EventThread: Visual Summarization and Stage Analysis of Event Sequence Data

Shunan Guo, Ke Xu, Rongwen Zhao, David Gotz, Hongyuan Zha, and Nan Cao

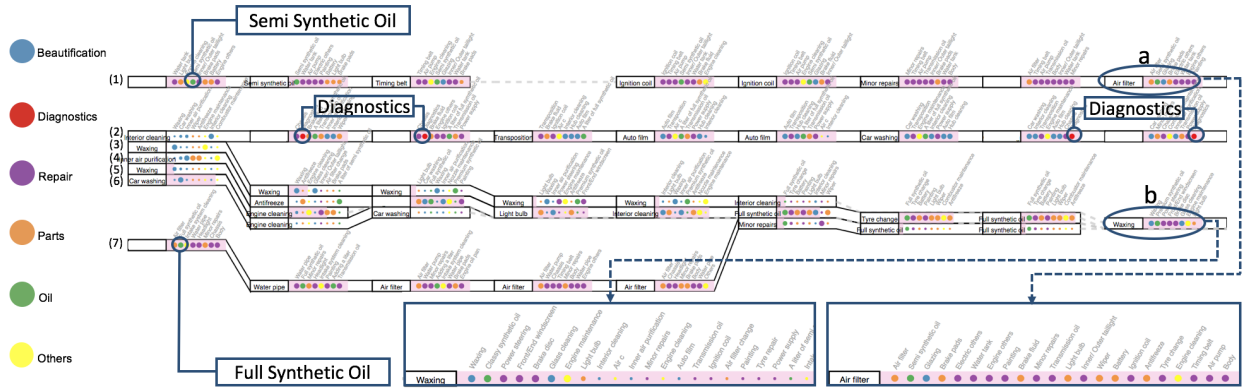

Fig. 1. EventThread system summarizes event sequence data into threads that are segmented into stages and clustered into latent stage categories, thus showing the evolution patterns of the event sequences. This figure shows different car maintenance patterns analyzed based on 1,112 car service records consists of 66 individual car maintenance events in 6 distinct event types. This figure illustrates cars use full-synthetic or premium oil usually have a better condition when compared with the cars use semi-synthetic oil.

Abstract— Event sequence data such as electronic health records, a person's academic records, or car service records, are ordered series of events which have occurred over a period of time. Analyzing collections of event sequences can reveal common or semantically important sequential patterns. For example, event sequence analysis might reveal frequently used care plans for treating a disease, typical publishing patterns of professors, and the patterns of service that result in a well-maintained car. It is challenging, however, to visually explore large numbers of event sequences, or sequences with large numbers of event types. Existing methods focus on extracting explicitly matching patterns of events using statistical analysis to create stages of event progression over time. However, these methods fail to capture latent clusters of similar but not identical evolutions of event sequences. In this paper, we introduce a novel visualization system named EventThread which clusters event sequences into threads based on tensor analysis and visualizes the latent stage categories and evolution patterns by interactively grouping the threads by similarity into time-specific clusters. We demonstrate the effectiveness of EventThread through usage scenarios in three different application domains and via interviews with an expert user.

Index Terms—Visual Knowledge Representation, Visual Knowledge Discovery, Data Clustering, Time Series Data, Illustrative Visualization

1 Introduction

Event sequences, such as those found in electronic health records, person's academic behaviors, or car service records, record an ordered series of events which have occurred over a period of time. For instance, electronic health records contain sequences of timestamped medical events (e.g., diagnoses, lab tests) for specific patients that are recorded over the course of a clinical process. Similarly, data capturing the career behaviors for an employee can include a series of milestone events (e.g., educational attainment, promotions, performance awards) while the service records of a car can consist of dates and types of service (e.g., oil changes and tire replacements) over a vehicle's lifespan.

Due to the importance and wide areas of application for event sequence data, a rich variety of both analysis and visualization techniques have been developed to help yield insights from this form of data. In particular, a variety of proposed analysis-based approaches have focused on either (1) summarization of large-scale event sequence data [3, 4, 18, 28], or (2) the extraction of latent "milestone" stages that occur frequently within a set of event sequences [9, 10, 15]. They produce highly summarized results which can highlight interesting high-level structures, but often fail to show important low-level event details (such as the raw, individual event features that contribute to an aggregate summarization) which can help in the crucial task of semantic interpretation of the discovered structures.

In contrast, much of the event sequence visualization designs [12, 34, 35]) focus on precisely capture details about how individual events occur in sequence over time. This has led to recent methods which focus on prioritization [12] or simplification [23] to enable these approaches to scale to the complexity required for many real-world tasks. Yet even in these cases, the visualized paths of event sequences are closely tied

- *Shunan Guo is with East China Normal University. E-mail: g.shunan@gmail.com*
- *Ke Xu is with the Hong Kong University of Science and Technology. E-mail: kxuak@connect.ust.hk*
- *Rongwen Zhao is with the iDV^x Lab, Tongji University. Email: rongwen.zhao@tongji.edu.cn*
- *David Gotz is with University of North Carolina at Chapel Hill. Email: gotz@unc.edu*
- *Hongyuan Zha is with East China Normal University. Email: zha@sei.ecnu.edu.cn*
- *Nan Cao is with the iDV^x Lab, Tongji University and is the corresponding author. Email: nan.cao@tongji.edu.cn*

This article is a preprint. An official version of this article will appear at IEEE VIS 2017 and as an article in IEEE Transactions on Visualization and Computer Graphics (TVCG). Please cite the TVCG article when referencing this manuscript.

to the low-level representation of individual events or sub-sequences, which makes it difficult to discover or understand higher-level structures within the data [11].

