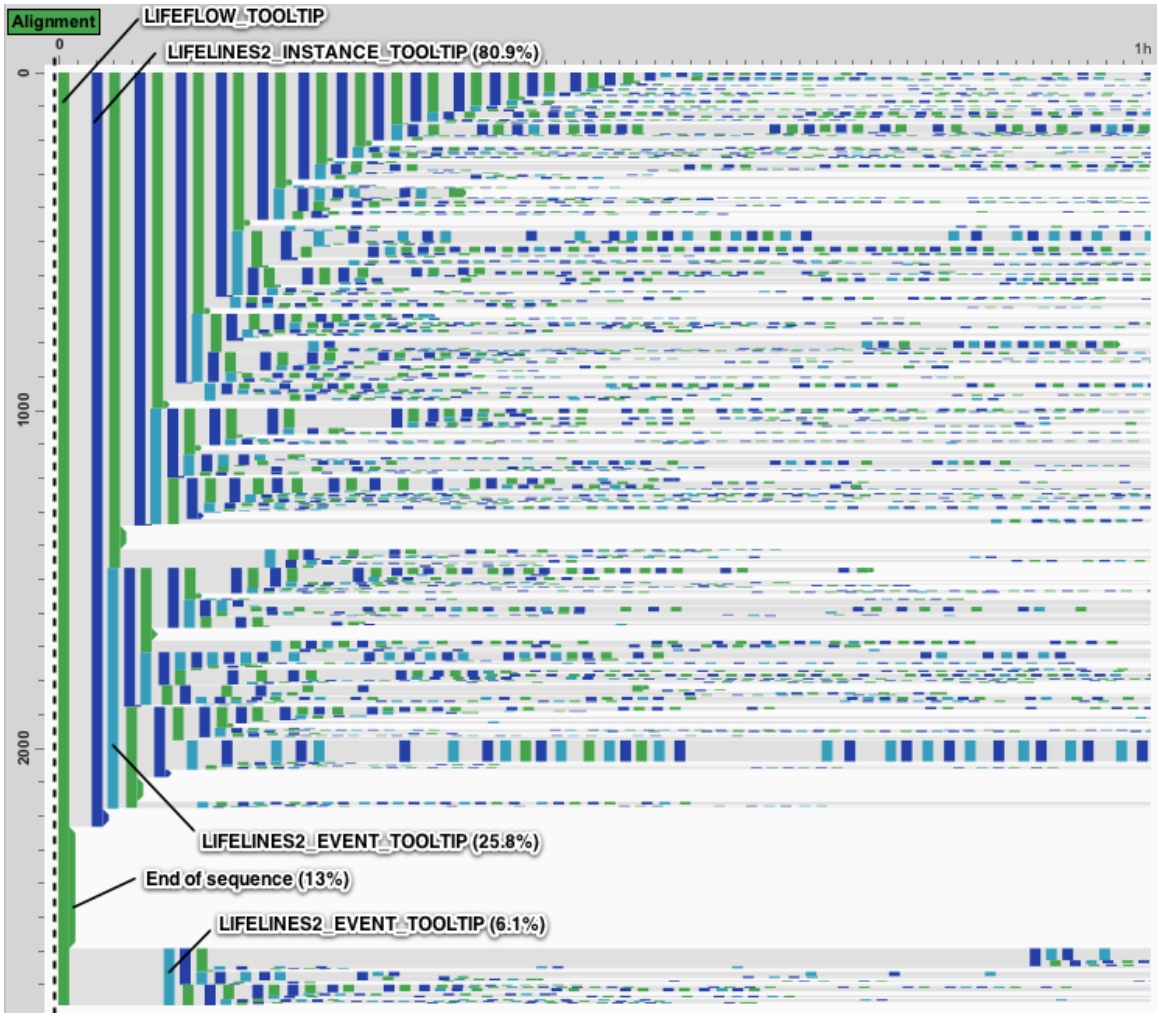

Figure 6.52: Analyzing LifeFlow's tooltip usage with LifeFlow: `LIFEFLOW TOOLTIP` (*green*), a tooltip for LifeFlow sequence, was often used and followed by `LIFELINES2 INSTANCE TOOLTIP` (*blue*), a tooltip for each record. After that, users often used `LIFEFLOW TOOLTIP` (*green*) again, or `LIFELINES2 EVENT TOOLTIP` (*light blue*), which is a tooltip for each event.

6.11 A Process Model for Exploring Event Sequences

Based on usage statistics and observation of user behaviors during the case studies, I have developed a process model for exploring event sequences (Figure 6.53). This process model follows the *sensemaking loop* by Card and others [95, 123], but emphasizes on event sequence analysis. My process model drew inspirations from Thomas and Cook's canonical 4-step process model for analytical reasoning [123] and its extensions [45, 133]. The process model consists of the following steps:

1. Defining goals.
2. Gathering information.
    (a) Preprocessing data
    (b) Cleaning data.
3. Re-representing the information to aid analysis.
4. Manipulating the representation to gain insight.
    (a) Manipulating the representation.
    (b) Exploring results of manipulation.
    (c) Searching for patterns.
    (d) Exploring search results.
    (e) Handling findings.
5. Producing and disseminating results.
    (a) Recording findings.
    (b) Producing results.
    (c) Disseminating results.

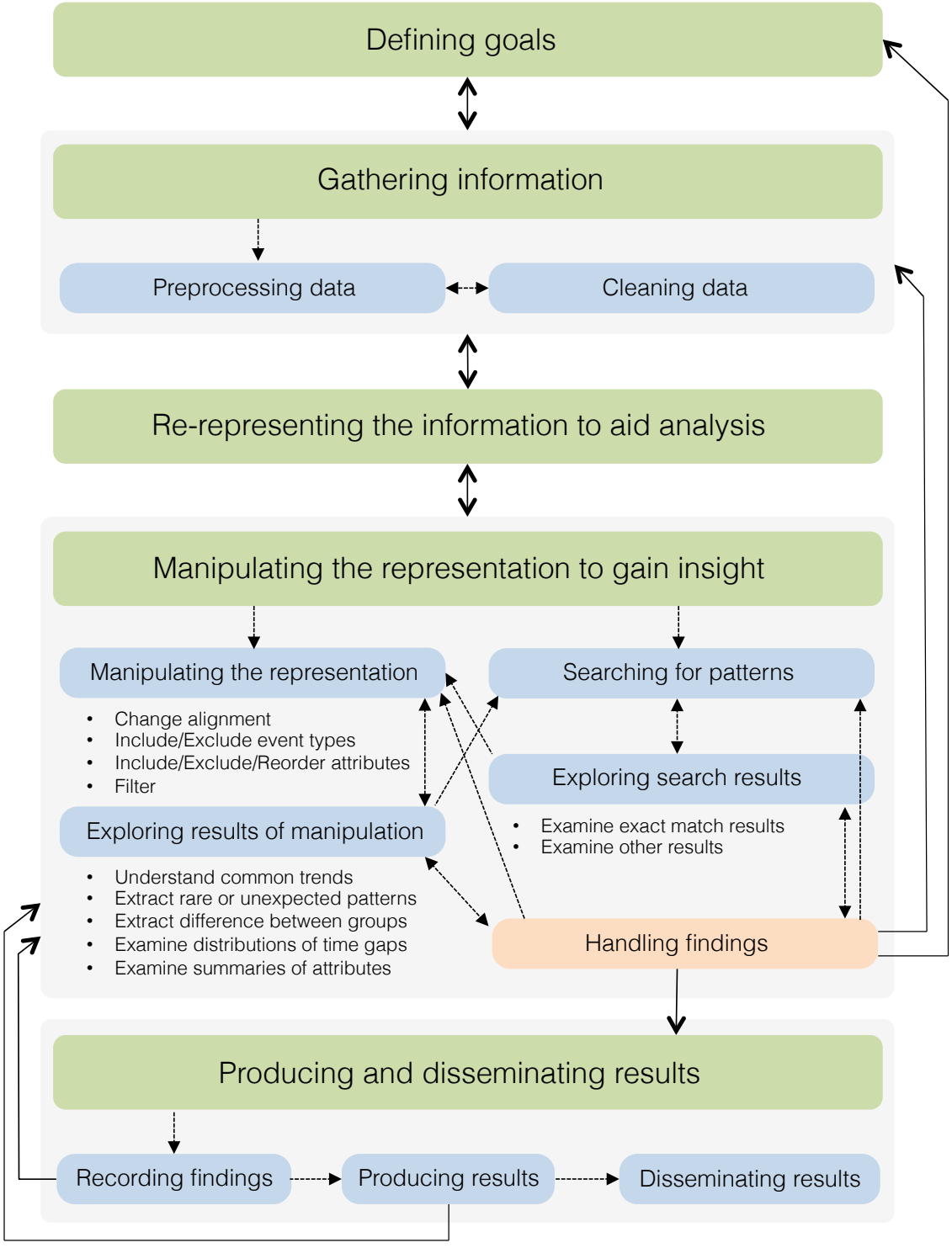

Figure 6.53: A Process Model for Exploring Event Sequences

6.11.1 Defining goals

In the beginning of an analysis, users are advised to define goals or questions that they seek to answer. Although it is not strictly required, having goals will clarify what will be needed in the dataset and guide the appropriate data conversion. For instance, in the medical case studies, the physicians can treat each visit as one record or multiple visits of the same patients as one record. In a patient transfer study, the goal is to learn about common trends of a visit. Treating each visit as one record emphasizes what happens within each visit. On the other hand, in a hospital readmission study, users focus on finding returning patients, so it is more suitable to group multiple visits into one record.

6.11.2 Gathering information

This step consists of two phases: preprocessing raw data into LifeFlow format and cleaning data errors.

6.11.2.1 Preprocessing data

LifeFlow does not connect to databases directly, but requires users to export and convert their data into a custom format. A tool called *DataKitchen* was developed to help users convert their data. However, users still need to understand the data format to fully exploit the benefits of LifeFlow. A dataset in LifeFlow consists of event sequence records. Each record contains events and a set of attributes. Each event consists of a timestamp, an event type and a set of attributes. The main

decisions in this step are defining *record* and *event types*.

- **Record**: Definitions of a record vary according to analysis goals. For example, in the ICDL study, I first defined a record as an entire http session for each user that contains all events from when users enter the ICDL website until they leave the website. However, after realizing that the goal was to understand how people read books, not understand how people navigate the website, I redefined a record as a book reading session instead. Each book reading session contains all events from one user from the time that he/she starts reading one book until he/she stops reading it. One user can have multiple book sessions.

- **Event types**: Although LifeFlow does not limit the number of event types, it is recommended to have less than 15 event types displayed at the same time. Therefore, it is important to define a minimal set of event types. Here are two common strategies.

    1. *Remove irrelevant events from the dataset.* For instance, in the ICDL study, all events that do not represent a page of a book are removed. Users can choose to leave these events in the dataset and include/exclude them during analysis as well. However, removing them will make the dataset more compact and reduce the chances that users will feel overwhelmed because of having too many event types to choose from.

    2. *Group multiple event types into a higher-level event type.* For example, ICU rooms on the 2nd floor (`ICU-2nd`) and ICU rooms on the 3rd floor (`ICU-3rd`) can be grouped into `ICU`. Users can have both high-level

(`ICU`) and low-level (`ICU-2nd`, `ICU-3rd`) event types in the dataset and include/exclude them according to their needs during analysis.

It is common for users to revisit this step again once they become unsatisfied with converted data or an analysis goal is redefined. An early phase of each analysis often consists of trial and error data preprocessings.

6.11.2.2 Cleaning data

After data are converted, users should check for data integrity. From my observations, this is the first thing to look for after loading data into LifeFlow for the first time. Users would look at the visualizations and try to ensure that their data were converted correctly. Some errors due to corrupted timestamps were very obvious. Common errors found in the case studies fall into these two categories:

1. *Illogical sequences*: For example, patients died before they were transferred to ICU (`Die` → `ICU`), or cement trucks started filling cement before reaching construction sites (`StartFillCement` → `EnterSite`).

2. *Unreasonably long gaps*: For example, in the traffic incident study, many incidents were reported to be one hundred years.

However, some errors could be obscured and not discovered until later in the analysis or even left without being noticed. In the soccer study, we found one match that Man U lost to the opponent, but the attribute `Result` for that match was `Win`. A systematic way to check for data integrity that could be further explored is to use integrity constraints in temporal databases as a source of error detection.

6.11.3 Re-representing the information to aid analysis

After users are satisfied with their data, a visual analysis would begin. I recommend users to customize the fundamental visual properties, especially color codings for event types. Choosing appropriate colors can strengthen cognitive process and aid analysis. Three color-coding strategies were used in the case studies:

1. **Distinct color schemes**: Use colors that are distinguishable from each other. It is easier to assign distinct colors to a smaller number of event types. However, not every dataset has a small (≤ 10) number of event types. When there are many event types, it can be quite difficult. Here are a few guidelines for assigning distinct colors:

    - *If possible, associate meanings of colors with the meanings of events.* For example, in the patient transfer study, `ICU` events are encoded in *red* to indicate critical conditions while `Floor` events are encoded in *green* to indicate safe conditions. In the soccer study, red cards and yellow cards are encoded with *red* and *yellow*, respectively.

    - *Use neutral color, e.g. different shades of gray, for less important events.* For instance, in the soccer study, `Kickoff` events are encoded in *gray* to capture less attention.

    - *Reuse the same color if necessary.* Some event types that are always apart can share the same color. For example, in the soccer study (Figure 6.34), `Kickoff` and `Final whistle` events are both encoded in *gray*. In the cement truck study (Figure 6.27), *blue* was used for start loading cement

(`StartLoadCement`) and start filling cement (`StartFillCement`).

- *Use the same color to provide a "bracket" effect.* In the cement truck study (Figure 6.27), *red* was used to indicate arrivals and departures from sites (`ArriveSite` and `LeaveSite`). It creates a visual block, from one *red* bar to another *red* bar, that contains `StartFillCement` and `EndFillCement` events in between.

2. **Sequential color schemes**: This type of scheme is useful for showing continuity when there is an expected sequence of events. For example, in the ICDL study, page numbers are encoded in a gradient from *blue* to *red* to indicate reading direction (Figure 6.18). This strategy can also be applied to binned numerical values, levels of intensity, age ranges, temperature ranges, etc. However, assigning different colors for long sequences (> 10 events) can still be a challenge. Users can assign the same color to consecutive events. For instance, page 27 and 28 in the ICDL study are encoded with the same color.

3. **A combination of both**: An advanced color scheme can combine distinct and sequential colors. As seen in the cement truck study (Figure 6.27), distinct colors are used primarily and sequential colors are used partially to indicate continuity of loading cement (`StartLoadCement` $\rightarrow$ `FinishLoadCement`) and filling cement (`StartFillCement` $\rightarrow$ `EndFillCement`). `StartLoadCement` and `StartFillCement` are encoded with *blue* while `FinishLoadCement` and `EndFillCement` are encoded with *light blue*.

6.11.4 Manipulating the representation to gain insight

Users can apply different visual operators available in LifeFlow to change the visual representation, search for pattern, explore results and gain insight. To guide a systematic analysis, I have included a methodology for event sequence analysis, which was inspired by the 7-step methodology for social network analysis by Perer and Shneiderman [89].

6.11.4.1 Manipulating the representation

Users can use the following visual operators to manipulate the LifeFlow visualization and display different perspectives of the data.

1. **Change alignment**: An extreme approach is to try all event types as alignment points. However, in a more practical approach, users pick only event type x that is interesting to ask what happen before or after x.

2. **Include/Exclude event types**: After each alignment is set, users can include/exclude different combinations of event types to ask what are the relationships between event types e_1, e_2, e_3, ..., e_n. An extreme approach is to try all combinations, which could be very time-consuming. Event types that are irrelevant to an alignment point can totally be excluded. For example, in the ICDL web logs, when a visualization is aligned by the second page of the books, an event type "enter wrong password during log in" seems to be irrelevant.

3. **Include/Exclude/Reorder attributes and use them to group records**:

After an alignment and a set of event types are specified, users may ask if there is any difference between groups of event sequences when they are grouped by attributes a_1, a_2, a_3, ..., a_n?", or what are the proportions of each group. LifeFlow allow users to include/exclude attributes and group the event sequences in different ways. Using multiple attributes in combination and changing the order of attributes used for grouping were found to be very useful during the soccer analysis.

4. **Filter**: During an exploration, users can select and remove uninterested patterns from the dataset to narrow down. This action can be used in between the three actions above. An example of this strategy is demonstrated in the patient transfer study (Section 6.2.4.2).

6.11.4.2 Exploring results of manipulation

After applying the visual operators in Section 6.11.4.1, users examine the updated visualization to extract findings. If nothing interesting is present, users may continue manipulating the representation or switch to searching.

1. **Understand common trends**: Examine both sides of an alignment and try to answer what happen before and after the alignment. Extract the major trends from patterns with high proportion. For instance, in the patient transfer study, most patients came to the ER and were discharged (Figure 6.1). In the cement truck study, most of the event sequences follow a certain procedure (Figure 6.27). In the ICDL study, after aligning by the second page, most of

the users seemed to read in order and did not have any event before the second page (Figure 6.19).

2. **Extract rare or unexpected patterns**: For example, users who read books in backward direction, traffic incidents that last for more than one hundred years, or patients who died before they were transferred to the ICU.

3. **Extract difference between groups**: If one or more attributes are used, find if there is any difference between groups. For example, in the soccer dataset when grouped by `Competition`, the visualization shows that Man U scored the first goal in the English Premier League (EPL) faster than in the UEFA Champions League (UCL) (Figure 6.38).

4. **Examine distributions of time gaps**: Look for interesting gap information (e.g., three goals occurred on the first minute) or irregularities (e.g., a skewed distribution of Agency C's clearance time in the traffic incident study). When there are many gaps, the rank-by-feature panel (Figure 3.14) can be used to recommend interesting distributions.

5. **Examine summaries of record and event attributes**: Look for frequent record attributes and event attributes. For example, find top goalscorers by looking at frequency of event attribute `Player` of event `Score` (Figure 6.45).

6.11.4.3 Searching for patterns

According to the case studies, there were a few situations that trigger a search.

1. During exploration, users find an interesting pattern . For example, in the hospital readmission study, my collaborator saw an interesting pattern (`Registration` → `Discharge` → `Registration` → `Die`) from one patient record and searched for other patients with that pattern (Section 6.5.6.6).

2. Once users are familiar with and gained better understandings of their datasets from manipulating the representation, they may question existences of particular scenarios. As reported in the soccer study, my collaborator asked towards the end of the session if there was any match in which Man U missed a penalty and then conceded a goal (Section 6.9.4.4).

3. It is also possible that users are already familiar with their datasets and can start an analysis by testing their hypotheses about existences of particular patterns.

6.11.4.4 Exploring search results

After the search is executed, users can explore search results to learn and gain new insight about the patterns that they have just searched for. Similarly, if nothing interesting is present, users may continue searching or switch to manipulating the representation.

1. **Examine exact match results**: See if a specified pattern exists in a dataset. Count the number of results. Do the results match users' expectations?

2. **Examine other results**: If the exact match results are empty, examine other

results to confirm that the specified pattern does not exist or check if there is any similar pattern. Even if the exact match results are not empty, there could be interesting similar patterns. For instance, my collaborator searched for a match that Man U received a red card and then conceded a goal. The exact match results are empty. However, the other results show a match that Man U received a red card and then scored—to the surprise of both of us (Figure 6.50).

6.11.4.5 Handling findings

Each finding from an exploration (Section 6.11.4.2 and 6.11.4.4) falls into one of these three categories: *positive* (e.g., one that helps users answer their questions), *negative* (e.g., one that tells them the answer they seek cannot be answered) and *unexpected* (e.g., an unanticipated characteristic of the data is found and led to new questions) [133]. During this step, users may also redefine their analysis goals according to their findings.

When users reach a positive finding, they would try to use their domain expertise to comprehend and explain the finding. If the observation sounds and provides interesting insights, they would record the results for dissemination. For example, in the ICDL study, we found sequences in which visitors accessed previous pages before continuing to the next pages, such as $3 \rightarrow 4 \rightarrow 3 \rightarrow 4 \rightarrow 5$. Using an understanding of how a person normally read a book, we implied that the sequences reflect behaviors of people who flipped back to a previous page to catch up some contents before

continuing (Section 6.6.4.3). Sometimes a positive finding also introduces new questions that lead to further analysis. For instance, in the soccer study, after noticing that Man U often scored first (Figure 6.43), an immediate follow-up question was how fast did they usually score? This encouraged us to analyze the distribution and found that the three fastest goals occurred on the first minute (Figure 6.44).

On the other hand, a negative finding could indicate a roadblock. In many cases, this roadblock can be passed with more data, or better preprocessing and cleaning, which will push users back to the information gathering step. For example, a physician wanted to find common diagnoses for patients' first visits (Section 6.5.6.7). However, the diagnoses were very detailed—such as "Chest Pain Nos", "Chest Pain Nec"—and, therefore, could not be grouped to show common trends until these diagnoses are preprocessed into higher level categories. Some negative findings are dead-ends which cannot be continued. Some are due to insufficient data or limitations of LifeFlow. In the hospital readmission study, the discharge timestamps were not guaranteed to be accurate, therefore preventing any meaningful interpretation of the time from registration to discharge. The software limitations might be resolved by adding new features or delegating further analysis to external tools. For example, I was asked if LifeFlow can produce a tabular report that summarizes number of patients according to number of visits (1,000 patients visited once, 200 patients visited twice, and so on). LifeFlow can provide number of patients via tooltips, but does not provide a table that contains all the numbers. This feature can be added or users can use an external tool to perform this task.

An unexpected finding is a wildcard that could lead to discovery. Similar to

a positive finding, if an observation sounds and provides interesting insight, users would record the results for dissemination. For instance, we found that some visitors accessed books on ICDL website in backward direction (Section 6.6.4.4). Unfortunately, many of these unanticipated findings are just illogical sequences, which indicate data errors and push users back to the information gathering step.

6.11.5 Producing and disseminating results

Once users reach their analysis goals, the last step will be preparing their findings to share with colleagues. I separate the work in this step into three phases: recording finding, producing results and disseminating results.

6.11.5.1 Recording findings

This phase occurs immediately after users handle findings and find interesting insight. Users record important information that will be refined into the final results. After that, if an analysis is not completed yet, users will go back to manipulate the representation to gain more insight. There are a few common activities in this phase.

- *Record findings*: Users jot down the findings, e.g., "In all matches that Man U won in the UCL, they scored the first goals" or "Patients with at least six visits, which is 2% of patients, account for 4.3% of visits."

- *Record provenance*: Users record an analysis route that took them to their findings. The provenance is very important because it allows users to reproduce the findings and explain their reasoning processes.

- *Capture screenshots*: Screenshots are taken and sometimes annotated.

- *Export data*: In some cases, users also export the final dataset for further analysis in external tools. For instance, in the patient transfer study, the physician can export medical record numbers of the bounce back patients to examine their medical histories in more detail.

6.11.5.2 Producing results

After all analyses have been completed, users compile all raw findings and include supplemental information from external sources into final results. If necessary, users can reproduce findings to create better screenshots, capture more detailed information, etc. Users may also gain more insight while reproducing results.

6.11.5.3 Disseminating results

When LifeFlow users have extracted their insights, the screenshots can be annotated (graphically and textually), captions can be added and a report generated as a slide presentation, document, or video. Live presentations to groups and email dissemination can reach stakeholders who will make decisions about taking action or refining the analysis.

6.11.6 Summary

The visual analytics process model described in this section is designed to guide users in learning how to explore event sequences and help designers improve future

visual analytic tools. This iterative process promotes insight discovery through multiple steps that involve data understanding, data cleaning, choosing representations, hypotheses generation, hypotheses validation and extraction of evidence to share with colleagues. Currently, this process is conceptual and not yet implemented in the user interface, but could be in the future.

6.12 Design Recommendations

The case studies, usage statistics and observations have revealed the strengths and weaknesses of LifeFlow. I have noticed many interesting user behaviors and generalized these into the a list of recommendations for designing a user interface for exploring event sequences.

1. **Use Align-Rank-Filter (ARF) framework**: The ARF framework [131, 132] was originally developed for a list of multiple event sequences. However, applications of this concept are not limited to the list view only. Alignment works well with the LifeFlow overview and has proven to be one of the most useful features in the case studies. Changing the reference point often opens up new perspectives and triggers new questions from a new angle. Ranking offers a way to reorder and organize visual components to gain new insight, such as ranking the LifeFlow sequences by time instead of cardinality. Filtering allows users to trim the dataset and focus on a subset of data. The filtering mechanisms interact directly with an underlying data model and, therefore, can operate independently from any visual components.

2. **Provide multiple levels of information that can be accessed in an organized manner**: Event sequence information can be grouped into three levels. Users should be able to drill down to a lower level from a higher level.

   (a) *Event-level*: Information specific to each event, such as time or event type. This is the lowest level.

   (b) *Record-level*: Information specific to each record, such as ID, and aggregations of event-level information within each record, such as number of events in each event type or sequence of events.

   (c) *Overview-level*: Aggregations of record-level information, such as number of male patients, and aggregations of event-level information across multiple records, such as a distribution of events over time or a frequency table of all sequences of events. This is the highest level.

3. **Provide multiple overviews from different aspects**: An overview usually requires data aggregation that sacrifices some aspects of the data. Therefore, it is very unlikely that only one overview will be enough to summarize all aspects of the data. For example, LifeFlow can provide a summary of sequences and gaps between sequences, but it cannot show the temporal aspects as well as a histogram, e.g., how often a patient visited in January 2010. To this end, I believe that having multiple overviews will complement each other and provide a better big picture.

4. **Use coordinated multiple views that are loosely coupled**: The ben-

efits of having multiple views is when using them in combination, they can complement each other. This can be achieved through *brushing and linking*. In addition, not every task will require all views at the same time. Therefore, each view should be independent and can function without knowing or requiring the existence of any other view, following the *Model-view-controller (MVC)* paradigm. This way, each view can be hidden or displayed as needed, to better utilize screen space.

5. **Incorporate non-temporal attributes**: It is challenging to gain insight from temporal information alone. Including other attributes in the analysis, such as patients' genders, physicians' names, can provide a better understanding of the data. These attributes sometimes can help explain the cause and effect relationships between events.

6. **Support rapid inclusion/exclusion of event types and grouping event types into hierarchy**: As evidenced in the case studies, an ability to include and exclude events rapidly allows quick explorations of subsets of the data without going back to another round of preprocessing. Several users expressed also their needs to dynamically change the granularity of the event types. For example, `ICU` can be split into `ICU-2ndFloor` and `ICU-3rdFloor`. In the beginning of an analysis, users want to treat both event types as `ICU`, but in the later stage, they want them to be separated.

7. **Provide search**: Search may often be a forgotten feature in the beginning of an analysis, but once users have identified an interesting pattern, it becomes

a valuable feature to have. Users can come back with the same query a month later or use the same query in another dataset.

8. **Support data preprocessing and cleaning**: Data preprocessing and cleaning are inevitably parts of the exploratory data analysis. Many visualization systems delegate these tasks to external tools [58, 57]. However, by including some simple tasks in the user interface, it can reduce the gap between preprocessing and analysis and facilitate a smoother and more continuous process.

9. **Keep history and handle mistakes**: An exploratory data analysis consists of many iterations. Without proper support from the user interface, users will have hard time remembering what has been done to the data. The final results should not be only insight, but also *analytic provenance*—a user's reasoning process through the study of their interactions with visualizations [94, 42]. Therefore, the user interface can help by storing history. Also, every once in a while, users will make a mistake. In my research prototype, "Revert to original" was found to be a successful feature because it provides a safe point for users to backtrack to, without reloading the dataset. However, being able to undo will prevent them from starting over from the beginning and save more time.

6.13 Summary

As much as it is difficult to judge a book by its cover, it is difficult to evaluate the benefits of a rich interactive visualization and user interface in a short period of

time, especially in a controlled environment. This chapter describes my attempts to engage with the users, learn and share the understandings of their data and problems, and use the innovations from my research to tackle their problems in a new way. Aiming to satisfy users' needs drives my research towards practical use in addition to its theoretical contributions. Many limitations from tiny design issues to requirements for new useful features were identified. Many benefits were confirmed and new use cases keep being discovered. These case studies provide benefits both to my dissertation research and to the satisfaction of the users, many of which are dealing with critical problems in healthcare. By helping them improve their processes and save more lives in the future, my research is becoming a small part in changing the world into a better place. My experience from these case studies also led to a process model and a set of design recommendations for temporal event sequence exploration that will be useful for future designers.

Chapter 7

Conclusions and Future Directions

7.1 Conclusions

Temporal event sequence analysis is an important task in many domains: medical researchers may study the patterns of transfers within the hospital for quality control; transportation experts may study traffic incident logs to identify best practices. While previous research has focused on searching and browsing these event sequences, overview tasks and exploratory search are often overlooked. This dissertation describes four contributions that provide new ways to interactively explore temporal event sequences and describe recommendations for future designers. All of which are summarized as follows:

1. **Overview visualization**: Lengthy reports often have an executive summary to provide an overview of the report. Unfortunately, there was no executive summary to provide an overview for event sequences. Therefore, the first contribution of my dissertation is developing an interactive visualization called *LifeFlow*—which includes data aggregation, visual representation and interaction techniques—to provide an overview of millions of event sequences in a million pixels. By seeing this overview, users can grasp the big picture of the dataset and often start asking questions about their data that they have never thought of before. A user study with ten participants confirmed that even

novice users with 15 minutes of training were able to learn to use LifeFlow and rapidly answer questions about the prevalence of interesting sequences, find anomalies, and gain significant insight from the data. Several MILCs illustrate its benefits in supporting event sequence analysis in real world problems.

2. **Similarity search interface**: In an exploratory search, users can be uncertain about what they are looking for and try to find event sequences that are similar to the query. I have developed the *M&M measure*, a similarity measure, and *Similan*, a similarity search interface, to support this uncertain scenario.

   The first version allows users to search for event sequences that are similar to selected target event sequence. A user study showed promising direction but also identified room for improvement. The second version addresses the limitations from the first version. The user interface allows users to draw an event sequence example and search for records that are similar to the example, with improved search results visualization. The similarity measure was also sped up and could be customized by more criteria, increasing its performance and flexibility.

   After that, a controlled experiment was conducted to assess the benefits of exact match and similarity search interfaces for five tasks, leading to future directions for designing query interfaces that combine the benefits of both interfaces.

3. **Hybrid search interface**: Searching with an exact match interface, users may not know what they have missed, but they have confidence in the result sets and can count the number of records easily. Searching with a similarity search interface, users can see all records sorted by similarity but have difficulty deciding how many records should really be accepted to the result sets. The *Flexible Temporal Search (FTS)* was then developed to combine the benefits from both interfaces and provide a more powerful search, so that the users do not have to choose between the two previous interfaces and make trade-offs. FTS allows users to see a clear cut-off point between records that pass the query and records that do not, making it easier to count and giving more confidence. All records are sorted by similarity, allowing users to see the "just-off" cases that are right beyond the cut-off point. FTS was found useful in medical scenarios and was often used in the case studies once the interesting patterns were identified from the LifeFlow overview.

4. **Case Study Evaluations, a Process Model and Design Guidelines**: To evaluate the benefits of this research outside of laboratory environment, I had recruited domain experts as users and applied these new techniques to tackle their data analysis problems or offer new ways to analyze their data. While this dissertation work was initially motivated from medical scenarios, the variety of applications shown in the case studies demonstrate its generalizability to many domains where event sequence is the main focus. The experience and lessons from these studies also led to a process model for exploring event sequences

and a set of design recommendations for future visualization designers.

7.2 Future Directions

One goal of this dissertation was to explore this research area and open up new directions for future researchers. Throughout my PhD study, I have encountered many limitations in my approach and discovered many new frontiers that I believe are interesting future directions. Some of which are more specific while others are more open-ended. I summarize and describe them in this section.

7.2.1 Improving the overview visualization

Although the evaluations demonstrated many benefits of LifeFlow in diverse applications, it also identified limitations to the design. Improvements would be welcome for:

1. **Representing event types**: LifeFlow uses color to represent event types. This approach has limitations when there are many event types. It is very difficult to choose different colors for more than 10 categories. Current workarounds are using many shades of each color to create variations or use same colors for many event types, but it requires users to carefully pick the colors and does not always solve the problem. However, there are also benefits of using color. It creates pre-attentive effect, allowing users to notice some event types easily. Using colors instead of text labels also make the visualization compact and clean. Future researchers who find a better way to represent numerous event

types, will facilitate still wider usage.

2. **Displaying very long event sequences**: The current visualization needs a better way to handle datasets with very long sequences, i.e., sequences with a large number of events. Each bar in LifeFlow is assigned a fixed width. The gaps between the bars are used to represent time. When displaying the overview of the entire dataset before any zooming or filtering, it should be able to display everything. When the sequences are long, the gaps between bars are compressed to fit all the bars on the screen. However, in the worst case, the sequence of bars still cannot fit on the screen even when all the gaps are zero pixel. New designs might be needed to support these very long sequences.

3. **Alignment**: When the alignment is set, users often get confused that the sequences on the same vertical position on the left and right side are the same records. Actually, they are not, and the two sides (before and after the alignment) are independent. Even expert users sometimes forget this fact.

4. **Displaying gaps between two non-consecutive events in the sequence**: The current design can display a sequence, such as `A` → `B` → `C`, and gaps between consecutive events (`A` → `B` and `B` → `C`). However, it does not show the gap between non-consecutive events (`A` → `C`). The horizontal distance between `A` and `C` are the sum of the mean/median time between `A` → `B` and `B` → `C`, which may be not equal to `A` → `C`. I addressed this limitation by adding an interaction (measurement tool) that allows the users to measure the time from `A` to `C`. However, many users did not realize in the beginning that horizontal

width of an entire sequence does not represent the time from the first event to the last event in the sequence. Future design needs to emphasize this limitation to avoid misunderstanding. Moreover, there might be a way to include gaps between non-consecutive events in the visual display.

7.2.2 Improving the search

The FTS has evolved through many design iterations—beginning from the similarity search in Similan. I have received many suggestions from users and colleagues along the way and used them to improve the design. However, there are still many improvements that can be made.

1. **Clustering results**: Instead of showing *other results* as a list of records. It could be useful to cluster those records by the reasons that they are excluded from *exact match results*. This way, records that fail similar constraints will be grouped together. By providing a visualization of these clusters, users can quickly understand an overview of the results.

2. **Caching intermediate results to speed up the search**: Currently, the query has to be re-executed every time users adjust weights, which makes it computationally expensive. It would be more efficient to cache intermediate results that can be reused to avoid unnecessary computation. The challenge is designing what could be stored to save the computation.

3. **Weights**: The FTS user interface allows users to adjust weights and customize the similarity measure. However, this freedom to adjust the knobs

could also be challenging for users. Previously in Similan, a few experimental weight presets were implemented and found promising. Developing a set of common weight presets could help users avoid adjusting the weights manually. Furthermore, a user study that focuses on weight adjustment could provide a better understanding of this issue and help designers learn how to design a better user interface.

4. **Including attributes in the query**: The FTS focuses only on the temporal aspect of the event sequences, by looking only at the order, existence and timing of events. However, there is also non-temporal aspect of the event sequences. They have attributes that should be included in a more completed query. The attributes include *record attributes*, such as patient's gender, and *event attributes*, such as room number or diagnosis for each visit. Including these attributes in the query will introduce new challenges to both the similarity measure and query specification.

7.2.3 Supporting more complex data

Event sequence data also have different complexity in each dataset. These following characteristics require further research to support them:

1. **Many similar patterns**: The current tree building algorithm groups the sequences by their prefixes based on the idea that order of occurrence is important. However, in some datasets, there are many similar sequences that should have been grouped together to reduce the complexity of the visualiza-

tion, but are not grouped because they have different prefixes. For example, `A` → `B` → `C` → `D` and `A` → `B` → `D`. Two interesting questions are (1) how to determine what should be grouped together, and (2) what should be displayed on the visualization?

2. **Streaming data**: The techniques in this dissertation were originally developed for static dataset. However, these techniques might be extended to support streams of real-time event sequences.

3. **Interval-based events**: My research focuses on point events, where each event represents a point in time. However, some events are not just points in time but have intervals. For example, drug usage contains a start and end time. This increases the complexity of both the search and visualization to handle new types of relationships, such as overlap, contain, etc. New approaches have to be developed to address these new challenges.

4. **Multiple concurrent activities**: In some cases, a process can be viewed as a combination of multiple concurrent activities. For example, in the hospital, patients with difficulty breathing were provided different breathing devices (e.g, `CPAP` → `BIPAP` → `Intubation`) throughout the treatment. At the same time, they were transferred between rooms in the hospital (`ER` → `ICU` → `Floor`). This process, as a result, consist of two concurrent event sequences: change of devices and room transfers. Each event sequence can be analyzed individually using the work in this dissertation. However, these two activities are connected and can provide more complete information when analyzed

together, and therefore should not be separated. There are opportunities for new techniques to handle multiple concurrent event sequences.

5. **Concurrent event sequences and numerical time series**: Some processes may not only consist of multiple activities but also include changes to numerical variables over time. For example, the patients were transferred between rooms while the level of blood pressure and heart rate were monitored. Could future researchers develop techniques to help users understand relationships between all these event sequences and numerical time series from multiple patients?

7.2.4 Supporting new tasks

Many exploratory tasks in analyzing event sequences were identified during the MILCs. Many were supported and included in this dissertation work. However, there were also some tasks that are beyond the scope of this dissertation but show potential future directions.

1. **Comparison**: Comparison is one of the interesting tasks that should be explored further. The aggregated data from LifeFlow, a tree of sequences, could be used to compare performance of one hospital to another, or performance of the same hospital in different periods. Using LifeFlow, users can inspect manually to see the difference between two visualizations, but it is still difficult to see the changes clearly. I believe that there are research opportunities for supporting comparisons between multiple tree of sequences, or other trees

with values in nodes and edges, in general.

2. **Ranking LifeFlow by Feature**: An extreme way to explore a dataset is to try all combinations to include/exclude event types and attributes, which could lead to a large number of LifeFlow visualizations. Analyzing all of them can be very exhaustive and time-consuming. Providing a rank-by-feature framework that can suggest interesting LifeFlow visualizations will be very helpful for the analysis. The main challenge is defining ranking criteria.