Therefore, there is a gap between the capabilities of existing analysis and visualization techniques designed for temporal event sequence data. A desired visual analysis system should be able to discover and communicate latent high-level structures within a complex collection of event sequence data, while at the same time providing users with information about the low-level events and sub-sequences of events which characterize those structures to support semantic interpretation of the findings.

However, designing such a system is a complex problem due to the following challenges: First, the analysis of sequences with a large number of different event types, occurring in different orders, and in sequences of different lengths, must occur within an enormously complex data space. It requires techniques that can transform large-scale heterogeneous event sequence data into an uniform data model without losing detailed information. Second, the methods designed to detect and represent high-level latent structures (e.g., the overall clinical pathway of a specific cohort) must be designed to include sufficient relevant context to enable low-level semantic interpretation of what those structures represent (e.g., detailed events in each extracted clinical paths). This requires the design of an analysis algorithm that automatically associate and correlate the analysis results (i.e., the high level structures) with the detailed data used to generate them during the computational process. Third, because there is often little ground truth information available to validate the results, it is important to create methods which allow users to tune parameters to experiment with the sensitivity of the extracted structures and their explanations to changes in parametrization.

To address the above challenges, we introduce EventThread, a comprehensive and integrated visual analysis system that is designed and developed for visually summarizing large scale and high-dimensional event sequence data. It allows interactive exploration of analytically discovered innate latent stages.The system is designed to support a number of event sequence analysis tasks including: clustering, pattern discovery, and stage analysis. Moreover, the system provides a suite of rich contextual visualizations to help with interpretation and data exploration. In its primary approach to this analysis problem panel, the system groups event sequences into representative *threads* based on tensor analysis in an offline procedure. The threads are further grouped into latent stage categories at different stages (time periods) based on an optimization-based layout algorithm and an interactive online clustering algorithm. A novel visualization is designed to represent these threads and their co-evolution over time, and linked side views showing contextual information that is coordinated with users' visual selections. More specifically, this paper describes the following contributions:

- **System.** We introduce a dynamic visual analysis system based on event sequence clustering via tensor decomposition to provide a visual summarization of event sequence data. This approach detects latent stages and captures the evolution of these stages with respect to the detailed features of supporting events.
- **Analysis.** We introduce an unsupervised analysis algorithm and the corresponding data modeling and transformation techniques for detecting sequential clusters of highly summarized and semantically meaningful latent stages from event sequence data. In particular, we transform the raw data into a tensor and preform a back-end orthogonal tensor analysis to derive event threads. These threads are a form of temporal clusters which can be traced from one time stage to the next, and adjusted via an interactive online clustering algorithm.
- **Visualization.** We employ a novel visualization metaphor which adopts a line map design which seamlessly represents the overlapping and interacting semantic concepts associated with the threads are various stages of time. An optimization-based layout algorithm is proposed which balances between the similarity of threads and the crossing of lines to place the visual representation of each thread at a proper position at each stage.
- **Evaluation.** We demonstrate the effectiveness of EventThread system through usage scenarios with real-world data in three distinct application domains. We also report feedback from an interview with an expert from the medical domain.

2 Related Work

This section provides an overview of research that is most closely related to our work, including (1) analysis methods for event sequence summarization and stage identification, and (2) techniques for event sequence visualization.

2.1 Event Sequence Analysis

Various analytical methods have been developed to support the analysis of event sequence data. These methods target a variety of different analysis applications including event sequence clustering, classification, pattern discovery, and prediction. We summarize a variety of methods focused on two specific types of problems most relevant to this paper: event summarization, and stage analysis.

Event Sequence Summarization. Generally speaking, the objective of event sequence summarization is to find the best method for grouping similar sequences based on their semantic content and proximity on the timeline [28]. Such summarization methods can help users perform more efficient pattern discovery and minimize the effort required for data comparison. For example, Osato et al. [26] used BLAST to compare similarity between cDNAs in order to cluster them into distinct family genes. Huang et al. [14] partitioned event logs in medical data into optimal time intervals and summarizes the segments horizontally so as to reveal common patterns within multiple clinical pathways. Mori et al. [24] utilized Hidden Markov Models to directly cluster segments of human behavioral records on social media for the purpose of summarizing human daily life.