3. **Associating outcomes with sequences of events** Many event sequences have associated *outcomes*. For example, outcomes for medical records could be measured by cost, mortality or discharge rates. Connecting sequences of events to their associated outcomes can help data analysts discover how certain sequences of events may lead to better or worse outcomes.

4. **Investigating the cause and effect relationships between events**: The work in this dissertation treat all events in an event sequence as *observations*, or *effects*, and try to show an overview of the effects. However, some event types could also be considered as *factors* that may cause the effects. Such factors, such as the administration of a drug to a sick patient, or a soccer player receiving a red card (which leaves his/her team short-handed), can often change the course of subsequent events. Understanding how these factors influenced the following subsequent events is another important task. Users should have the options to treat some event types differently as factors and the visualization should help users detect factors that may influence the effects.

7.2.5 Scalability and Performance

My research focuses on the design of the visualizations and user interface, which are the front-end components. However, the ideal visual exploration tool also need to be able to handle large datasets effectively and provide responsive feedback independent of the size of the dataset, and that requires more work in the back-end components.

1. **In-memory issue**: For portability and ease of deployment, the working prototype was designed to load all data into memory. However, this design choice limits the size of the largest datasets to available memory. A scalable design should hold only necessary data in the memory and store the rest of the data in the secondary storage via databases or other techniques.

2. **Database**: Use of database can resolve the in-memory issue and also improve the performance when the database is properly indexed. A carefully-designed data model, indexing and query optimization can increase the performance of the event sequences query and other operations in the back-end of the visualization.

3. **Rendering**: Often, when dealing with large datasets, the user interface becomes less responsive. This is partly because of the data processing, but another reason is also the rendering. It is possible that a dataset with millions of records can lead to millions of objects on the screen. Rendering these many objects is a very heavy task. In addition, these objects need to be tracked for event handling and user interactions.

4. **Cloud computing**: Another rising trend in recent years is the rise of cloud computing and MapReduce clusters. Many time-consuming tasks, such as data processing, query and rendering, can be accelerated by parallelizing and offloading the heavylifting to the cloud. Supporting a gigantic dataset in a responsive manner is a very challenging task, but cloud computing seems promising to make this task possible.

7.3 Summary

This chapter summarizes all results and contributions from this dissertation. Each contribution was supported with evaluation results that demonstrate its benefits. I believe that this dissertation has refreshed the way people think about analyzing multiple event sequences and shows the impact of using event sequences for understanding many aspects of people's lives from book reading behaviors to hospital procedures and more. I have also combined many lessons and feedback learned from my users and colleagues into a list of interesting future directions, which could lead to challenging and fruitful research projects. I believe that there are still many promising opportunities for making sense of these fascinating event sequences, waiting for the research communities to explore.

Appendix A

Examples of Temporal Event Sequences

To exemplify how my research can be used beyond the case studies mentioned in this dissertation, help prospective users decide whether their datasets are applicable or not, and inspire future researchers about possible datasets, I have compiled a list of additional event sequence data that I see the possibilities of exploring them with the work in this dissertation. [1]

1. *Human activities*: Record your daily activities for a period of time and use the data to reflect on yourself. For example, what is the average time gap between when the alarm clock rang and the time that you actually woke up? How long do you take a shower in the morning?

2. *Bus stops*: Keep track when the buses arrive at each bus stop. Each record is one bus trip. By analyzing these data, user can track the buses' performance or find bottlenecks on the route.

3. *Web logs*: Each record can be a user or an IP address, and contains all the pages visited.This can be used to analyze the order that visitors visited pages on websites. How long did they spend on the pages? Which page did they visit next?

[1] Note: Each dataset may require additional cleaning and preprocessing.

4. *Usability logs*: Analyze how users use an application or user interface. See what features do users often use or what features are often used after certain actions?

5. *U.S. Bill status*: The Library of Congress keep history of the U.S. Bills under a system called *Thomas* (http://thomas.loc.gov/). A bill history usually follows this pattern: `propose` → `vote` → `pass/decline/amend` → etc. Each bill can be declined and revised many times, leading to different patterns.

6. *Sports events*: In the case study, I demonstrate how to analyze soccer events. Similar analyses can be performed on football, hockey, baseball, and other sports.

7. *Role-playing game (RPG) characters evolution*: In role-playing game, such as World of Warcraft or Ragnarok (Figure A.1), characters can evolve by advancing to higher job classes, for example, `novice` → `thief` → `assassin`. Players have to collect experience points or complete requirements to achieve the promotion. In game administrators' and developers' perspective, it might be interesting to learn which jobs the players often chose or how long it took for each job transition, so they can adjust the difficulty and make the game more balanced.

8. *Bug tracking*: Project management and bug/issue tracking software, such as *TRAC*, are widely used in many companies. Users can report bugs by creating tickets. Each ticket contains time when it was reported, accepted, completed,

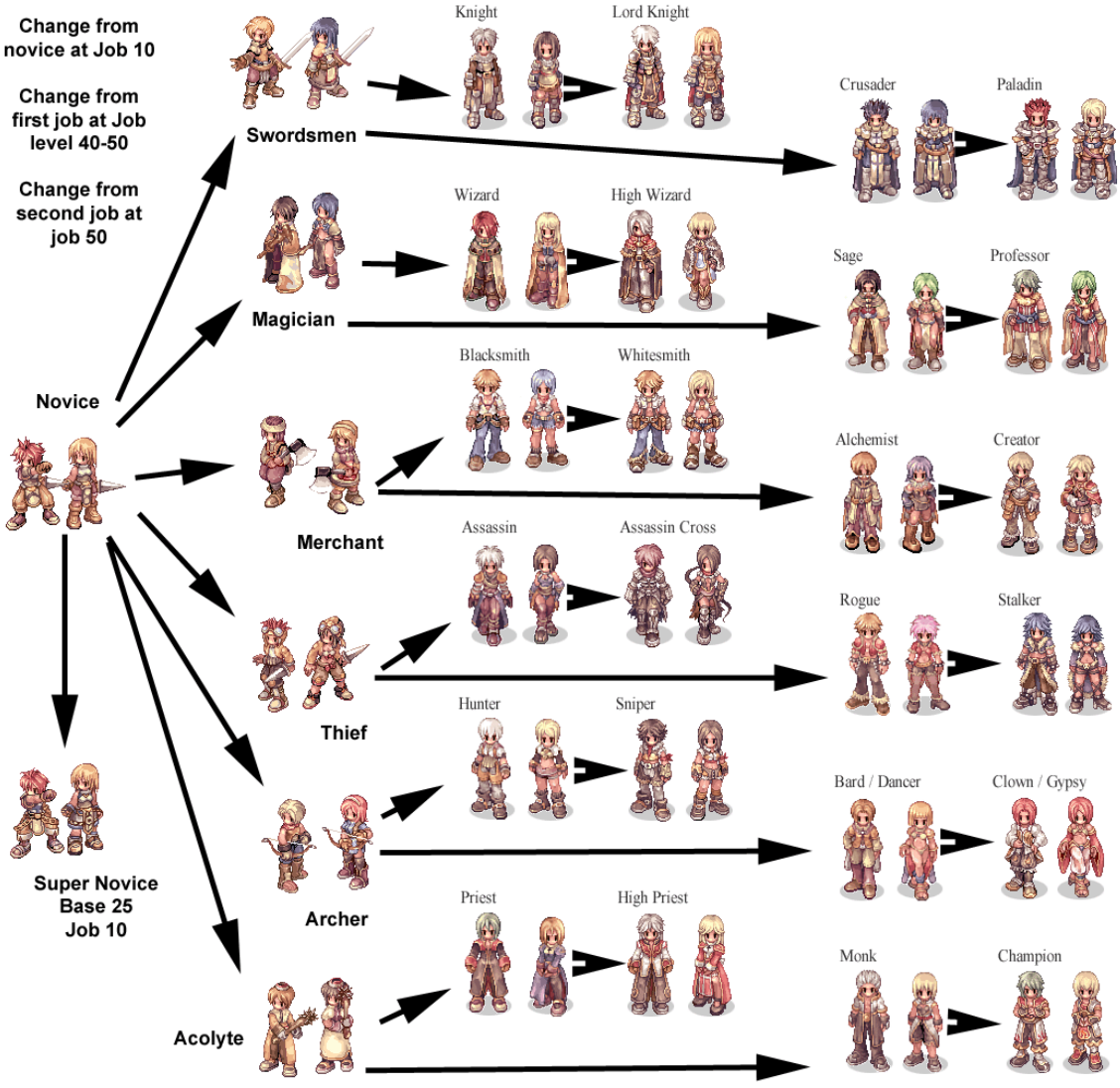

Figure A.1: Ragnarok job tree

reopened, etc. Users can also indicate the importance level of each ticket or assign a ticket to a particular developer. These can be used as attributes.

9. *Package tracking*: Millions of packages are being delivered everyday by carriers such as FedEx, UPS or DHL. How many of these packages reach the final destination? How many of these packages are delivered by the guaranteed time? How often they have to try the second or third delivery attempts?

10. *Journal paper review process*: Publishing in a journal often requires many rounds of revising and reviewing. A paper submission could lead to an immediate acceptance, a request for a revision, or a rejection. The revision also follows the same process again. The publishers might be interested in analyzing the peer review process to see how many papers were rejected? How long does it take to publish? What are the common paths? Is there any steps that take too much time and can be accelerated to reduce submission-to-publication time?

11. *Medical check-up*: Each person has to go through several stations in check-up process, for example, blood pressure checking, chest x-rays, pulmonary function testing, blood test, urinalysis, etc. By tracking the time spent in each stations and waiting time, the hospital administrators can learn how to improve the check-up process and customer's experience.

12. *Researchers' publications*: A publication can be viewed as an event, which can be categoried using the type of the venue (journal, conference, workshop,

ldots), type of publication (full paper, short paper, note, *ldots*), authorship (first author, second author, *ldots*), and other attributes. The event sequences of publications could be used to review researchers' performance.

13. *Student progress*: Academic institutions keep track of when students take classes, propose their dissertation topics, publish papers and graduate. These academic histories can reflect the quality of education.

14. *Manufacturing process*: For many manufacturing processes, there are certain procedure that needs to be followed. Each step in the process can be tracked and recorded. Production managers then can review the current process and find room for improvement.

15. *Actions on social networks*: Billions of users are using the social network, such as *Facebook* or *Twitter*. What are they doing? Is there any interesting pattern among the random activities? Can we learn their behaviors from their actions?

16. *Recurring attendees of an event*: There are many annual conferences, symposiums, workshops, meeting, etc. Event organizers may wonder how many attendees keep coming back? The organizers can keep records of their attendees to answer this question. Each record represents an attendee and contains all events that he/she attended. For example, Mr. Brown attended CHI'08, CHI'10, CHI'11 and CHI'12 while Miss Scarlet attended CHI'06, CHI'07, CHI'08, CHI'10 and CHI'11.

17. *StarCraft build orders*: Many of the industry's journalists have praised Star-

Craft as one of the best and most important real-time strategy video games of all time. It is particularly popular in South Korea, where players and teams participate in professional competitions, earn sponsorships, and compete in televised tournaments. A build order is an early game strategy that is designed to optimize economy and timing. It indicates an order in which buildings and units should be produced to maximize efficiency. Expert players can record their games and parse the build orders from their recorded replays. Using LifeFlow, these players can find the steps in build order that they often made mistake to improve their skills.

Appendix B

LifeFlow Software Implementation

B.1 Overview

LifeFlow is implemented in Java. Its runtime requirements are machines with Java Runtime Environment (JRE) 1.6 or above, with at least 1024x768 screen resolution. It has been tested on Mac OS X 10.5-10.7, Windows XP and 7, and Ubuntu Linux.

As of March 20, 2012, the LifeFlow SVN repository contains 61,188 lines of code from 867 revisions. This appendix describes its input data format, software architecture and implementation overview.

B.2 Input Data Format

To simplify deployment and reduce adoption overhead, LifeFlow does not require any database installation and accepts text files as input data. It uses two input data files and one config file:

- (Required) *Event data file (*.txt)* contains all events and event attributes.
- (Optional) *Attribute data file (*.attrib)* contains record attributes.
- (Optional) *Config file (*.xml)* keeps the preferred settings of visualization.

Table B.1: Sample event data file (sample_data.txt)

Record ID	Event type	Date and Time	Event attributes
4318	Arrival	2010-01-19 03:53:00	Entrance-Diagnosis="Back pain"
4318	ER	2010-01-19 04:14:00	
4318	ICU	2010-01-19 04:49:00	Room No.="125"
4318	Floor	2010-01-19 16:12:00	Room No.="854"
4318	Exit-DISC	2010-01-20 06:44:00	Physician="Dr. X"; Exit-Diagnosis="Back pain"
5397	Arrival	2010-01-19 06:53:00	Entrance-Diagnosis="Heartburn"
5397	ER	2010-01-19 07:20:00	
5397	Floor	2010-01-19 07:46:00	Room No.="854"
5397	Exit-DISC	2010-01-20 09:44:00	Physician="Dr. J"; Exit-Diagnosis="Heartburn"

B.2.1 Event Data File

An event data file is a 4-column tab-delimited text file (Table B.1).

- First column is record ID.
- Second column is event type.
- Third column is date and time in `YYYY-MM-DD HH:MM:SS` format.
- Fourth column contains event attributes in the format:

```
attribute-name1="attribute-value1" ; attribute-name2="attribute-value2" ; ...
```

Use ; to separate between attributes. Leave this column blank if there is not any event attribute.

B.2.2 Attribute Data File (optional)

An attribute data file stores non-temporal attributes. This file is not required if there is not any record attribute. It is a 3-column tab-delimited text file (Table B.2), which has the same name as the event data file but with extension *.attrib, instead of *.txt.

Table B.2: Sample attribute data file (sample_data.attrib)

Record ID	Attribute name	Attribute value
4318	Gender	Male
5397	Gender	Female

- First column is record ID.
- Second column is attribute name.
- Third column is attribute value.

B.2.3 Config File (optional)

A config file helps users save their customized settings, such as colors of events, and reuse them in the next analysis. It is an XML file, which is self-descriptive.

B.3 Design

B.3.1 Software Architecture

LifeFlow architecture follows the *Model-View-Controller (MVC)* paradigm. In the implementation, I adopted *PureMVC*, a free open source MVC Framework (available from `http://puremvc.org`).

In PureMVC, each view is wrapped by a *mediator* while each model is wrapped by a *proxy*. The mediator and proxy provide PureMVC functionalities while leaving the original view and model independent from PureMVC. The central Controller is called *facade*, which acts like a mothership that knows about all views and models. The facade let all views and models communicate without knowing each other. The

facade also maintains a list of *commands* that can be triggered by notifications, such as loading dataset or exit. Commands handle actions that involve multiple components or actions that should not be handled by mediators or proxies.

Each view is independent and do not know that other views exist. However, when users perform an action in one view, that action can trigger changes in other views although the views do not know each other. To achieve this, these components send notifications. When a notification is issued, the facade (i.e. the mothership) will notify all mediators and commands that are registered to receive that type of notifications. Each view has to declare types of notifications that it will handle.

In the ideal MVC, no views and models will know about each other. In LifeFlow, I relaxed this concept a bit by allowing a mediator (view) to know about proxy (model). A reference to the proxy of dataset is stored in each mediator. This allows direct calls to the dataset proxy without passing messages through the facade.

B.3.2 Code Organization

The code are organized to separate classes that are reusable and classes that are specific to this software.

1. **Reusable**: Most of the classes in LifeFlow are designed to be indepedent and reusable. More generic classes, such as time, counter, popup menu or color palettes, are split into package `edu.umd.cs.hcil.frozenlife`. Classes that are more specific to the work in this dissertation are organized in the following packages:

(a) *Data structures*: Events, records and tree of sequences are contained in the package `edu.umd.cs.hcil.lifeflow.datastructure`.

(b) *I/O*: Package `edu.umd.cs.hcil.lifeflow.io` contains classes that handle importing/exporting data from/to text files.

(c) *ARF*: Package `edu.umd.cs.hcil.lifeflow.arf` contains classes that provide different kind of alignment, ranking and filtering for the Align-Rank-Filters framework.

(d) *Similarity measure*: The similarity measure used in FTS and related classes are contained in the package `edu.umd.cs.hcil.lifeflow.similarity`.

(e) *User interface components*: Package `edu.umd.cs.hicl.lifeflow.components` contains LifeFlow, LifeLines2, FTS and other visual components.

2. **Application-specific**: Classes that are specific to LifeFlow and cannot be reused are contained in the package `edu.umd.cs.hcil.lifeflow.mvc`. These classes make use of the reusable components and combine them into the LifeFlow software.

   (a) Package `edu.cs.umd.hcil.lifeflow.mvc.baseclasses` contains custom extensions to the PureMVC classes.

   (b) *Models* are in the package `edu.cs.umd.hcil.lifeflow.mvc.model`.

   (c) *Views* are in the package `edu.cs.umd.hcil.lifeflow.mvc.view`.

   (d) *Commands* are in the package `edu.cs.umd.hcil.lifeflow.mvc.command`.

B.4 Data Structures

B.4.1 Time

These two classes are often used to represent time in LifeFlow:

1. *AbsoluteTime*: Real time, such as Jan 12, 2012. This class wraps Java's standard `Calendar` class. This object can be converted to number of milliseconds since January 1, 1970, 00:00:00 GMT, and vice versa.

2. *RelativeTime*: After an alignment is set, the time is represented as time relative to the alignment time, such as 1 month or -2 years. Relative time can be positive (after alignment time), negative (before alignment time) or zero (at alignment time). This object can be converted to number of milliseconds after alignment time, and vice versa.

These two classes also provide many useful methods for manipulating time. Both classes extend the `ITime` interface.

B.4.2 Fundamental Data Structures

These are fundamental building blocks in LifeFlow. All classes are contained in the package `edu.umd.cs.hcil.lifeflow.datastructure.tc`. "TC" stands for "Temporal Categorical".

- *Event*: A tuple of time and event type, for example, (August 17, 2008, Arrival to a hospital). Time is stored as a number of milliseconds since January 1, 1970, 00:00:00 GMT. An immutable class `TCEvent` represents an event.

- *Event Attribute*: Property of an event. For example, attending physician and diagnosis are attributes of each visit (Arrival event). Event attributes are stored in a map (keys,values) in each `TCEvent` object using attribute names as keys and attribute values as values.

- *Record*: One entity of temporal event sequence, for example, a patient. Class `TCRecord` represents an event. `TCRecord` contains a unique ID and multiple lists of `TCEvent`: one list that contains all events and one list per event type. We separate this to optimize filtering events by event types. All `TCEvent` in these lists are sorted by time.

- *Record Attribute*: Property of a record that is not associated with time, for example, gender and age. Record attributes are stored in a map (keys, values) in each `TCRecord` object using attribute names as keys and attribute values as values.

- *Instance*: Each copy of record that is shown on the visualization. Class `TCInstance` represents an instance. It wraps a record and stores an alignment time, a flag that indicates if the instance is selected. One record usually have only one instance with an alignment time at zero, meaning that an alignment point is not specified. One record can also have multiple instances. For example, patient no. 10002 has three *Floor* events. After aligning this record using **all** `Floor` events, LifeFlow will duplicate the record into three instances of this record, each with an alignment point at one of the *Floor* events. Many data structures in LifeFlow are extended from `TCInstance` using the decorator

design pattern.

- *Group of Instances*: Multiple instances of the same record are grouped together in an `TCInstanceGroup` object. Both `TCInstance` and `TCInstanceGroup` implements an interface `ITCInstance`.

- *List of Instances*: Class `ITCInstanceList` is a list of `ITCInstance`, which can be an instance (`TCInstance` or its extensions) or a group of instances (`TCInstanceGroup`). It keeps track of selected records and instances and provides methods for rapid selection of records and instances.

B.4.3 Handling Dataset

Each dataset is loaded from text files into a datastructure called `RawDataSet`. `RawDataset` consists of a mapping from category into colors (`CategoryLookUpTable`) and an immutable array of all records.

In the beginning of an analysis, when users select a dataset and click on "Explore", a new working dataset (`WorkingDataSet`) is created from the selected raw dataset `RawDataSet`. A working dataset contains lists of instances and parameters for the visualizations. Each `TCRecord` from the raw dataset is converted into an `TCInstance` and added to a list of instances. This list is called the "original list" (`originalList`). A "working list" (`workingList`) is copied from the original list and used for all operations. When users revert the dataset, a working list is cleared and copied from the original list again.

A new working dataset can also be created from another working dataset. In

this case, the original list in the new working dataset is copied from the current working list in the old working dataset.

Class `WorkingDataSet` provides methods for manipulating dataset, such as align, filter or select. Class `WorkingDataSetProxy` wraps the `WorkingDataSet` to provide the same functionalities and send MVC notifications when the data are changed.

Class `CoreModel` maintains a list of all working datasets that are loaded into the system. It is wrapped into a `CoreModelProxy` and displayed as a list of all available datasets in the user interface that allow users to choose from.

B.5 Main Components

B.5.1 Main class

Main class is contained in the package `edu.umd.cs.hcil.lifeflow.mvc`. During startup, class `LifeFlowStarter` initializes LifeFlow application by creating an `AppFacade`. The `AppFacade` then instantiates all MVC components and open a `MainWindow`. The `MainWindow` contains a menubar and `MainPanel`, which contains all the panels.

B.5.2 LifeFlow

B.5.2.1 Tree of Sequences

Class `TreeOfSequences` is the backbone of LifeFlow visualization. It aggregates multiple event sequences into a tree that is later rendered as a LifeFlow visualization. `TreeOfSequenceBuilder2` takes a list of instances and builds a tree of sequences from it. User can choose to create a positive tree or a negative tree by setting a parameter. It iterates through each instance and add the instance to existing nodes if possible, or add new nodes to the tree if necessary and add the instance to the new nodes. Each node in the tree (`AbstractTreeNode`) contains markers (`EventMarker`) that can backtrack to all instances and events that are contained in this node. It also provides methods for finding average time between nodes.

B.5.2.2 Flexible Geometry

I define a coordinate system that can change according to horizontal zoom factor and vertical zoom factor to support semantic zooming in LifeFlow component. A value in the FlexibleGeometry is called a `FlexibleValue`, which is defined by two values and one factor. This can be described by the following equation:

$$\text{FlexibleValue } v = fixedValue + variableValue * multiplyFactor$$

$$v = v.f + v.v * v.F$$

When two `FlexibleValue` objects share the same multiply factor, they can

perform add and subtract operations by adding and substracting their fixed values and variable values.

$$\begin{aligned} v + w &= (v.f + w.f) + (v.v + w.v) * F \\ v - w &= (v.f - w.f) + (v.v - w.v) * F \end{aligned}$$

It also supports multiplication by a scalar value s.

$$v * s = (v.f * s) + (v.v * s) * F$$

A point in the FlexibleGeometry (`FlexiblePoint`) is a tuple of two `FlexibleValue` objects: x and y. A rectangle in the FlexibleGeometry (`FlexibleRectangle`) contains four `FlexibleValue` objects: x, y, width and height.

B.5.2.3 LifeFlow Component

Class `JLifeFlow2` is the LifeFlow visual component. It is a `JComponent` and can be added to Swing containers. The number "2" is for its version. (I have rewritten the entire component once to clean up the spaghetti code from the first version.) `JLifeFlow` uses *Piccolo2D*, a scene graph library for Java. Rendering a LifeFlow visualization consists of the following steps:

1. *Preprocessing*: All instances are preprocessed to remove excluded event types.
2. *Building tree*: Convert a list of preprocessed instances into two trees of sequences: positive tree and negative tree.

3. *Plotting*: All nodes and edges in the trees are converted into drafts of visual objects. To support semantic zooming, bounds of a draft are defined using *FlexibleGeometry*.

   For x position and width, the multiply factor is `pixelPerTime`, variable value is time in milliseconds and fixed value is a fixed value in pixels. For example,

   $$\text{Node.width} = fixedWidth + time * pixelPerTime$$

   For y position and height, the multiplication factor is `pixelPerInstance`, variable value is number of instances and fixed value is a fixed value in pixels. For example,

   $$\text{Node.height} = fixedHeight + numberOfInstances * pixelPerInstance$$

4. *Pinpointing*: Calculate the offset of the visualization and multiplication factors (`pixelPerTime` and `pixelPerInstance`) according to component size, zoom level and offset of the viewport.

5. *Painting*: Calculate actual dimensions of the visual objects from computed drafts and zoom factors in the previous steps, create these visual objects and add them to the Piccolo canvas and bind them with event handlers. Only objects that are within bounds and visible are created.

6. *Painting selection*: Render highlight to show selection.

This pipeline can be triggered from any step and all following steps after that will be executed. For example, resizing a window will trigger pinpointing because the size of the window is change and will affect the zoom factor and offset of the visualization. After the pinpointing is completed, the visualization and selections are painted.

B.5.3 LifeLines2

I implement LifeLines2 component in LifeFlow using visual design from the original LifeLines2 software, but did not reuse its code. Class `JLifeLines2` is the new component. `JLifeLines2` groups instances of the same records together and allows users to collapse or expand the group as needed. It uses an *ItemRenderer* to render each component. Future developers can write their custom renderers and replace the default renderer with theirs. I include two renderers in this software distribution: one used for default display and another for instances with similarity scores. `JLifeLines2` also provides custom events and event handler interface. The rendering process consists of the following steps:

1. *Creating draft*: Create a draft for each instance in the list. It is also used to keep the state of `TCInstanceGroup` whether it is currently expanded by user or not.

2. *Sort*: Sort the drafts according to specified criteria.

3. *Update position*: Calculate position on the screen for each draft.

4. *Plot time*: Plot timeline according to component size and zoom factor.

5. *Paint*: Draw only objects that are within display bounds and visible, add them to the Piccolo canvas and bind event handlers.

For convenience, I implement class `JLifeLines2Panel` that wraps `JLifeLines2` and includes a scrollbar and header.

B.5.4 Flexible Temporal Search (FTS)

B.5.4.1 Similarity search

A `SimilarityQuery` object contains query specification. A query consists of multiple query elements (`IQueryElement`). When a similarity search is executed, an engine (`SimilarityEngine`) is created for the given `SimilarityQuery`. This `SimilarityEngine` contains a table for dynamic programming, which can be reused for multiple matchings. The engine performs the similarity search using the given `SimilarityQuery` on a given list of `ITCInstance` and return results (`SimilaritySearchResult`). The results contain two lists of instances that are exact match results and other results. Each instance in these two lists is an `ITCInstanceWithSimilarityScore`. It wraps the original ITCInstance that was passed to the engined and adds a similarity score following the decorator design pattern.

B.5.4.2 User Interface

Class `SearchPanel` is the FTS user interface. It consists of two main parts:

1. *Query Panel*: This part is a component called `QueryPanel`. It has its own data model (`QueryModel`) in the back and can convert from this model to `SimilarityQuery`, or vice versa. It handles users' interaction to add `Glyph` or `Gap` to the model. Glyphs include all events and negations. The `QueryPanel` utilizes `QueryRenderer` for rendering different types of glyphs and gaps.

2. *Results Panel*: This panel consists of exact match results and other results. Both parts are `JLifeLines2` components with item renderer set to a renderer for displaying an instance with similarity score. When right-click on any instance and select "Show comparison", it will initialize a `ComparisonDialog` and pass the selected `ITCInstanceWithSimilarityScore` to show the comparison. The time scale of these two `JLifeLines2` components are bound together.

Appendix C

Spin-off: Outflow

C.1 Introduction

This appendix describes a spin-off project during my internship with the Healthcare Transformation group at IBM T.J. Watson Research Center. I worked with Dr. David Gotz to develop an alternative visualization for exploring flow, factors, and outcomes of temporal event sequences. This visualization, , called *Outflow* [141], aggregates multiple event sequences in a new way and includes summaries of factors and associated outcomes.

Many event sequences have associated *outcomes*. For example, outcomes for EMR data could be measured by cost, mortality or discharge rates. For sports applications the outcome could be a win, loss or draw. Analyzing common patterns, or *pathways*, in a set of event sequences can help people better understand aggregate event progression behavior. In addition, connecting these pathways to their associated outcomes can help data analysts discover how certain progression paths may lead to better or worse results.

For example, consider a medical dataset containing information about a set of patients with a dangerous disease. Each patient is described by an outcome measurement (e.g., if they survived the disease or not) and an event sequence containing the date that certain symptoms were first observed by a doctor. An analysis of path-

ways in such a dataset might lead to the discovery that patients with symptoms A, B and C were likely to die in the hospital while patients with symptoms A, B and D were more likely to recover.

Similarly, consider a dataset representing a set of soccer matches where goals scored are events and wins are considered a good outcome. An analysis of pathways in this dataset could help answer questions such as, "Does a team win more often when it comes from behind?"

While analyzing event sequence data as described above can help answer many questions, there are often external *factors*—beyond the set of event types that define an event sequence—that make an analysis even more complex. Such factors, such as the administration of a drug to a sick patient, or a soccer player receiving a red card (which leaves his/her team short-handed), can often change the course of subsequent events. These factors must be incorporated into an analysis to understand how they influence outcomes.

Finally, event collections can be massive. Major healthcare institutions have millions of medical records containing millions of event sequences and many different event types. The scale and variability of this problem can lead to an extremely complex set of pathways for many scenarios. For example, even for a small data set with just five event types and where each event sequence has just five events, there are $3,125$ (5^5) possible pathways. This vast amount of information can be overwhelming and makes these datasets difficult to analyze.

To address these challenges, I have designed *Outflow*, an interactive visualization that combines multiple event sequences and their outcomes into a graph-based

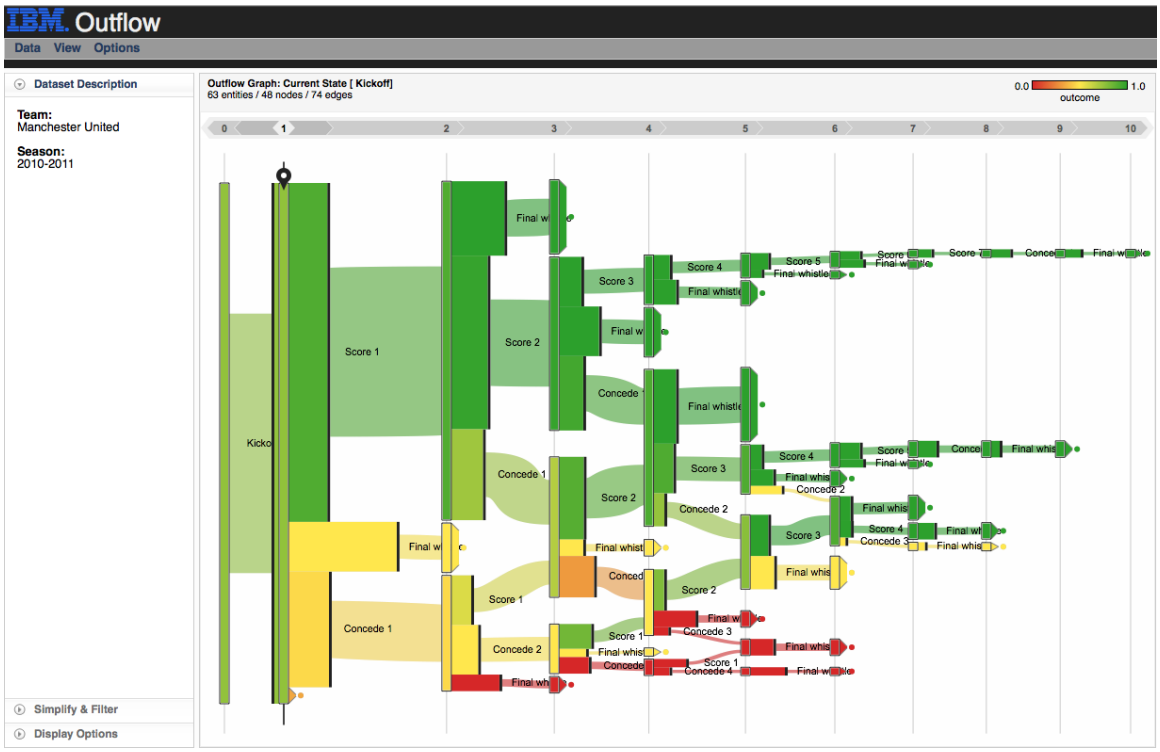

Figure C.1: Outflow processes temporal event data and visualizes aggregate event progression pathways together with associated statistics (e.g. outcome, duration, and cardinality). Users can interactively explore the paths via which entities arrive and depart various states. This screenshot shows a visualization of Manchester United's 2010-2011 soccer season. Green shows pathways with good outcomes (i.e., wins) while red shows pathways with bad outcomes (i.e., losses).

visual representation. Outflow can summarize collections of event sequences and display all pathways, time gaps between each step, and their associated outcomes. Users can interact with the visualization through direct manipulation techniques (e.g., selection and brushing) and a series of control widgets. The interactions allow users to explore the data in search of insights, including information about which factors correlate most strongly with specific pathways.

This appendix describes the Outflow visualization in detail by explaining key design, rendering process and interaction techniques. I illustrate the generalizability of Outflow by discussing two applications (a medical use case and a sports statistics use case) and demonstrate its power via two example analyses and a user study.

The remainder of this appendix is organized as follows: Section C.2 presents two motivating applications. The Outflow design is discussed in Section C.3. I then report the evaluation results in Section C.4 and C.5, and summarize in Section C.6.

C.2 Motivation

Outflow provides a general solution for a class of event sequence analysis problems. This section describes two examples from different application domains which served as motivating problems for this work.

C.2.1 Congestive Heart Failure (CHF)

Outflow was originally inspired by a problem faced by a team of cardiologists. They were working to better understand disease evolution patterns using data from

a cohort of patients at risk of developing *congestive heart failure (CHF)*. CHF is generally defined as the inability of the heart to supply sufficient blood flow to meet the needs of the body. CHF is a common, costly, and potentially deadly condition that afflicts roughly 2% of adults in developed countries with rates growing to 6-10% for those over 65 years of age [80]. The disease is difficult to manage and no system of diagnostic criteria has been universally accepted as the gold standard.

One commonly used system comes from the *Framingham study* [79]. This system requires the simultaneous presence of at least two major symptoms (e.g., S3 gallop, Acute pulmonary edema, Cardiomegaly) or one major symptom in conjunction with two minor symptoms (e.g., Nocturnal cough, Pleural effusion, Hepatomegaly). In total, 18 distinct Framingham symptoms have been defined.

While these symptoms are used regularly to diagnose CHF, my medical collaborators are interested in understanding how the various symptoms and their order of onset correlate with patient outcome. To examine this problem, I were given access to an anonymized dataset of 6,328 patient records. Each patient record includes timestamped entries for each time a patient was diagnosed with a Framingham symptom. For example:

Patient#1:(27 Jul 2009, Ankle edema), (14 Aug 2009, Pleural effusion), ...

Patient#2:(17 May 2002, S3 gallop), (1 Feb 2003, Cardiomegaly), ...

The dataset also contains information about medication orders and patient metadata. Available metadata includes date of birth, gender, date of CHF diagnosis, and (when applicable) date of death.

In line with the use of Framingham symptoms for diagnosis, I assume that once a symptom has been observed it applies perpetually. I therefore filter the event sequences for each patient to select only the first occurrence of a given symptom type. The filtered event sequences describe the *flow* for each patient through different disease states. For example, a filtered event sequence ***symptom A*** → ***symptom B*** indicates that the patient's flow is ***no symptom*** → ***symptom A*** → ***symptoms A and B***. I used the presence (or lack thereof) of a date of death as an outcome measure (dead or alive).

An inspirational task was to examine aggregated statistics for the flows of many patients to find common disease progression paths. In addition, I wanted to discover any correlations between these paths and either (1) patient outcomes (i.e. mortality) or (2) external factors (i.e. medications).

C.2.2 Soccer Result Analysis

Although originally inspired by the medical application outlined above, Outflow itself is not domain specific and can generalize to other application areas. To demonstrate the broad applicability of this work, Outflow is also used to analyze soccer match results. For example, Figure C.1 shows an Outflow visualization of the 2010-2011 season for Manchester United Football Club (Man U.), an English Premier League soccer club based in Old Trafford, Greater Manchester. Man U. has won the most trophies in English soccer and is one of the wealthiest and most widely supported teams in the world.

The 2010–2011 season was another successful one for Man U. in which they won multiple trophies. To better understand their route to success, I collected data from all 61 matches that Man U. played that season. For both Man U. and their opponents, I captured time-stamped events for every kickoff, every goal scored, and every final whistle. I also recorded the outcome for every match (win, loss, or draw) along with timestamped records of every yellow and red card received.

As in the healthcare case, events are cumulative. Each time a goal is scored, it is added to the scoreline with the goal tally increasing over the course of a match.

Using Outflow, sports analysts are able to see the average outcome associated with each scoreline, find the average time to the next goal, predict what is likely to occur next, and understand how non-goal factors, such as red cards, can impact how a match may progress. This use case is the one tested in an evaluation where users were asked to perform many of these tasks. The study design and results are described in detail in Section C.5.

C.3 Description of the Visualization

Outflow's design consists of four key elements: data aggregation, visual encoding, the rendering process, and user interaction. This core design is then expanded to support two important features: *simplification* and *factors*.

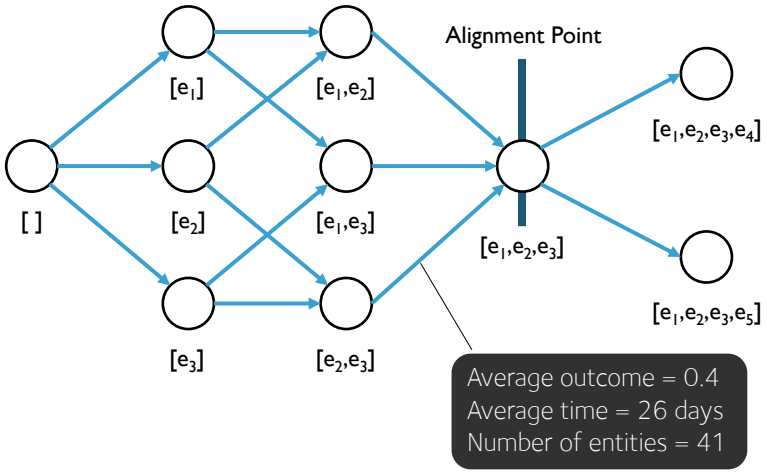

Figure C.2: Multiple temporal event sequences are aggregated into a representation called an *Outflow graph*. This structure is a directed acyclic graph (DAG) that captures the various event sequences that led to the alignment point and all the sequences that occurred after the alignment point. Aggregate statistics are then anchored to the graph to describe specific subsets of the data.