A variety of different approaches have been taken to address the event sequence summarization challenge. For example, Pham et al. [28] proposed an event substitution algorithm which in each time interval replaces a variety of related events with a single representation. This approach leverages a pre-defined taxonomy of the event domain, which can be difficult to define in many cases. Kiernan et al. [18] introduced a different approach in which each segment's events of different types are grouped based on their frequency of occurrence in the segment. A summarization tool EventSummarizer [19] was further developed to support event query. However, this method was initially designed to summarize database log records, and only one event type is contained in each user's sequence. This greatly limits the applicability of this approach to other application domains. More sophisticated summarization methods often employing clustering algorithms. A commonly used tool for clustering event sequence data of variable length is Hidden Markov Models (HMM). However, both standard HMM-based models [22, 25, 31] and mixture models of HMM [3, 4] have the drawback that each event sequence can only be assigned to one cluster. Yet, in real-world data, sequences may evolve over time and cluster in different ways at different stages. Moreover, the results of these model-based methods are so highly summarized that they do not effectively support user-driven exploratory analysis. More specifically, users are unable to drill down to the detailed event-level features and required to identify the semantic meaning of an analysis result. This leads to great difficulty in result validation and evaluation. In our work, therefore, we develop a tensor-based summarization method which provides time-based clustering, as well as the ability to retrieve detailed event-level features that support user-driven exploratory analysis and interpretation.

Stage Analysis. A variety of methods have been proposed to detect underlying stages (i.e., states) within a large collection of event sequence data. Such stage analysis methods have great potential value in extracting higher-level representations of event sequence progression. For example, considering medical event data where stages might represent disease states and their evolution over time. This stage information can potentially help doctors with more accurate and early diagnosis, treatment planning, and risk management. However, traditional staging in methods [9, 15, 30, 32, 33] largely rely on subjective assessments, resulting in great imprecision. Recent efforts have tried to address this by modeling disease progression using machine learning and statistical techniques based on observed medical records. For example, Fonteijn

et al. [10] applied probabilistic classification to familial Alzheimers disease and Huntingtons disease. Jackson et al. [16] developed a multi-stage Hidden Markov Model to analyze aneurysm screening. Cohen et al. [7] performed hierarchical clustering using minute-by-minute physiological, clinical and treatment variables to identify patient states. Zhou et al. [39] proposed a fused group lasso formulation with given biomarkers. In research most relevant to our work, Yang et al. [36] modeled patient records as time sequences and subsequences with classification labels, then utilized EM algorithm to soft assign each subsequence to stages. Although these methods succeed in segmenting event sequences into several stages, they fail to reveal the diversity of subsequence event features within the same stage. This makes assessment and interpretation of the results very difficult if not impossible. The method proposed in our work offers a more scalable and interpretable result. Our method combine both summarization of sequences with detailed explanatory information about the composition and variability of detected stages.

2.2 Event Sequence Visualization

A wide variety of methods have been designed to visualize temporal data. While many of these techniques focus on time series data [1], the area most relevant to our work is event sequence data visualization. The focus of these methods is on visual summarization of event data, often in ways that scale effectively for datasets with many sequences and with large numbers of event types. These methods can be broadly categorized into two types: (1) flow-based visualizations and (2) the matrix-based visualizations.

Flow-based Approaches. A number of efforts have focused on visualizing temporal event sequence data using basic timeline metaphors. For example, Lifelines [29], CloudLines [21], and other timeline-based designs [2, 8, 17] simply displayed the original sequence data along a common time axis. Although these methods can provide users with detailed event path of individuals, aggregated analysis of sequence groups such as pattern discovery is difficult to carry out. To overcome this limitation, flow-based techniques similar to Sankey Diagrams have been adopted along with aggregation methods in visualizations such as LifeFlow [35], Outflow [34] and EventFlow [23]. These techniques successfully aggregated large numbers of event sequences, but cannot effectively handle datasets with large numbers of event types and/or long sequences. Decisionflow [12] supported the analysis of event sequences containing a very large amount of event types via rich interactions. It has proven to be efficient in a variety of application scenarios including medical event sequence exploration. However, like the other flow-based methods, it still captures patterns only based on the occurrence of specific individual events, and fails to summarize latent patterns or stages within the data, making it hard to identify certain types of patterns [11].

Matrix-based Approaches. As an alternative to flow-based methods, a number of visualizations have adopted matrix-based techniques. These visualizations utilize matrix-based icons to provide visual links and comparisons between multiple events. For example, MatrixFlow [27] used adjacency matrices to display the co-occurring clinical events within a period of time. User-adjustable time parameters provide limited control over the temporal granularity of the visualization, which allow doctors to observe the temporal evolution of symptoms. However, this technique does not reveal latent states or their transitions as symptoms progress. MatrixWave [38] augmented traditional Sankey Diagrams with adjacency matrices to depict visual links between nodes in neighboring layers, and matrices of different layers through a sophisticated layout method. Grids within the matrices were also adopted in various ways to represent proportions of cohort transition. Egolines [37], which most directly inspired our visual design, used a combination of aggregated flow-based and matrix-based methods, along with a subway metaphor to present temporal patterns in dynamic ego-networks. Although our work has a similar visual design, our method are different from Egolines in two critical aspects. First, our work aims to visualize a fundamentally different type of data. We focus on event sequence data, while Egolines is designed for egocentric networks in which the focus is on central actor and all patterns surround it. Second, the temporal threads in Egolines all extend horizontally along the full extend of the time axis. In contrast, our approach by necessity must handle vertical structures which represent latent stage categories. This requires a more sophisticated visual layout.

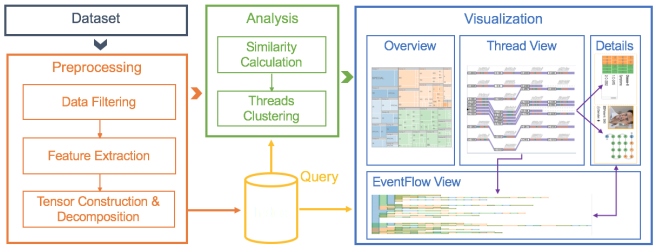

Fig. 2. EventThread system overview and data processing pipeline.

3 System Overview

Our system is motivated by the real world analysis requirements from the health-care domain. Based on discussions with experts and ongoing collaborative research with experts in the field of medical data analysis, a variety of design requirements were identified. These requirements, as well as preliminary designs created to meet these requirements, were discussed regularly over a four-month period resulting in iterative improvements to the final design. Based on these interactions and discussions, as well as a review of related work as summarized above, a list of the most critical requirements (R1-R4) that guide the design of our solution is as follows:

- **R1 Dealing with noisy data.** Arrange events of high heterogeneity and sequences of variable length to reveal the exact order of event occurrence before data analysis.
- **R2 Summarizing data via cluster analysis.** Group similar event sequences (i.e.,sequences with similar events occurring in similar order) into summarized views of latent patterns. Similar threads should be further grouped at each stage to derive higher-level latent patterns to illustrate meaningful temporal structures.
- **R3 Interpreting analysis results in context.** Illustrate the analysis results (i.e., the threads and higher-level structures) in the context of detailed low-level event data to facilitate result interpretation.
- **R4 Keep humans in the analysis loop.** Due to a lack of ground truth, it is important to help users make adjustments and explore the impact of different clustering choices during the analysis.

Based on these requirements, we have designed the EventThread system to summarize event sequence data into a meaningful form. The system, as shown in Fig 2, consists of three major modules: (1) the prepossessing module, (2) the analysis module, and (3) the visualization module. The prepossessing module transforms the heterogeneous raw event sequence data into a uniform format (i.e, a three way data tensor capturing the multi-way structure of "Instance-Time-Event") to meet **R1**. The analysis derives latent threads by clustering the raw event sequences based on tensor analysis which also produces the latent context information that is used for detecting higher-level structures, which we call latent stage categories. This supports **R2**. The visualization module represents the analysis results using multiple coordinated views which show the analysis results within a multifaceted context to support **R3**. Interactions with the visualizations are designed to let users easily explore the data and adjust the analysis parameters, supporting **R4**.

4 Event Sequence Summarization

In this section, we first introduce the data model and preprocessing methods designed to uniformly represent the heterogeneous event sequence data. We then describe a tensor-based pattern analysis technique developed to summarize the event sequence data.

4.1 Data Model and Transformation

An event sequence associated to an entity is an ordered series of events which occurred over a period of time. Here, the entity is an object around which the sequence is produced or generated. For example, in electronic health records, the entity represents a patient around which the events such as diagnosis and labs are made; in the car service

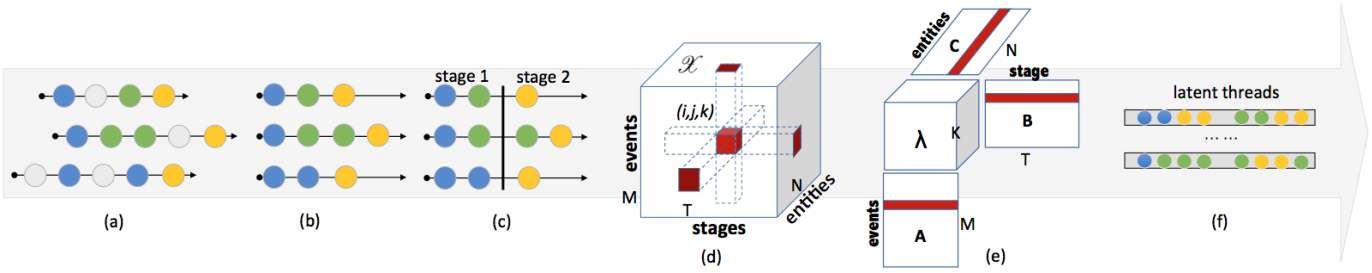

Fig. 3. Data transformation and analysis pipeline: (a) data filtering, (b) sequence alignment, (c) event folding, (d) data modeling, (e) tensor decomposition, and (f) latent threads extraction.

dataset, the entity is a car around which the services such as waxing and repairing are made. We transform collections of heterogeneous event sequence data (with different lengths, event types, and event orders) into a tensor-based data model via four major steps, as illustrated in Fig. 3(a - d). In particular, we first filter events in the sequences to reduce data noise. Then, we align the sequences based on event occurrence order. Third, the aligned event sequences are segmented into stages for summarization. Finally, the stages are transformed into a tensor.