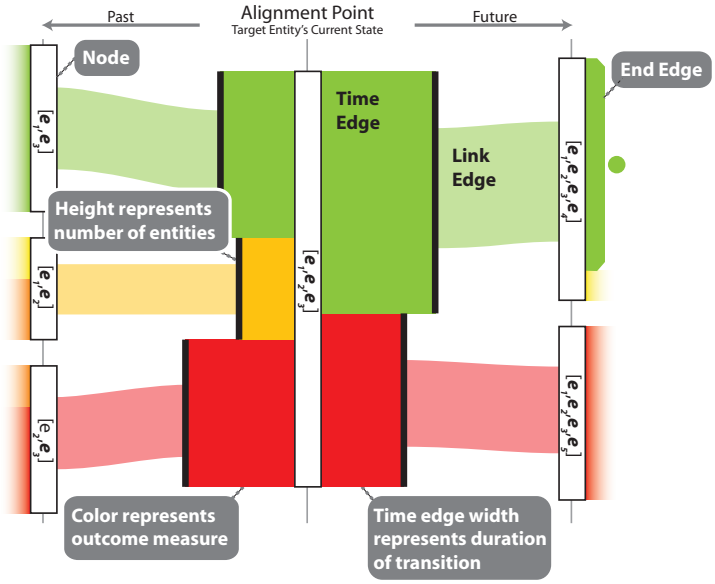

Figure C.3: Outflow visually encodes nodes in the Outflow graph using vertical rectangles. Edges are represented using two distinct visual marks: time edges and link edges. Color is used to encode average outcome.

C.3.1 Data Aggregation

The first step in creating an Outflow visualization is data aggregation. I define an entity E (e.g., a patient record) as a timestamped (t_i) progression through different states (S_i). Each S_i is defined as a set of zero or more events (e_j) that an entity has experienced at or before time t_i. A transition from state S_m to state S_n is denoted by $T_{m \to n}$.

$$
\begin{aligned}
E &= (S_0, t_0) \to (S_1, t_1) \to (S_2, t_2) \to (S_3, t_3) \to \ldots \to (S_n, t_n) \\
S_i &= [e_1, e_2, e_3, \ldots, e_i]
\end{aligned}
$$

Given a collection of entities, $\{E\}$, I begin by choosing one state (which must be experienced by all $E \in \{E\}$) as an *alignment point*. For example, users can align a set of medical records around a state where all patients have the same three symptoms (and no other symptoms). After choosing an alignment point,[1] I aggregate all entities that pass through the alignment point into a data structure called an *Outflow graph* (Figure C.2).

An Outflow graph is a state diagram expressed as a directed acyclic graph (DAG). A node is added to the graph for each unique state observed in $\{E\}$ (e.g., a node for each unique combination of co-occurring symptoms). Edges are used to represent each state transitions observed in $\{E\}$, and they are annotated with various statistics: the number of entities that make the corresponding transition,

[1]The system uses S_0 (the state where no events have occurred) as the default if no other alignment point is specified.

the average outcome for these entities, and the average time gap between the states.

Therefore, an Outflow graph captures all event paths in $\{E\}$ that lead to the alignment point and all event paths that occur after the alignment point. In the medical analysis example, users can select a target patient from the database and use the target patient's current state as the alignment point. This approach allows for the analysis of historical data when considering the possible future progression of symptoms for the selected target patient. In the soccer analysis, users can align by a state with a specific score (e.g., 2-1), which would include only matches that, at some point in time during the game, reached the specified state (e.g., two "Score" events and one "Concede" event). This could be useful for prediction by looking at historic data.

C.3.2 Visual Encoding

Based on the information contained in the Outflow graph, I have designed a rich visual encoding that displays (1) the time gap for each state change, (2) the cardinality of entities in each state and state transition, and (3) the average outcome for each state and transition. Drawing in part on prior work from FlowMap [93] and LifeFlow [142], I developed the visual encoding shown in Figure C.3.

Node (State): Each state is represented by a rectangle whose height is proportional to the number of entities.

Layer: A graph is sliced vertically into layers. Layer i contains all states with i events. The layers are sorted from left to right, showing information from the past

to the future. For example, in Figure C.1, the first layer (layer 0) contains only one node, which represents all records before any event. The next layer (layer 1) also has one node because all games begin with a "Kick off" event. Layer 2, however, has three nodes because each game evolved in one of three different ways: "Score", "Concede", or "Final whistle".

Edge (Transition): Each state transition is displayed using two visual marks: a *time edge* and a *link edge*. Time edges are rectangles whose width is proportional to the average time gap of the transition and height is proportional to the number of entities. Link edges connect nodes and time edges to convey sequentiality.

End Edge: Each entity can end in a different state. A trapezoid followed by a circle marks these end points. Like transition edges, the height of the trapezoid is proportional to the number of entities that end at the corresponding state. The circles are included to ensure that small end edges remain visible.

Color-coding: Colors assigned to edges are used to encode the average outcome for the corresponding set of entities. The outcome values are normalized to the [0,1] range with higher values representing better outcomes. The normalized values are then mapped to a color scale. In this prototype, the color coding scales linearly from red (outcome of 0) to yellow (outcome of 0.5) to green (outcome of 1). The color scale can be adjusted to accommodate users with color vision deficiency.

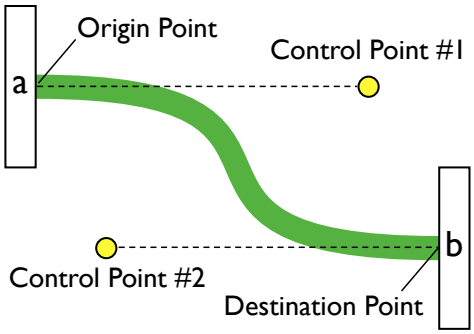

Figure C.4: Link edges are rendered using quadratic Bézier curves. Control point placement is selected to ensure horizontal starting and ending edge slopes.

C.3.3 Rendering

Graphs with many nodes and edges can be difficult to visualize due to possible edge crossings and overlapping visual marks. I apply several rendering techniques to emphasize connectivity and reduce clutter in the visualization.

C.3.3.1 Bézier Curve

Each link edge is rendered as a quadratic Bézier curve to emphasize the connectivity and flow of the paths in the visual display. I make the control line from the origin point to the first control point perpendicular to the origin node, and the control line from the destination point to the second control point perpendicular to the destination node. As shown in Figure C.4, this ensures that the edges are horizontal at both the start and end.

C.3.3.2 Sugiyama's Heuristics

Outflow initially sorts nodes and edges in each layer according to their outcomes. However, this often leads to unnecessarily complex visualizations because of

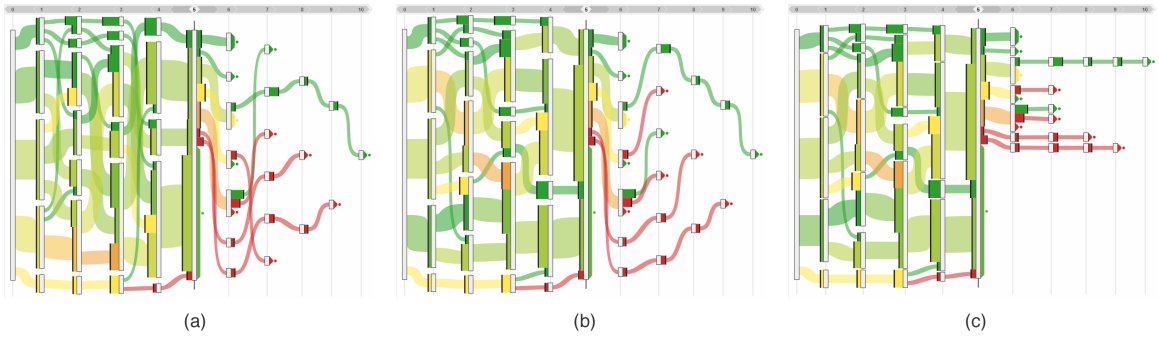

Figure C.5: A multi-stage rendering process improves legibility. (a) Initial layout after sorting edges by outcome. (b) After applying Sugiyama's heuristics to reduce crossings. (c) The final visualization after both Sugiyama's heuristics and Outflow's force-directed layout algorithm to obtain straighter edges.

edge crossings. Therefore, I apply Sugiyama's heuristics [119], a well-known graph layout algorithm for DAGs, to reorder the elements in each layer to reduce edge crossings. Figures C.5a and C.5b show example layouts of the same data before and after applying Sugiyama's heuristics, respectively.

C.3.3.3 Force-directed Layout

Once the order of nodes in a layer has been determined, the next step is to calculate the actual vertical position for each node. Positions are initially assigned by distributing the nodes equally along a layer's vertical axis. However, this approach can often result in unnecessarily curvy paths (as shown in Figure C.5b) that make the visualization more difficult to follow.

To produce straighter paths, I apply a spring-based force-directed optimization algorithm [35] that reduces edge curvature. As illustrated in Figure C.6, nodes and edges are simulated as particles and springs, respectively. To prevent nodes from getting too close to each other, spring repulsions are also inserted between nodes.

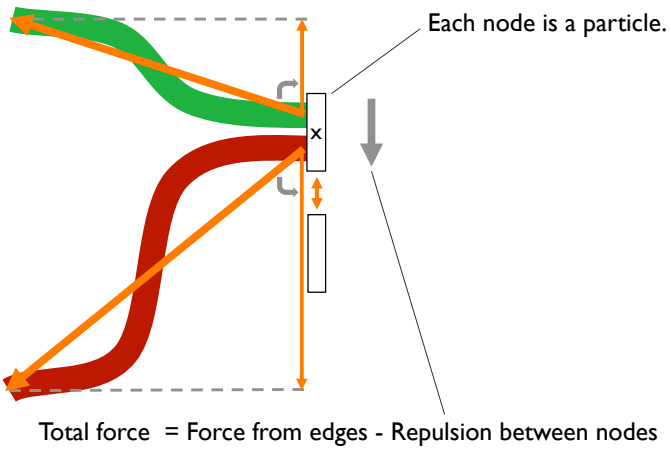

Figure C.6: A spring-based optimization algorithm is used to obtain straighter (and easier to read) edges. Nodes and edges are simulated as particles and springs, respectively. Spring repulsions are inserted between nodes. During the optimization, nodes are gradually moved along a vertical axis to reduce the spring tension in their edges.

Through a series of iterations, nodes are gradually moved along the layer's vertical axis to reduce the spring tensions. The optimization stops either when the entire spring system is stable or when a maximum number of iterations has been reached. Figure C.5c shows the more readable visualization with straightened edges obtained by applying this force-directed layout algorithm.

C.3.3.4 Edge Routing

When large time differences exist between two alternative paths, time edges and link edges can overlap as shown in Figure C.7a. This can make it more difficult for users to trace paths through the visualization during an analysis. I resolve this issue by routing link edges through an intermediate point that avoids crossings (Figure C.7b). The intermediate point is calculated by (1) finding the longest time edge from n neighbor edges in the direction traveled by a link edge (up/down), and

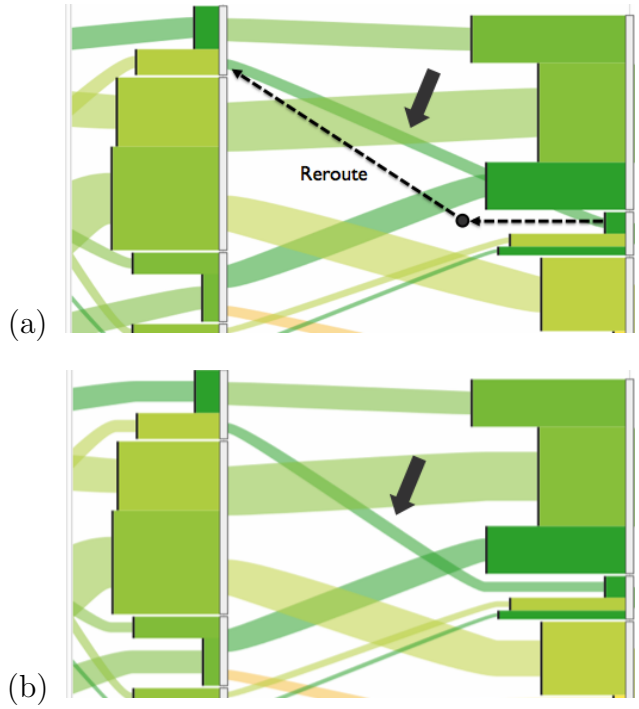

Figure C.7: Edge routing prevents overlaps between time and link edges. (a) A link edge is seen passing "behind" the time edge above it. Outflow's edge routing algorithm extends the link edge horizontally beyond the occluding time edge. (b) The new route avoids the overlap and makes the time edge fully visible.

(2) moving the origin of the link edge horizontally beyond the longest time edge's x position. This method does not guarantee avoidance for neighbors further than n. However, in practice a low value of n (e.g., $n = 3$) provides effective reduction in overlaps without excessive routing of edges.

C.3.4 Basic Interactions

To allow interactive data exploration, I further designed Outflow to support the following user interaction capabilities.

Panning & Zooming: Users can pan and zoom to uncover detailed structure.

Filtering: Users can filter both nodes and edges based on the the number of

associated entities to remove small subgroups.

Event Type Selection: Users can select which event types are used to construct the Outflow graph. This allows, for instance, for the omission of events that users deem uninteresting. For example, users in the medical use case can include/exclude a symptom (e.g., "Ankle Edema") if they deem it relevant/irrelevant to an analysis. In response, the visualization will be recomputed dynamically.

Brushing: Hovering the mouse over a node or an edge will highlight all paths traveled by all entities passing through the corresponding point in the Outflow graph (Figure C.8).

Tooltips: Hovering also triggers the display of tooltips which provide more information about individual nodes and edges. Tooltips show all events associated with the corresponding node/edge, the average outcome, and the total number of entities in the subgroup (Figure C.8).

Pinning: Users can "pin" a node or edge to freeze the brushed selection. This interaction is performed by clicking on the element to be pinned. Users can then move the mouse pointer to display tooltips for brushed subsets. This allows the quick retrieval of information about subsets that satisfy two constraints. For example, a user in soccer use case can pin soccer games that reached a 2-2 score before moving the mouse pointer to hover over the 1-0 state to see detailed information about the set of matches that pass through both nodes.

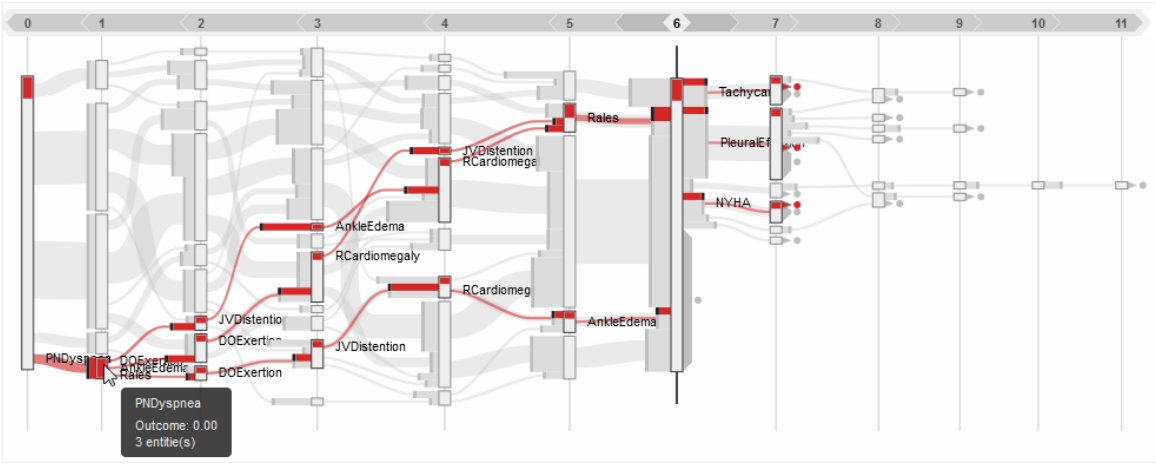

Figure C.8: Interactive brushing allows users to highlight paths emanating from specific nodes or edges in the visualization. This allows users to quickly see alternative progression paths taken by entities passing through a given state.

C.3.5 Simplification

The rendering techniques outlined earlier in this section can significantly reduce visual clutter and make the visualization more legible. However, there are still situations when visual complexity arises due to inherent complexity in the underlying data. To enable analyses of these more challenging datasets, Outflow includes a simplification algorithm that actively simplifies the underlying graph structure used to construct the visualization.

I apply a hierarchical clustering method that reduces the number of states in an Outflow graph by merging similar states within the same layer. States with a similarity distance less than a user-specified threshold are grouped together, while states that do not satisfy the threshold remain distinct. The user controls the threshold via a slider on the user interface. This prototype defines the similarity distance between two states as the difference in average outcomes (Equation C.1). Alternative measures could be easily substituted (e.g., a similarity-based metric

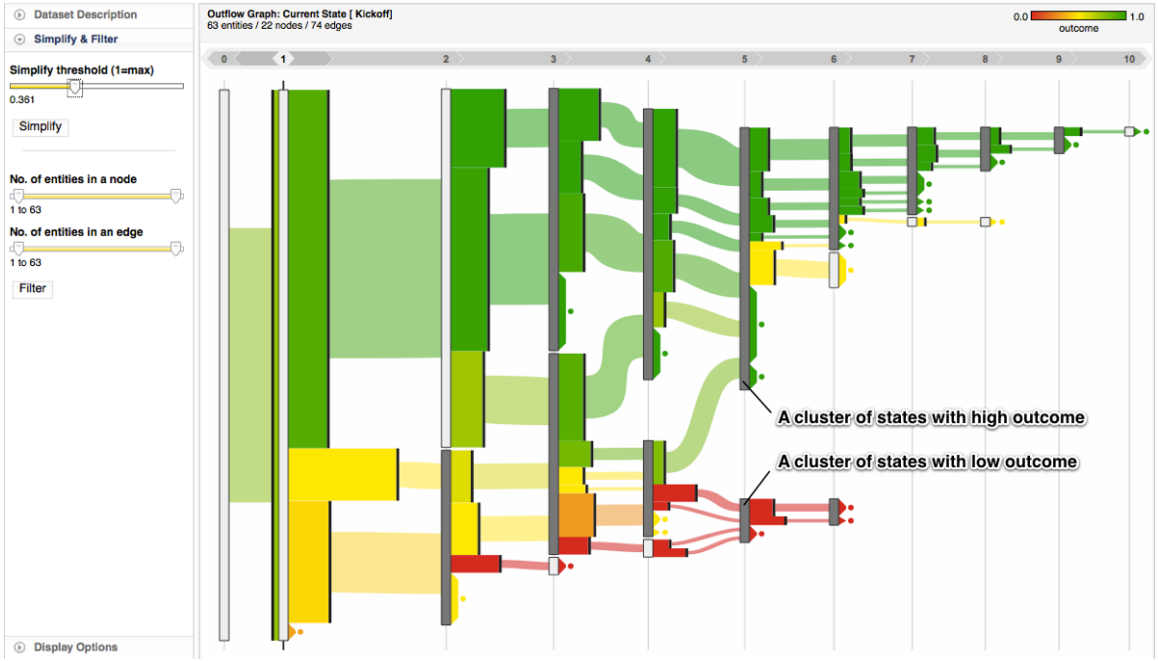

Figure C.9: Using the same dataset as illustrated in Figure C.1, a user has adjusted the simplification slider to group states with similar outcomes. Clustered states are represented with gray nodes. This simplified view shows that as more events occur (i.e., as more goals are scored), the paths diverge into two distinct sets of clustered states. The simplified states more clearly separate winning scorelines from losing scorelines. As more goals are scored, the probable outcomes of the games become more obvious.

using application-specific properties of the underlying entities).

$$d(node_A, node_B) = |node_A.outcome - node_B.outcome| \tag{C.1}$$

The hierarchical process begins as follows. First, each state in a layer is assigned to its own cluster for which it is the only member. Then, distances between all pairs of clusters are computed. The distance between two clusters, defined in Equation C.2, is determined by the average of the distances between all nodes in

the first cluster to all nodes in the second cluster.

$$d(cluster_X, cluster_Y) = \frac{\sum_{m \in cluster_X \& n \in cluster_Y} d(m,n)}{size(cluster_X) * size(cluster_Y)} \quad \text{(C.2)}$$

Once the distances have been computed for all pairs of clusters in a given layer, clusters are merged in a greedy fashion. The most similar pair of clusters is merged, and the cluster distances are updated to reflect the new clustering. The greedy process repeats until either (1) only one cluster remains, or (2) the most similar pair of remaining clusters has a distance above the threshold specified by users.

After the simplification process completes, clusters containing multiple states are rendered as a single node filled with gray in the visualization. To preserve state transition information, the edges are not merged even though their origin nodes or destination nodes may have been simplified (Figure C.9). Nodes that are the alone in their cluster are rendered using the normal Outflow visual encoding.

C.3.6 Factors

As described so far, Outflow provides an interactive visualization of event pathways and their associated outcomes. However, it does not yet incorporate external factors that can often influence how the events progress. For example, while goals determine the pathways in soccer use case, yellow and red cards can have a major impact on how a game unfolds. Similarly, a CHF patient's symptoms may be strongly influenced by the medications they are prescribed. In cases where

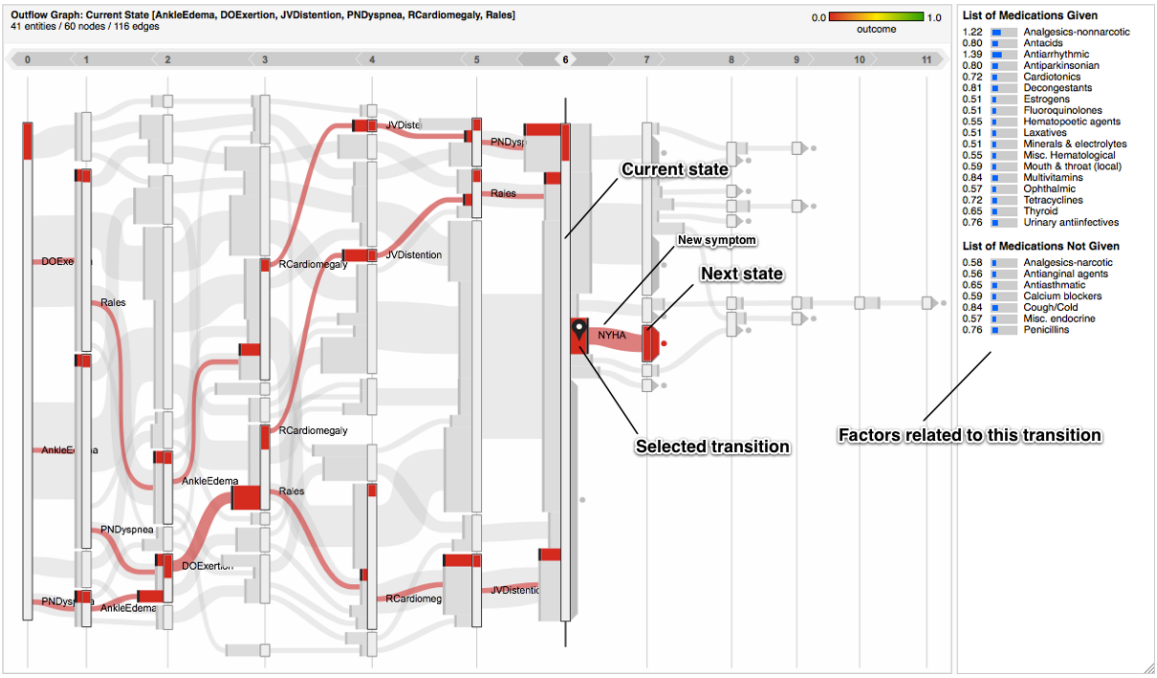

Figure C.10: Outflow highlights factors that are strongly correlated with specific event pathways. In this screenshot, a physician has focused on a group of patients transitioning from the current state due to the onset of the "NYHA" symptom. This transition seems to be deadly, as seen from the color-coding in red. The right sidebar displays medications (factors) with high correlations to this transition. The factor with the highest correlation in this example is prescribing antiarrhythmic agents. This correlation, which may or may not be causal, can help clinicians generate hypotheses about how best to treat a patient.

they can be controlled, factors can be important clues to analysts that are working to figure out if they can influence how an entity's events may progress.

Given the importance of factor analysis, I extend the basic Outflow data model to associate a set of timestamped factors (f_i, t_i) with each entity. Because of the timestamps, each occurrence of a factor can be placed within the sequence of events associated with the corresponding entity. The Outflow graph for a set of entities is constructed as before using only the entities' events. For each node and edge in the graph, I then compute additional statistics to identify correlated factors and suggest them to users via the user interface.

For this prototype, I have derived two metrics to detect factors that occur unusually often (or rarely) before a given state or transition. These metrics could be easily replaced with more sophisticated metrics that are more suitable for specific scenarios. In fact, I envision having a collection of metrics that users could choose from to measure various types of associations.

The prototype's baseline metrics are inspired by the **term frequency-inverse document frequency ($tf\star idf$)** measure [106] used widely in information retrieval: *presence correlation* and *absence correlation*.

1. *Presence correlation* detects factors that are unusually frequent for a given state or transition. For states, if a factor f_i often occurs before reaching state S_j while f_i rarely occurs elsewhere in the dataset, then f_i will be given a high presence correlation value for the corresponding state. I measure this correlation using by a **present factor frequency-inverse state frequency ($pff\star isf$)** score, which I define as follows.

$$pff = \frac{\text{number of entities with } f_i \text{ before } S_j}{\text{number of entities reaching } S_j} \tag{C.3}$$

$$isf_p = log\left(\frac{\text{number of states}}{\text{number of states preceded by } f_i + 1}\right) \tag{C.4}$$

$$pff\star isf = pff \cdot isf_p$$

A similar calculation is made for transitions by substituting S_j with $T_{m\rightarrow n}$ in Equation C.3 and replacing number of states with number of transitions in

Equation C.4. For isf_p, only states (or transitions) for the current layer or earlier are counted.

2. *Absence correlation* detects factors that are unusually rare for a given state or transition. For states, if a factor f_i rarely occurs before reaching state S_j while f_i occurs commonly elsewhere in the dataset, then f_i will be given a high absence correlation value for the corresponding state. This correlation is formulated as the **absent factor frequency-inverse state frequency** ($aff\star isf$) score defined below.

$$aff = \frac{\text{number of entities without } f_i \text{ before } S_j}{\text{number of entities reaching } S_j} \tag{C.5}$$

$$isf_a = log\left(\frac{\text{number of states}}{\text{number of states not preceded by } f_i + 1}\right) \tag{C.6}$$

$$aff\star isf = aff \cdot isf_a$$

A similar calculation is made for transitions by substituting S_j with $T_{m\to n}$ in Equation C.5 and replacing number of states with number of transitions in Equation C.6. For isf_a, only states (or transitions) for the current layer or earlier are counted.

A new sidebar panel is added to the Outflow user interface to display these correlation scores. When users mouse over any time edge, the panel is updated to display the most highly correlated present and absent factors for the corresponding

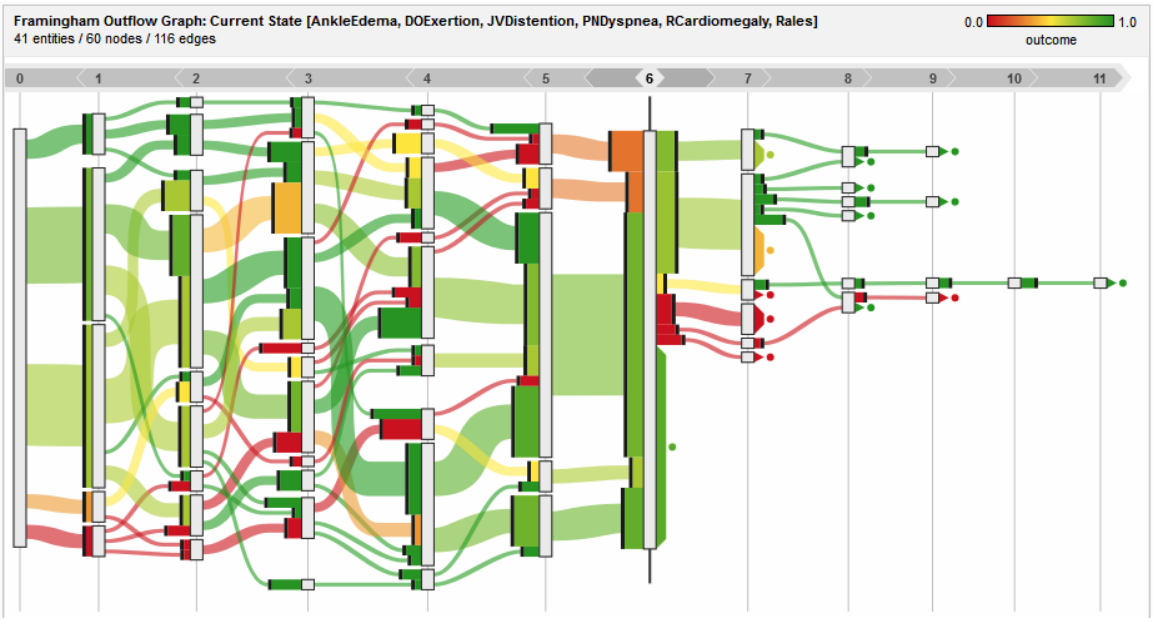

Figure C.11: Outflow aggregates temporal event data from a cohort of patients and visualizes alternative clinical pathways using color-coded edges that map to patient outcome. Interactive capabilities allow users to explore the data and uncover insights.

transition. The factor panel can be seen on the right side of Figure C.10. Factors are listed alphabetically and displayed with a histogram to convey the strength of the correlation.

C.4 Preliminary Analysis

Outflow was used to view the evolution over time for a cohort of CHF patients similar to a clinician's current patient. An initial analysis illuminates a number of interesting findings and highlights that various types of patients evolve differently. Here are two examples of the type of analysis that can be performed using the Outflow technique.

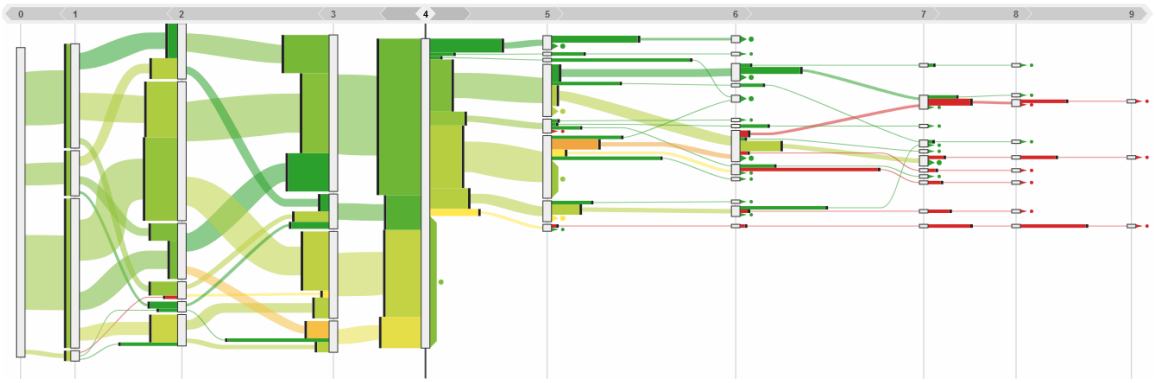

Figure C.12: The progression from green to red when moving left to right in this figure shows that patients with more symptoms exhibit worse outcomes.

Leading Indicators: In several scenarios, patient outcome is strongly correlated with certain leading indicators. For example, consider the patient cohort visualized in Figure C.11. The strong red and green colors assigned to the first layer of edges in the visualization shows that the eventual outcome for patients in this cohort is strongly correlated with the very first symptom to appear. Similarly, the strong red and green colors assigned to the first layer of edges after the alignment point show that the next symptom to appear may be critical in determining patient outcome.

Progressive Complications: In contrast to the prior example, which showed strong outcome correlation with specific paths, the patient cohort in Figure C.12 exhibits very different characteristics. At each time step, the outcomes across the different edges are relatively equal. However, the outcomes transition from green to red when moving left to right across the visualization. This implies that for this group of patients, no individual path is especially problematic historically. Instead, a general increase in co-occurring symptoms over time is the primary risk factor.

C.5 User Study

I conducted a user study to evaluate Outflow's ability to support a variety of event sequence analysis tasks. I first describe the study's design which asked users to answer questions about Man U.'s soccer season using a visualization of the data described in Section C.2.2. I then report the study's results and discuss its findings.

C.5.1 Design

Twelve users (eight males and four females) participated in this study. All users were adult professionals who are comfortable with computers. None of the users would consider themselves "soccer experts", but all have a basic understanding of the game. None of the users had any prior experience using Outflow.

C.5.1.1 Procedure

Each user participated in a single 60-minute session during which they were observed by a study moderator. Each session started with a brief orientation in which the moderator explained Outflow's design and interactions. Participants were then allowed to experiment with the visualization to gain some experience working with the system.

After roughly 15 minutes, the formal section of the study began. Participants were asked to perform a list of tasks. While the tasks were performed, the moderator recorded both accuracy and time to completion for each task. Users were then asked to freely explore the data and describe any interesting findings. After that,

users were given a written questionnaire to gather subjective feedback. Finally, the moderator debriefed the participants to learn about their experience and any comments or suggestions they might have.

C.5.1.2 Tasks and Questionnaire

Each user was given a list of 16 tasks to perform. The tasks were designed to to evaluate people's ability to understand proportion and cardinality of states and transitions, transition time, outcome of states and transitions, and factors associated with transitions.

The first nine tasks were designed to measure the participant's ability to interpret Outflow's visual representation. These tasks were further divided into two sets: practice tasks and test tasks. The first four tasks, unbeknownst to the participants, were used only to ensure that participants fully explored Outflow's visual design. Timing/accuracy data for these tasks was not included in this analysis. The five test tasks (T1–T5) asked questions similar to the preceding practice tasks, but on different aspects of the dataset.

The next seven tasks were designed to measure performance when using Outflow's interactive capabilities. Once again, these tasks were split into two groups: practice tasks and test tasks. The first three were designed to give participants practice using Outflow's interactive features and results were not included in this analysis. The four remaining tasks (T6–T9) asked similar questions about other aspects of the dataset. The test tasks used in the study are as follows:

T1. "Can you find the state where Man U. conceded the first goal?" *Objective*: Traverse graph using labels.

T2. "What happened most rarely after Man U. conceded the first goal?" *Objective*: Interpret proportion from height of time edge.

T3. "Was it faster to concede a goal or to score a goal from the state in T1?" *Objective*: Interpret time from time edge width.

T4. "Was it more likely for Man U. to win or lose after the state in T1?" *Objective*: Interpret state outcome.

T5. "What is the most common score after two goals are scored (2-0, 1-1 or 0-2)?" *Objective*: Traverse graph and interpret proportion from state node height.

T6. "Can you find the state where Man U. led 2-1?" *Objective*: Traverse graph using labels.

T7. "Which transition from the state in T6 led to the lowest percentage of winning? How many percent?" *Objective*: Use tooltip.

T8. "Which factor(s) are highly correlated with the transition in T6?" *Objective*: Understand factors.

T9. "For games that reached 2-2, which situation resulted in better outcomes? (a) Scoring to move from down 1-2 into a 2-2 tie or (b) conceding to move from a 2-1 lead to a 2-2 tie?" *Objective*: Use tooltip. The actual data shows a 0.73 outcome for (a) and a 0.67 outcome for (b). While the similar outcome values make the difference in color-coding hard to see, the tooltip provides access to the numerical outcome statistics.

At the end of their study session, participants completed a questionnaire. It contained eight questions (Q1–Q8) to which users responded on a 7-point Likert scale (1=very easy, 7=very hard). The questionnaire also included free response questions to gather more subjective feedback. The eight Likert-scale questions included the following:

Q1. Is it easy or hard to learn how to use?

Q2. Is it easy or hard to interpret proportion of states?

Q3. Is it easy or hard to interpret proportion of transitions?

Q4. Is it easy or hard to interpret transition time?

Q5. Is it easy or hard to interpret outcome of states?

Q6. Is it easy or hard to interpret outcome of transitions?

Q7. Is it easy or hard to understand factors correlated with a transition?

Q8. Is it easy or hard to find a particular state in the graph?

C.5.2 Results

C.5.2.1 Accuracy

Overall, participants were able to complete the tasks accurately with only three mistakes observed out of 108 total tasks (97.2% accuracy).

Two users erred on T4. These participants were able to identify the node that was the focus of the question. However, neither responded based on the color of the identified node. One user looked at all of the future paths and mentally

aggregated/averaged the values (incorrectly) to guess at the eventual outcome. This approach is subject to bias because long paths cover more pixels even if their outcome is not representative. When told of their mistake, the user said that looking at the color of the node "would have been trivial. I just forgot." The second user who answered T4 incorrectly responded based on the color of the largest outbound path. While that path corresponds to the most often occurring next event, it does not by itself represent the overall average outcome for the identified node. I hypothesize that both errors were due in large part to the users' lack of experience.

The only other error occurred on T9. However, during the questionnaire portion of the study session, it was determined that the user misunderstood the task. He actually used Outflow correctly to answer what he thought the question was asking.