Data Filtering. Real world event sequence data is often very noisy, containing large numbers of event types of which many occur sparsely and with limited frequency. To capture the primary sequential patterns within this raw set of event sequence data, it is important to minimize the data noise and only select the events that best represent a sequence to capture its innate characteristics. To this end, we employ Term Frequency - Inverse Document Frequency (*TF-IDF*) [6], a technique used frequently in text mining, to measure the importance of individual events. In text mining, the *TF-IDF* score of a word in a document reflects the word's importance in terms of clearly separating the document from others in the corpus. Specifically, a word having a high *TF-IDF* score in a document indicates it has high frequencies in the focal document when compared to the overall corpus, and is therefore representative of the document. An event sequence can be characterized by characteristic events in a similar way. Here, an event sequence in a dataset is analogous to a document in a collection. Thus, an event within an event sequence corresponds to a word in a document. Based on this conceptual mapping, we are able to use *TF-IDF* to estimate the importance of each event. This estimate is then used to determine which events best differentiate the sequence from others. In our implementation, n-gram (n = 1,2) is used in the calculation, and histogram is used for showing the distribution, based on which a set of events with significantly low *TF-IDF* scores are treated as noise and removed.

Sequence Alignment. Event sequence data captures a series of events and the times that they occur over a period of time. Often, only the relative time and the order of the sequence instead of the specific occurrence time is meaningful in terms of capturing the primary sequential patterns. For example, it is the order and duration of different interventions which matters when treating a disease, rather than the specific dates on which the interventions are performed. Therefore, we shift the event sequences and align them along a common origin time axis as shown in Fig. 3(b).

Stage Folding. We further segment the aligned event sequences into multiple stages, within which the events are folded together to provide a higher-level summarization. This sequence of stages is shorter in length than the original sequence, which is particularly useful for dealing with large-scale event sequence data collected over long periods of time. In our implementation, the sequences are segmented and the events are folded based on either fixed time intervals (e.g., a year, a month or a day) which can be selected depending on typical event occurrence frequency or the number of major event that has occurred (e.g., the course of treatment, the round of car maintenance).

Data Modeling. After the above prepossessing steps, we capture the key features of the transformed event sequence data within a tensor-based data model. A *tensor*, denoted as $\mathscr{X}$, is a multidimensional array which extends the concepts of scalars (denoted as x), vectors (denoted as $\mathbf{x}$), and matrices (denoted as $\mathbf{X}$) to higher dimensions. We refer to a tensor with n-dimensions as an n-way tensor, where the dimensionality is determined by the number of elements contained within the tensor. For example, the three-way tensor $\mathscr{X} \in R_+^{N_1 \times N_2 \times N_3}$ has three dimensions, corresponding to N_1, N_2, and N_3. The notation R_+ indicates that all the elements of $\mathscr{X}$ contain non-negative values, which reflects that the values correspond to numbers of observed instances.

With the above definitions, the processed event sequence data can be represented by a 3-way tensor $\mathscr{X} \in R_+^{M \times T \times N}$ (as shown in Fig. 3(d)) where M, T, N respectively indicate the number of *events*, *stages*, and *entities*. In this way, $\mathscr{X}_{ijk}$ indicates the i-th event to have occurred in the j-stage for the k-th entity.

4.2 Summarization Algorithm

We decompose the tensor of an event sequence dataset, $\mathscr{X}$, to derive latent sequential patterns, namely *threads* (Fig. 3(f)). Intuitively, each thread represents a summarized view of a cluster of similar event sequences. The event sequence clusters are derived through the decomposition process with the aim of best representing the variation within the dataset.

The *tensor decomposition* process (Fig. 3(e)) factorizes the tensor $\mathscr{X}$ into the product of multiple components: a core tensor, and a set of factor matrices (one for each dimension). We seek to minimize the error between $\mathscr{X}$ and the product of these components. Formally, the decomposition can be described as an optimization problem as follows:

$$min\,||\mathscr{X} - [\![\lambda; \mathbf{A}, \mathbf{B}, \mathbf{C}]\!]||^2$$

$$\text{Subject to: } \mathbf{B}^T\mathbf{B} = \mathbf{I}, \mathbf{A}, \mathbf{B}, \mathbf{C} \in R_+^{\{N,T,M\} \times K}$$

where $\mathscr{X} \in R_+^{N \times T \times M}$ is the aforementioned three-way tensor that models a raw event sequence dataset. $[\![\cdot]\!]$ denotes the CP tensor decomposition algorithm [13], which produces factor matrices $\mathbf{A}$, $\mathbf{B}$, and $\mathbf{C}$ in shapes of $N \times K$, $T \times K$, and $M \times K$, with K as an input parameter indicating the number of latent threads. $\mathbf{A}$, $\mathbf{B}$, and $\mathbf{C}$ respectively represent the distributions of the threads over events, stages, and entities. In particular, $\mathbf{B}_{tk}$ indicates the likelihood of the k-th thread occurred at stage t; $\mathbf{A}_{ik}$ and $\mathbf{C}_{jk}$ indicate the relevance between the k-th thread and the i-th events and j-th entities, respectively. $\lambda \in R_+^{K \times K \times K}$ is the core tensor whose diagonal elements indicate the importance of the corresponding thread. Constraint $\mathbf{B}^T\mathbf{B} = \mathbf{I}$ ensures the orthogonality of the threads, so that their meanings can be easily interpreted.