C.5.2.2 Speed

Participants were able to finish the tasks rapidly with average completion times ranging between 5.33 to 64.22 seconds.[2] The wide range in timings reflects in part variations in task complexity. However, the standard deviations between timings for individual tasks were fairly large (up to 30.47 s). In general, some users quickly mapped tasks to a feature of the visualization while others spent more time thinking about what exactly the question was asking before settling on an answer. Overall, the times are quite fast and I believe that allowing novice users to precisely answer

[2]Task completion times were as follows: (T1) Mean=5.33 ± SD=4.18 s; (T2) 8.79 ± 7.09 s; (T3) 8.16 ± 7.64 s; (T4) 26.26 ± 14.39 s; (T5) 49.53 ± 30.47 s; (T6) 10.19± 3.01 s; (T7) 16.98 ± 10.81 s; (T8) 7.98 ± 4.01 s; (T9) 64.22 ± 26.92 s

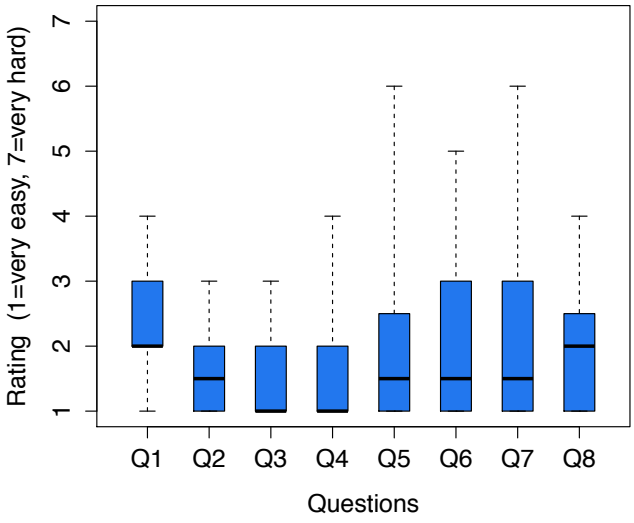

Figure C.13: Questionnaire results for each of the eight questions (Q1–Q8) answered on a 7-point scale by study participants.

a complex task like T9 in as little as one minute highlights Outflow's utility.

C.5.2.3 Unrestricted Exploration

After completing the study tasks, users were given time to freely explore the visualization. Participants were able to quickly identify several interesting aspects of the dataset. For example, many users quickly found the highest scoring game which had a total of eight goals. Participants were also drawn to states that had multiple "arriving" edges and compared outcomes for the alternative paths. This is similar to what they were asked to do in T9. One user discovered a general trend that Man U. rarely received yellow cards when they lost. This could be interpreted as a sign of lack of intensity in those games.

Users also enjoyed "the ability to investigate the progression of a large number of games on one screen." Users observed a number of strong global trends, including:

(1) Man U. wins a large majority of their games, (2) Man U. wins most high scoring games, (3) Man U. often comes back after falling behind, and (4) Man U. rarely loses a game when ahead.

C.5.2.4 Questionnaire and Debriefing

Results from the questionnaire are shown in Figure C.13. A pairwise t-test between all questions shows no significant difference between the ratings of each question. Average ratings from all questions are between 1.50–2.33, suggesting that the participants generally found Outflow easy to learn and easy to use.[3]

However, not all participants responded in the same way. As shown in Figure C.13, there were a small number of higher ratings for questions Q5-Q7 that indicate some frustration. In fact, all of the high scores (> 4) came from a single participant who had difficulty understanding the difference between the outcome of a *state* and the outcome of a *transition*. He repeatedly asked for clarification and commented in the questionnaire that tasks on these topics required "some effort to parse the wording". The moderator explained that while a state (e.g., 1-1) has one outcome, the transitions to that state (e.g., conceding to go from 1-0 to 1-1 vs. scoring to go from 0-1 to 1-1) can have different outcomes. The user understood for a moment, but then quickly became confused again. His difficulty in grasping this concept led to his high responses (and slower task completion times).

Based on free response questions and interviews with the moderator, partici-

[3]Question ratings were as follows: (Q1) Mean=2.33± SD=0.78; (Q2) 1.58±0.67; (Q3) 1.5±0.67; (Q4) 1.58 ± 0.90; (Q5) 2.08 ± 1.56; (Q6) 2.17 ± 1.53; (Q7) 2.25 ± 1.71; and (Q8) 2 ± 0.953

pants felt overall that Outflow was "pretty", "looks cool", and that the colors were "very meaningful". They said that it provides a "good high level view" that encodes a lot of information into a single "simple to follow" visualization. In addition, users felt that the ability to see outcome associated with alternative paths out of (or into) a given state, and not just the state itself, is a powerful feature. "I like the difference between states and transitions [which allowed me] to compare two paths to the same state and to understand differences." Another participant commented that "highlighting of paths was very helpful."

When asked about learning to use Outflow, some participants expressed that the tool is unique, and therefore required some training to get used to it. However, those participants also felt strongly that by the end of the study they were proficient in using the tool. One remarked that they would have done much better on the study tasks "with a few more minutes of training at the start" suggesting a short learning curve.

Some limitations of the current design were also identified during the study. One participant pointed out that users tend to view width as time, but that the widest sequences don't necessarily take the longest (in fact, soccer games all take roughly the same amount of time). This is indeed a limitation of the technique. Outflow handles graphs with multiple incoming edges to a node. Because these different paths to a node can have different durations, time can not be represented horizontally using an absolute time axis. While making comparisons between alternative paths easier, this design choice can make temporal comparisons across multiple steps somewhat harder.

Participants also suggested ideas for new features including (1) the ability to pin multiple states at once, and (2) moving the display of correlated factors into the main visualization space (instead of the separate sidebar used in the current design).

C.6 Summary

This appendix shows an example of new ideas that are inspired from the work in this dissertation. *Outflow*, a new visualization technique, was designed to summarize temporal event sequences, and show aggregated pathways, factors and outcomes. New visual representation, a number of interactive features and two simple metrics for factor recommendations are presented. I provided a detailed description of the visualization's design including a multi-step rendering process designed to reduce visual complexity. A portion of this process—the combination of Sugiyama's algorithm to reduce edge crossings and force-directed layout to straighten unnecessarily curvy edges—is also applicable to a more general class of DAG layout problems. Two preliminary analysis highlight some of the capabilities of this approach. Results from a user study with twelve participants demonstrated that users were able to learn how to use Outflow easily within fifteen minutes of training, and were able to accurately and quickly perform a broad range of analysis tasks.

There are many promising directions to further explore including integration with forecasting/prediction algorithms, the use of more sophisticated similarity measures, and deeper evaluation studies with domain experts.

Bibliography

[1] Jae-wook Ahn and Peter Brusilovsky. Adaptive visualization of search results: Bringing user models to visual analytics. *Information Visualization*, 8(3):167–179, 2009.

[2] Jae-wook Ahn, Peter Brusilovsky, Daqing He, Jonathan Grady, and Qi Li. Personalized web exploration with task models. In *Proceeding of the 17th international conference on World Wide Web - WWW '08*, page 1. ACM Press, 2008.

[3] Jae-wook Ahn, Krist Wongsuphasawat, and Peter Brusilovsky. Analyzing User Behavior Patterns in Adaptive Exploratory Search Systems with LifeFlow. In *Proceedings of the Workshop on Human-Computer Interaction and Information Retrieval (HCIR)*, 2011.

[4] Alfred V. Aho. Algorithms for finding patterns in strings. In *Handbook of Theoretical Computer Science, Volume A: Algorithms and Complexity*, chapter 5, pages 255–400. 1990.

[5] Wolfgang Aigner and Silvia Miksch. CareVis: integrated visualization of computerized protocols and temporal patient data. *Artificial Intelligence in Medicine*, 37(3):203–18, July 2006.

[6] Wolfgang Aigner, Silvia Miksch, Heidrun Schumann, and Christian Tominski. *Visualization of Time-Oriented Data*. Number 1997 in Human-Computer Interaction Series. Springer, 2011.

[7] Wolfgang Aigner, Silvia Miksch, Bettina Thurnher, and Stefan Biffl. PlanningLines: Novel Glyphs for Representing Temporal Uncertainties and Their Evaluation. In *Proceedings of the International Conference on Information Visualization (IV)*, pages 457–463. IEEE, 2005.

[8] Diane Lindwarm Alonso, Anne Rose, and Catherine Plaisant. Viewing personal history records: a comparison of tabular format and graphical presentation using LifeLines. *Behaviour & Information Technology*, 17(5):249–262, September 1998.

[9] Stephen F. Altschul, Warren Gish, Webb Miller, Eugene W. Myers, and David J. Lipman. Basic local alignment search tool. *Journal of Molecular Biology*, 215(3):403–410, 1990.

[10] Paul André, Max L. Wilson, Alistair Russell, Daniel A. Smith, Alisdair Owens, and M.c. Schraefel. Continuum: designing timelines for hierarchies, relationships and scale. *Proceedings of the Annual ACM Symposium on User Interface Software and Technology (UIST)*, page 101, 2007.

[11] Henrik André-Jönsson and Dushan Z. Badal. Using signature files for querying time-series data, 1997.

[12] Francis J. Anscombe. Graphs in Statistical Analysis. *American Statistician*, 27(1):17–21, 1973.

[13] Ragnar Bade, Stefan Schlechtweg, and Silvia Miksch. Connecting time-oriented data and information to a coherent interactive visualization. In *Proceedings of the Annual SIGCHI Conference on Human Factors in Computing Systems (CHI)*, pages 105–112. ACM, 2004.

[14] Benjamin B. Bederson, Jesse Grosjean, and Jon Meyer. Toolkit design for interactive structured graphics. *IEEE Transactions on Software Engineering*, 30(8):535–546, August 2004.

[15] Benjamin B. Bederson, Ben Shneiderman, and Martin Wattenberg. Ordered and quantum treemaps: Making effective use of 2D space to display hierarchies. *ACM Transactions on Graphics*, 21(4):833–854, 2002.

[16] Donald J. Berndt and James Clifford. Using dynamic time warping to find patterns in time series. In *AAAI-94 Workshop on Knowledge Discovery in Databases*, pages 229–248, 1994.

[17] Jorik Blaas, Charl P. Botha, Edward Grundy, Mark W. Jones, Robert S. Laramee, and Frits H. Post. Smooth graphs for visual exploration of higher-order state transitions. *IEEE Transactions on Visualization and Computer Graphics*, 15(6):969–76, 2009.

[18] Christine Bonhomme and Marie-Aude Aufaure. Mixing icons, geometric shapes and temporal axis to propose a visual tool for querying spatio-temporal databases. In *Proceedings of the Working Conference on Advanced Visual Interfaces (AVI)*, pages 282–289. ACM, 2002.

[19] Christine Bonhomme, Claude Trépied, Marie-Aude Aufaure, and Robert Laurini. A visual language for querying spatio-temporal databases. In *Proceedings of the ACM International Symposium on Advances in Geographic Information Systems (GIS)*, pages 34–39. ACM, 1999.

[20] Taylor Booth. *Sequential Machines and Automata Theory*. John Wiley and Sons, New York, NY, USA, 1967.

[21] Michael Bostock and Jeffrey Heer. Protovis: a graphical toolkit for visualization. *IEEE Transactions on Visualization and Computer Graphics*, 15(6):1121–8, 2009.

[22] Kevin Buchin, Bettina Speckmann, and Kevin Verbeek. Flow Map Layout via Spiral Trees. *IEEE transactions on visualization and computer graphics*, 17(12):2536–44, December 2011.

[23] Michael Burch, Fabian Beck, and Stephan Diehl. Timeline trees: visualizing sequences of transactions in information hierarchies. In *Proceedings of the Working Conference on Advanced Visual Interfaces (AVI)*, pages 75–82, Napoli, Italy, 2008. ACM.

[24] Giuseppe Carenini and John Loyd. ValueCharts: analyzing linear models expressing preferences and evaluations. In *Proceedings of the Working Conference on Advanced Visual Interfaces (AVI)*, pages 150–157. ACM, 2004.

[25] John V. Carlis and Joseph A. Konstan. Interactive visualization of serial periodic data. In *Proceedings of the Annual ACM Symposium on User Interface Software and Technology (UIST)*, pages 29–38, New York, New York, USA, 1998. ACM.

[26] N. S. Chang and K. S. Fu. Query-by-Pictorial-Example. *IEEE Transactions on Software Engineering*, 6:519–524, 1980.

[27] Remco Chang, Mohammad Ghoniem, Robert Kosara, William Ribarsky, Jing Yang, Evan Suma, Caroline Ziemkiewicz, Daniel Kern, and Agus Sudjianto. WireVis: Visualization of Categorical, Time-Varying Data From Financial Transactions. In *Proceedings of the IEEE Symposium on Visual Analytics Science and Technology (VAST)*, pages 155–162. IEEE, October 2007.

[28] Cleve Cheng, Yuval Shahar, Angel Puerta, and Daniel Stites. Navigation and visualization of abstractions of time-oriented clinical data. *Section on Medical Informatics Technical Report No. SMI-97*, 688, 1997.

[29] Luca Chittaro. Visualization of patient data at different temporal granularities on mobile devices. *Proceedings of the Working Conference on Advanced Visual Interfaces (AVI)*, page 484, 2006.

[30] Jan Chomicki. Temporal query languages: A survey. In *Proceedings of the International Conference on Temporal Logic*, volume 827, pages 506–534. Springer, 1994.

[31] James Clifford and Albert Croker. The Historical Relational Data Model (HRDM) and Algebra Based on Lifespans. In *Proceedings of the IEEE International Conference on Data Engineering (ICDE)*, pages 528–537. IEEE, 1987.

[32] Steve B. Cousins and Michael G Kahn. The visual display of temporal information. *Artificial Intelligence in Medicine*, 3(6):341–357, 1991.

[33] Steve B. Cousins, Michael G Kahn, and Mark E Frisse. The display and manipulation of temporal information. In *Proceedings of the Annual Symposium on Computer Application in Medical Care*, pages 76–80, November 1989.

[34] Raimund Dachselt and Markus Weiland. TimeZoom: A Flexible Detail and Context Timeline. In *Proceedings of the Annual SIGCHI Conference on Human Factors in Computing Systems (CHI) - Extended Abstracts*, page 682. ACM, 2006.

[35] Guiseppe Di Battista, Peter Eades, Roberto Tamassia, and Ioannis G. Tollis. Algorithms for drawing graphs: an annotated bibliography. *Computational Geometry*, 4(5):235–282, October 1994.

[36] Simon Dobrisek, Janez Zibert, Nikola Pavesić, and France Mihelic. An edit-distance model for the approximate matching of timed strings. *IEEE transactions on pattern analysis and machine intelligence*, 31(4):736–41, April 2009.

[37] Jerry Fails, Amy Karlson, Layla Shahamat, and Ben Shneiderman. A Visual Interface for Multivariate Temporal Data: Finding Patterns of Events across Multiple Histories. In *Proceedings of the IEEE Symposium on Visual Analytics Science and Technology (VAST)*, volume 0, pages 167–174. IEEE, October 2006.

[38] Jean-Daniel Fekete. The InfoVis Toolkit. In *Proceedings of the IEEE Symposium on Information Visualization (InfoVis)*, pages 167–174. IEEE, 2004.

[39] Samuel Fomundam and Jeffrey W. Herrmann. A Survey of Queuing Theory Applications in Healthcare. *Mechanical Engineering*, pages 1–22, 2007.

[40] Michael Friendly. Visions and re-visions of Charles Joseph Minard. *Journal of Educational and Behavioral Statistics*, 27(1):31–51, 2002.

[41] Cristina Gómez-Alonso and Aida Valls. A Similarity Measure for Sequences of Categorical Data Based on the Ordering of Common Elements. In Vicenç Torra and Yasuo Narukawa, editors, *Modeling Decisions for Artificial Intelligence*, number 1, chapter 13, pages 134–145. Springer, 2008.

[42] David Gotz and Michelle X. Zhou. Characterizing users visual analytic activity for insight provenance. *Information Visualization*, 8(1):42–55, January 2009.

[43] John Alexis Guerra Gómez, Krist Wongsuphasawat, Taowei David Wang, Michael L. Pack, and Catherine Plaisant. Analyzing Incident Management Event Sequences with Interactive Visualization. In *Transportation Research Board Annual Meeting Compendium*, Washington, DC, January 2011.

[44] Richard W. Hamming. Error Detecting and Error Correcting Codes. *The Bell System Technical Journal*, 29(2):147–160, 1950.

[45] Derek L Hansen, Dana Rotman, Elizabeth Bonsignore, Nataša Milić-frayling, Eduarda Mendes Rodrigues, Marc Smith, Ben Shneiderman, and Tony Capone. Do You Know the Way to SNA ?: A Process Model for Analyzing and Visualizing Social Media Data. Technical Report 3, University of Maryland: Human Computer Interaction Lab, 2009.

[46] Beverly L. Harrison, Russell Owen, and Ronald M. Baecker. Timelines: an interactive system for the collection and visualization of temporal data. In *Proceedings of Graphics Interface (GI)*, pages 141–141. Citeseer, 1994.

[47] J. A. Hartigan. Representation of Similarity Matrices by Trees. *Journal of the American Statistical Association*, 62(320):1140–1158, 1967.

[48] Susan Havre, Elizabeth Hetzler, Paul Whitney, and Lucy Nowell. ThemeRiver: visualizing thematic changes in large document collections. *IEEE Transactions on Visualization and Computer Graphics*, 8(1):9–20, 2002.

[49] K. Priyantha Hewagamage, Masahito Hirakawa, and Tadao Ichikawa. Interactive visualization of spatiotemporal patterns using spirals on a geographical map. *Proceedings of the IEEE Symposium on Visual Languages*, pages 296–303, 1999.

[50] S. Hibino and E.A. Rundensteiner. A visual query language for identifying temporal trends in video data. In *Proceedings of the International Workshop on Multi-Media Database Management Systems*, pages 74–81. IEEE, 1995.

[51] Stacie Hibino and Elke A. Rundensteiner. User interface evaluation of a direct manipulation temporal visual query language. In *Proceedings of the ACM International Conference on Multimedia (MULTIMEDIA)*, pages 99–107. ACM, 1997.

[52] Harry Hochheiser and Ben Shneiderman. Dynamic query tools for time series data sets: Timebox widgets for interactive exploration. *Information Visualization*, 3(1):1–18, 2004.

[53] Barry E. Jacobs and Cynthia A. Walczak. A Generalized Query-by-Example Data Manipulation Language Based on Database Logic. *IEEE Transactions on Software Engineering*, SE-9(1):40–57, 1983.

[54] Jing Jin and Pedro Szekely. QueryMarvel: A visual query language for temporal patterns using comic strips. In *Proceedings of the IEEE Symposium on Visual Languages and Human-Centric Computing (VL/HCC)*, pages 207–214. IEEE, September 2009.

[55] Brian Johnson and Ben Shneiderman. Tree-maps: A space-filling approach to the visualization of hierarchical information structures. In *Proceedings of the IEEE Conference on Visualization (Vis)*, pages 284–291. IEEE, 1991.

[56] Geir Jordet, Esther Hartman, Chris Visscher, and Koen A P M Lemmink. Kicks from the penalty mark in soccer: the roles of stress, skill, and fatigue for kick outcomes. *Journal of Sports Sciences*, 25(2):121–129, 2007.

[57] Sean Kandel, Andreas Paepcke, Joseph Hellerstein, and Jeffrey Heer. Wrangler: interactive visual specification of data transformation scripts. In *Proceedings of the Annual SIGCHI Conference on Human Factors in Computing Systems (CHI)*, page 3363, New York, New York, USA, 2011. ACM Press.

[58] Hyunmo Kang, Lise Getoor, Ben Shneiderman, Mustafa Bilgic, and Louis Licamele. Interactive entity resolution in relational data: a visual analytic tool and its evaluation. *IEEE transactions on visualization and computer graphics*, 14(5):999–1014, 2008.

[59] Gerald M. Karam. Visualization using timelines. In *Proceedings of the ACM SIGSOFT International Symposium on Software Testing and Analysis*, pages 125–137. ACM, 1994.

[60] Toshikazu Kato, Takio Kurita, Nobuyuki Otsu, and Kyoji Hirata. A sketch retrieval method for full color image database-query by visual example. In *Proceedings of the IAPR International Conference on Pattern Recognition*, pages 530–533. IEEE, 1992.

[61] Ernst Kleiberg, Huub van De Wetering, and Jarke J. van Wijk. Botanical visualization of huge hierarchies. In *Proceedings of the IEEE Symposium on Information Visualization (InfoVis)*, page 87. IEEE, 2001.

[62] Denis Klimov, Yuval Shahar, and Meirav Taieb-Maimon. Intelligent selection and retrieval of multiple time-oriented records. *Journal of Intelligent Information Systems*, 35(2):261–300, September 2009.

[63] Denis Klimov, Yuval Shahar, and Meirav Taieb-Maimon. Intelligent visualization and exploration of time-oriented data of multiple patients. *Artificial Intelligence in Medicine*, 49(1):11–31, May 2010.

[64] Anthony C. Klug. Abe: a query language for constructing aggregates-by-example. In *Proceedings of the LBL Workshop on Statistical Database Management (SSDBM)*, pages 190–205. Lawrence Berkeley Lab, 1981.

[65] Milos Krstajic, Enrico Bertini, and Daniel A. Keim. CloudLines: Compact Display of Event Episodes in Multiple Time-Series. *IEEE Transactions on Visualization and Computer Graphics*, 17(12):2432, 2011.

[66] Joseph B. Kruskal and James M. Landwehr. Icicle Plots: Better Displays for Hierarchical Clustering. *The American Statistician*, 37(2):162–168, 1983.

[67] Harold W. Kuhn. The Hungarian method for the assignment problem. *Naval Research Logistics Quarterly*, 2(1-2):83–97, March 1955.

[68] Jonh Lamping and Ramana Rao. The Hyperbolic Browser: A Focus+Context Technique for Visualizing Large Hierarchies. *Journal of Visual Languages & Computing*, 7(1):33–55, March 1996.

[69] Ivica Letunic and Peer Bork. Interactive Tree Of Life (iTOL): an online tool for phylogenetic tree display and annotation. *Bioinformatics*, 23(1):127–8, January 2007.

[70] Vladimir I. Levenshtein. Binary codes capable of correcting deletions, insertions and reversals. *Soviet Physics Doklady*, 10(8):707–710, 1966.

[71] Wen-Syan Li, K. Selcuk Candan, Kyoji Hirata, and Yoshinori Hara. IFQ: a visual query interface and query generator for object-based media retrieval. In *Proceedings of the IEEE International Conference on Multimedia Computing and Systems*, pages 353–361. IEEE, 1997.

[72] Jessica Lin, Eamonn Keogh, and Stefano Lonardi. Visualizing and discovering non-trivial patterns in large time series databases. *Information Visualization*, 4(2):61–82, April 2005.

[73] Jessica Lin, Eamonn Keogh, Stefano Lonardi, Jeffrey P. Lankford, and Donna M. Nystrom. Visually mining and monitoring massive time series. In *Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD)*, page 460, New York, New York, USA, 2004. ACM.

[74] Jessica Lin, Eamonn Keogh, Stefano Lonardi, Jeffrey P. Lankford, and Donna M. Nystrom. VizTree: a tool for visually mining and monitoring massive time series databases. In *Proceedings of the International Conference on Very large data bases (VLDB)*, volume 30, page 1272. VLDB Endowment, 2004.

[75] Lauro Lins, Marta Heilbrun, Juliana Freire, and Claudio Silva. VISCARETRAILS: Visualizing Trails in the Electronic Health Record with Timed Word Trees, a Pancreas Cancer Use Case. In *Proceedings of the IEEE VisWeek Workshop on Visual Analytics in Healthcare*, pages 13–16, 2011.

[76] Heikki Mannila and Pirjo Moen. Similarity between Event Types in Sequences. In *Proceedings of the International Conference on Data Warehousing and Knowledge Discovery (DaWaK)*, pages 271–280. Springer, 1999.

[77] Heikki Mannila and Pirjo Ronkainen. Similarity of Event Sequence. In *Proceedings of the International Workshop on Temporal Representation and Reasoning (TIME)*, pages 136–139, 1997.

[78] Heikki Mannila and Jouni K. Seppänen. Finding similar situations in sequences of events via random projections. In *Proceedings of the SIAM International Conference on Data Mining*, pages 1–16. Citeseer, 2001.

[79] P. A. McKee, W. P. Castelli, P. M. McNamara, and W. B. Kannel. The natural history of congestive heart failure: the Framingham study. *The New England journal of medicine*, 285(26):1441–6, December 1971.

[80] John J. McMurray and Marc A. Pfeffer. Heart failure. *Lancet*, 365(9474):1877–1889, 2005.

[81] Marcel Mongeau and David Sankoff. Comparison of Musical Sequences. *Computer and the Humanities*, 24(3):161–175, 1990.

[82] Tamara Munzner. H3: laying out large directed graphs in 3D hyperbolic space. *Proceedings of Visualization Conference, Information Visualization Symposium and Parallel Rendering Symposium (VIZ)*, pages 2–10,, 1997.

[83] Tadao Murata. Petri nets: Properties, analysis and applications. In *Proceedings of the IEEE*, volume 77, pages 541–580. IEEE, 1989.

[84] Gonzalo Navarro. A guided tour to approximate string matching. *ACM Computing Surveys*, 33(1):31–88, March 2001.

[85] Hannes Obweger, Martin Suntinger, Josef Schiefer, and Gunther Raidl. Similarity searching in sequences of complex events. In *Proceedings of the International Conference on Research Challenges in Information Science (RCIS)*, pages 631–640. IEEE, May 2010.

[86] Gultekin Ozsoyoglu, Victor Matos, and Meral Ozsoyoglu. Query processing techniques in the summary-table-by-example database query language. *ACM Transactions on Database Systems*, 14(4):526–573, December 1989.

[87] Gultekin Ozsoyoglu and Huaqing Wang. Example-based graphical database query languages. *Computer*, 26(5):25–38, May 1993.

[88] William R. Pearson and David J. Lipman. Improved tools for biological sequence comparison. In *Proceedings of the National Academy of Sciences of the United States of America*, volume 85, pages 2444–2448, April 1988.

[89] Adam Perer and Ben Shneiderman. Systematic yet flexible discovery: Guiding Domain Experts through Exploratory Data Analysis. In *Proceedings of the International Conference on Intelligent User Interfaces (IUI)*, page 109, New York, New York, USA, 2008. ACM.

[90] Carl Adam Petri. Communication with automata. Technical report, DTIC Research Report, 1966.

[91] Doantam Phan, John Gerth, Marcia Lee, Andreas Paepcke, and Terry Winograd. Visual analysis of network flow data with timelines and event plots. In *Proceedings of the International Workshop on Visualization for Computer Security (VizSEC)*, pages 85–99. Springer, 2007.

[92] Doantam Phan, Andreas Paepcke, and Terry Winograd. Progressive multiples for communication-minded visualization. In *Proceedings of Graphics Interface (GI)*, page 225, New York, New York, USA, 2007. ACM.

[93] Doantam Phan, Ling Xiao, Ron Yeh, Pat Hanrahan, and Terry Winograd. Flow map layout. In *Proceedings of the IEEE Symposium on Information Visualization (InfoVis)*, pages 219–224. IEEE, 2005.

[94] William A Pike, John Stasko, Remco Chang, and Theresa A OConnell. The science of interaction. *Information Visualization*, 8(4):263–274, 2009.

[95] Peter Pirolli and Stuart Card. The sensemaking process and leverage points for analyst technology as identified through cognitive task analysis. In *Proceedings of International Conference on Intelligence Analysis*, volume 2005, pages 2–4, 2005.

[96] Catherine Plaisant, Jesse Grosjean, and Benjamin B. Bederson. SpaceTree: supporting exploration in large node link tree, design evolution and empirical evaluation. In *Proceedings of the IEEE Symposium on Information Visualization (InfoVis)*, pages 57–64. IEEE, 1998.

[97] Catherine Plaisant, Rich Mushlin, A. Snyder, J. Li, D. Heller, and Ben Shneiderman. LifeLines: using visualization to enhance navigation and analysis of patient records. In *Proceedings of the AMIA Annual Symposium*, pages 76–80. American Medical Informatics Association, 1998.

[98] Seth M. Powsner and Edward R. Tufte. Graphical Summary of Patient Status. *Lancet*, 344(8919):386–389, 1994.

[99] A. Johannes Pretorius and Jarke J. van Wijk. Visual analysis of multivariate state transition graphs. *IEEE Transactions on Visualization and Computer Graphics*, 12(5):685–92, 2006.

[100] A. Johannes Pretorius and Jarke J. van Wijk. Visual Inspection of Multivariate Graphs. *Computer Graphics Forum*, 27(3):967–974, May 2008.

[101] Edward M. Reingold and John S. Tilford. Tidier Drawings of Trees. *IEEE Transactions on Software Engineering*, SE-7(2):223–228, March 1981.

[102] Patrick Riehmann, Manfred Hanfler, and Bernd Froehlich. Interactive Sankey diagrams. In *Proceedings of the IEEE Symposium on Information Visualization (InfoVis)*, pages 233–240. IEEE, 2005.

[103] Isidore Rigoutsos and Aris Floratos. Combinatorial pattern discovery in biological sequences: The TEIRESIAS algorithm. *Bioinformatics*, 14(1):55–67, January 1998.

[104] George G. Robertson, Stuart K. Card, and Jack D. Mackinlay. Information visualization using 3D interactive animation. *Communications of the ACM*, 36(4):57–71, 1993.

[105] Richard Rudgley. *The Lost Civilizations of the Stone Age*. Simon & Schuster, 1999.

[106] Gerard Salton and Christopher Buckley. Term-weighting approaches in automatic text retrieval. *Information Processing & Management*, 24(5):513–523, 1988.

[107] Michael Schroeder, David Gilbert, Jacques van Helden, and Penny Noy. Approaches to visualisation in bioinformatics: from dendrograms to Space Explorer. *Information Sciences*, 139(1-2):19–57, November 2001.

[108] Jinwook Seo and Ben Shneiderman. A rank-by-feature framework for interactive exploration of multidimensional data. *Information Visualization*, 4(2):96–113, 2005.

[109] Yuval Shahar, Dina Goren-Bar, David Boaz, and Gil Tahan. Distributed, intelligent, interactive visualization and exploration of time-oriented clinical data and their abstractions. *Artificial Intelligence in Medicine*, 38(2):115–35, 2006.

[110] Reza Sherkat and Davood Rafiei. Efficiently evaluating order preserving similarity queries over historical market-basket data. In *Proceedings of the International Conference on Data Engineering (ICDE)*, pages 19–30, 2006.

[111] Ben Shneiderman. Direct Manipulation: A Step Beyond Programming Languages. *Computer*, 16(8):57–69, August 1983.

[112] Ben Shneiderman. The eyes have it: a task by data type taxonomy for information visualizations. In *Proceedings of the IEEE Symposium on Visual Languages*, pages 336–343. IEEE, 1996.

[113] Ben Shneiderman. Extreme visualization: squeezing a billion records into a million pixels. In *Proceedings of the ACM SIGMOD international conference on Management of data*, pages 3–12. ACM, 2008.

[114] Ben Shneiderman and Catherine Plaisant. Strategies for evaluating information visualization tools. In *Proceedings of the AVI workshop on BEyond time and errors novel evaluation methods for information visualization (BELIV)*, pages 1–7. ACM, 2006.

[115] Richard T. Snodgrass. The temporal query language TQuel. *ACM Transactions on Database Systems*, 12(2):247–298, June 1987.

[116] Richard T. Snodgrass. *The TSQL2 temporal query language*. Kluwer Academic Publishers, 1995.

[117] Robert R. Sokal and Peter H. A. Sneath. *Principles of Numerical Taxonomy*. W. H. Freeman and Co., San Francisco, 1963.

[118] John Stasko and E. Zhang. Focus+context display and navigation techniques for enhancing radial, space-filling hierarchy visualizations. In *Proceedings of the IEEE Symposium on Information Visualization (InfoVis)*, pages 57–65. IEEE, 2000.

[119] Kozo Sugiyama, Shojiro Tagawa, and Mitsuhiko Toda. Methods for Visual Understanding of Hierarchical System Structures. *IEEE Transactions on Systems, Man, and Cybernetics*, 11(2):109–125, 1981.

[120] Martin Suntinger, Josef Schiefer, Hannes Obweger, and M. Eduard Groller. The Event Tunnel: Interactive Visualization of Complex Event Streams for Business Process Pattern Analysis. In *Proceedings of the IEEE Pacific Visualization Symposium (PacificVis)*, pages 111–118. IEEE, March 2008.

[121] Abdullah U. Tansel, M. Erol Arkun, and Gultekin Ozsoyoglu. Time-by-example query language for historical databases. *IEEE Transactions on Software Engineering*, 15(4):464–478, April 1989.

[122] Abdullah U. Tansel and Erkan Tin. The expressive power of temporal relational query languages. *IEEE Transactions on Knowledge and Data Engineering*, 9(1):120–134, 1997.

[123] James J. Thomas and Kristin A. Cook. *Illuminating the path: The research and development agenda for visual analytics*. IEEE, 2005.

[124] John W. Tukey. *Exploratory Data Analysis*. Addison-Wesley, 1977.

[125] Frank van Ham, Huub van De Wetering, and Jarke J. van Wijk. Interactive visualization of state transition systems. *IEEE Transactions on Visualization and Computer Graphics*, 8(4):319–329, October 2002.

[126] Katerina Vrotsou. *Everyday mining Exploring sequences in event-based data*. PhD thesis, Linkoping University, 2010.

[127] Katerina Vrotsou, Kajsa Ellegard, and Matthew Cooper. Everyday Life Discoveries: Mining and Visualizing Activity Patterns in Social Science Diary Data. In *Proceedings of the International Conference on Information Visualization (IV)*, pages 130–138. IEEE, July 2007.

[128] Katerina Vrotsou and Camilla Forsell. A Qualitative Study of Similarity Measures in Event-Based Data. In *Proceedings of the Human Interface and the Management of Information. Interacting with Information Symposium on Human Interface*, pages 170–179. Springer, 2011.

[129] Katerina Vrotsou, Jimmy Johansson, and Matthew Cooper. ActiviTree: interactive visual exploration of sequences in event-based data using graph similarity. *IEEE Transactions on Visualization and Computer Graphics*, 15(6):945–52, 2009.

[130] Taowei David Wang, Amol Deshpande, and Ben Shneiderman. A Temporal Pattern Search Algorithm for Personal History Event Visualization. *IEEE Transactions on Knowledge and Data Engineering*, 22(12):1–12, 2010.

[131] Taowei David Wang, Catherine Plaisant, Alexander J. Quinn, Roman Stanchak, Shawn Murphy, and Ben Shneiderman. Aligning temporal data by sentinel events: discovering patterns in electronic health records. In *Proceedings of the Annual SIGCHI Conference on Human Factors in Computing Systems (CHI)*, pages 457–466. ACM, 2008.

[132] Taowei David Wang, Catherine Plaisant, Ben Shneiderman, Neil Spring, David Roseman, Greg Marchand, Vikramjit Mukherjee, and Mark Smith. Temporal Summaries: Supporting Temporal Categorical Searching, Aggregation and Comparison. *IEEE Transactions on Visualization and Computer Graphics*, 15(6):1049–1056, November 2009.

[133] Taowei David Wang, Krist Wongsuphasawat, Catherine Plaisant, and Ben Shneiderman. Extracting insights from electronic health records: case studies, a visual analytics process model, and design recommendations. *Journal of medical systems*, 35(5):1135–52, October 2011.

[134] Yasuyuki Watai, Toshihiko Yamasaki, and Kiyoharu Aizawa. View-Based Web Page Retrieval using Interactive Sketch Query. In *Proceedings of the IEEE International Conference on Image Processing*, pages 357–360. IEEE, 2007.

[135] Martin Wattenberg. Sketching a graph to query a time-series database. In *Proceedings of the Annual SIGCHI Conference on Human Factors in Computing Systems (CHI) - Extended Abstracts*, pages 381–382. ACM, 2001.

[136] Martin Wattenberg. Visual exploration of multivariate graphs. In *Proceedings of the Annual SIGCHI Conference on Human Factors in Computing Systems (CHI)*, page 811. ACM, 2006.

[137] Marc Weber, Marc Alexa, and Wolfgang Muller. Visualizing time-series on spirals. In *Proceedings of the IEEE Symposium on Information Visualization (InfoVis)*, pages 7–13. IEEE, 2001.

[138] Charles Wetherell and Alfred Shannon. Tidy Drawings of Trees. *IEEE Transactions on Software Engineering*, SE-5(5):514–520, September 1979.

[139] Ryen W. White and Resa A. Roth. Exploratory Search: Beyond the Query-Response Paradigm. In *Synthesis Lectures on Information Concepts, Retrieval, and Services*, pages 1–98, April 2009.

[140] William E. Winkler. The state of record linkage and current research problems. Technical report, Statistical Research Division, U.S. Census Bureau, 1999.

[141] Krist Wongsuphasawat and David Gotz. Outflow: Visualizing patient flow by symptoms and outcome. In *IEEE VisWeek Workshop on Visual Analytics in Healthcare*, pages 25–28, 2011.

[142] Krist Wongsuphasawat, John Alexis Guerra Gómez, Catherine Plaisant, Taowei David Wang, Meirav Taieb-Maimon, and Ben Shneiderman. LifeFlow: Visualizing an Overview of Event Sequences. In *Proceedings of the Annual SIGCHI Conference on Human Factors in Computing Systems (CHI)*, pages 1747–1756. ACM, 2011.

[143] Krist Wongsuphasawat, Catherine Plaisant, Meirav Taieb-Maimon, and Ben Shneiderman. Querying Event Sequences by Exact Match or Similarity Search: Design and Empirical Evaluation. *Interacting with computers*, 24(2):55–68, March 2012.

[144] Krist Wongsuphasawat and Ben Shneiderman. Finding comparable temporal categorical records: A similarity measure with an interactive visualization. In *Proceedings of the IEEE Symposium on Visual Analytics Science and Technology (VAST)*, pages 27–34. IEEE, October 2009.

[145] Moshé M. Zloof. Query by example. In *Proceedings of the National Computer Conference and Exposition (AFIPS)*, pages 431–438. ACM, 1975.

[146] Moshé M. Zloof. Office-by-Example: A business language that unifies data and word processing and electronic mail. *IBM Systems Journal*, 21(3):272–304, 1982.