Intuitively speaking, this analysis process can be viewed as somewhat analogous to topic modeling in text analysis. Just as topic analysis derives latent topics from a document collection, the above analysis derives latent threads from a collection of event sequences. In contrast to topic modeling, however, where a topic can be interpreted as a set of keywords, the above tensor-based approach takes a multi-way structure into consideration. This allows for a much richer interpretation of the latent threads three different aspects: stages, events, and entities.

5 Visualization

This section presents a set of design tasks that guide our visualization design, and the specific visualizations created to support these tasks.

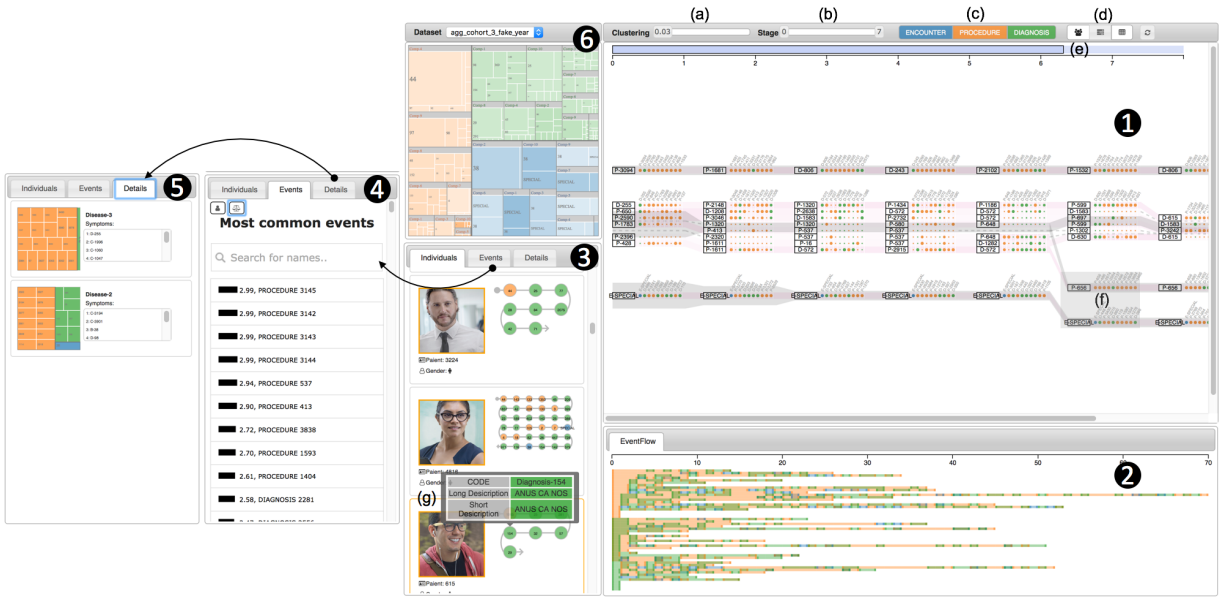

Fig. 4. The EventThread system contains six interactively coordinated views, including (1) a threads view, (2) an event flow view, (3) an entity list view, (4) an event list view, (5) a thread list view and (6) a global overview. The clustering level of threads can be adjusted through (a) the cluster slider. Users can choose to display and hide stage phases via (b) the stage slider, and (c) domain-specific event types. Entity proportions of latent stage categories and threads can be revealed by (d) adding backgrounds. Event description can be obtained via (g) informative tooltips. Other useful techniques for analysis are also available, including (e) zooming in on the timeline and (f) brushing components across specific stages.

5.1 Design Tasks

The design of visualizations to represent the above analysis results have been guided by a set of required design tasks. These tasks, in turn, are based on the requirements outlined in R1-R4. Generally, a desired visualization tool should support a simultaneous illustration and exploration of the overall evolution of the threads, as well as information to support interpretation of corresponding events and entities. We formalize these goals through the following design tasks:

- **T1 Show summarized sequential patterns in context.** The visualization should be designed to clearly reveal the summarized sequential patterns (i.e., the latent threads) within the context of corresponding stages, events and entities to provide interpretable overviews of the patterns.
- **T2 Differentiate between different thread evolution patterns.** The visualization should be able to reveal the details about how threads evolve from stage to stage. Groups of threads may relate to similar sets of events and entities at various stages, so the visualization should be able to group similar threads and separate threads that are more distinct. Moreover, these groups can evolve from one stage to another.
- **T3 Promote pattern inspection and comparison.** The visualization design should support rich interactions and coordinated views to help users inspect and compare the visualized threads.
- **T4 Provide easy access to raw event data.** While high-level summarization of temporal patterns is essential, the visualization should also be able to communicate the raw event sequences corresponding to those patterns. This is critical for both interpretation and validation of the discovered patterns.

5.2 User Interface

Guided by the above tasks, we design the user interface of EventThread system as primary thread view (Fig. 4(1)) surrounded by a set of coordinated side views which provide contextual information from different perspectives (**T1**). In particular, when a selection is made in the thread view, the context views are updated to show details about the selected threads or thread stages. More specifically, the event flow view (Fig. 4(2)) shows an aggregate representation of raw event sequence data for the selection; the entity list (Fig. 4(3)) shows the corresponding entities (captured in factor matrix **C**) and their profiles; the event list (Fig. 4(4)) illustrates the most frequently occurring events within the selection; and the thread list view (Fig. 4(5)) shows the event type distributions within the selected threads for comparison via treemaps (**T2**). Complimenting these selection-driven contextual views, an overall overview is also provided (Fig. 4(6)) to illustrate the overall event distribution cross the full set of the threads in factor matrix **A**. The color-coding palettes are consistent across different views, with each color representing a specific category of event types.

5.3 Thread View

In this subsection, we introduce the design of the thread view and the corresponding layout algorithms.

5.3.1 Visualization Design and Encoding

The thread view, as shown in Fig. 5, is designed to visually summarize the key concepts (threads, stages, events, and entities) derived from either the raw data or the tensor analysis results. This is done within one integrated visual representation which reveals the evolution patterns of event sequences over different stages. The design details for each part of the thread view are as follows.

Threads. We visualize the threads derived by tensor analysis as segmented linear stripes, following a line map metaphor. The threads link the key concepts of stage, events, and entities together. Each thread (Fig. 5(a)) is equally divided into several segments based on the number of stages recorded in the tensor (Fig. 5(b)). The color opacity of the thread in each stage represents the thread's likelihood of occurrence, which is given by the corresponding column vector of factor matrix **B**. When the likelihood is zero for a given stage, the thread will be visualized as a dashed-line as shown in Fig. 5(e), indicating the temporary absence of any corresponding events at that time for that thread. This visual thread design aims to providing a concise overview of the summarized sequential patterns and their probabilities of occurrence at different stages (**T1**).

Stages. In our design, a stage represents a fixed, temporal unit that is recorded in the tensor (see Section 4.1) and visually represented as a line segment in a thread (Fig. 5(b)). The segments within different threads at the same stage can be further grouped into latent stage categories based on similarities of the events that are likely to occur in the stage. A sophisticated layout algorithm balancing between thread similarity and line crossing is developed and applied to place the threads in their

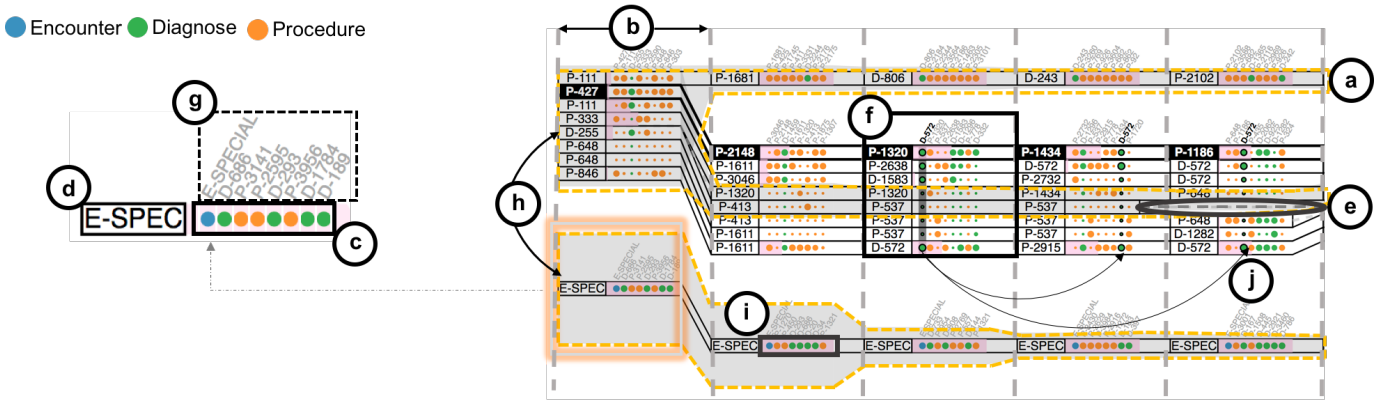

Fig. 5. The thread visualization design shows information for threads, stages, events and entities as derived from the raw data through tensor analysis. Threads are shown as lines (a), and are divided into several stages (b) with each segment containing nodes (c) indicating events with high likelihood of occurrence. Nodes are labeled with the event type with the highest likelihood (d). Threads having zero occurrence likelihood at a certain stage are encoded with dashed-lines (e). Threads in each stage are further clustered into latent stage categories (f), with labels showing the names of the most likely events to be found inside the threads (g). The entire view is under-laid with a gray background whose height (h) indicates the number of entities involved in a latent category or thread. A process bar (i) is used to encode the proportion of entities in a particular thread with respect to the cluster. Interactions such as selection and coordinated highlighting of events (j) are also employed.

positions. The layout algorithm is discussed in Section 5.3.2. This design helps users identify latent stage categories within each stage, as well as distinguish between different evolution patterns as evidenced by the merging and diverging of categories. (**T2**).

Events. The set of events occurring within a thread at a given stage is the most important contextual information available to help interpret the meaning of a thread. To make this information visible, event occurrence probabilities are visually encoded as part of the thread representation in each stage (**T1**). As mentioned in 4.2, each event i has a likelihood of occurrence for the stage t in the thread k, which is given by $(\mathbf{A}_{ik}\mathbf{B}_{tk})$. As shown in Fig. 5(c), circular nodes within each thread are used to represent the events with the highest occurrence likelihood in a given stage, with the circle size indicating the likelihood and the circle color corresponding to the event type. For each latent stage category (i.e., each cluster of thread stages), the displayed events are selected by considering their overall likelihood of occurrence across all threads in the category. Events for each stage category are vertically aligned to allow comparison, with a label at the top for identification (Fig. 5(g)). Events are sorted from left to right beginning with the highest likelihood of occurrence. The most likely event is used as the segment's label (Fig. 5(d)) for emphasis. The label is left blank if the segment contains multiple likely events.This visual design, together with interactive highlighting (described in Section 5.5), helps users track events across the threads.

Entities. Entities are visualized in our design as light grey backgrounds under-laid beneath the threads (Fig. 5(h)). The height of grey backgrounds, designed to reveal the volume of entities flowing along each branch in the visualization, correspond to the number of entities associated with the combined set of threads in a given category (**T1**). This number is given by $(\mathbf{C}_{ik}\mathbf{B}_{tk})$ with $\mathbf{C}_{ik}$ indicates the relevance between the i-th entity and the k-th thread and $\mathbf{B}_{tk}$ indicates the likelihood of the k-th thread occurred at the t-th stage. The backgrounds are optional, and when enabled the thread layout adapts thread positions to account for the variations in height across thread categories (Fig. 5). To communicate the proportion of entities involved in each thread within a stage category, we use the length of a small bar (Fig. 5(i)) to indicate the percentage of entities in the category which come from a particular thread.

5.3.2 Layout Algorithm

To layout the threads, we first fix the x-positions of the stages by equally dividing the width of the display. After that, a novel layout algorithm groups threads together based on their event similarities at each stage, while minimizing during layout the crossing of the thread stripes to reduce visual clutter. These factors determine the y-positions of each thread at each stage. These layout constraints are formulated

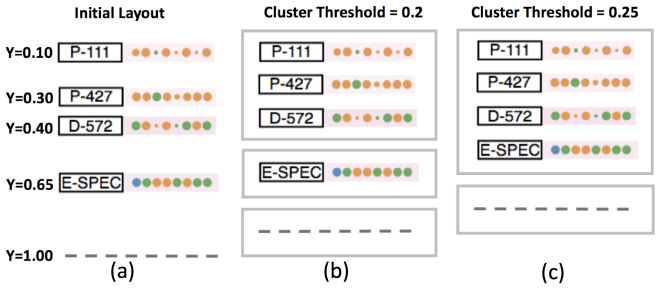

Fig. 6. Users can interactively adjust the clustering level to group threads into different latent stage categories.

as an optimization problem, in which the following layout energy is minimized:

$$\sum_{t=0}^{T}(\alpha\sum_{i<j}w_{ij}(t)\left\|y_i(t)-y_j(t)\right\|^2+(1-\alpha)\sum_i\|y_i(t)-y_i(t-1)\|^2)$$

where $y_i(t)$ indicates the vertical position of a thread at the stage t and $w_{ij}(t)$ indicates the event similarity between thread i and j at a given stage t by using inverse quadratic Euclidean Distance,

$$w_{ij}(t)=\frac{1}{\mathrm{d}(\mathbf{v_i}(\mathbf{t}),\mathbf{v_j}(\mathbf{t}))^2}$$

where $v_i(t)$ is the feature vector of thread with m events i in stage t which is of the form $v_i(t)$=($\mathbf{A}_{1i}\mathbf{B}_{ti}$,$\mathbf{A}_{2i}\mathbf{B}_{ti}$,...,$\mathbf{A}_{mi}\mathbf{B}_{ti}$). Intuitively, the first term minimizes the Euclidean distances between pairs of threads and will group threads with larger event similarity (i.e., have a larger $w_{ij}(t)$ value, which can be solved based on spectrum analysis [20]). The second term reduces thread crossings by minimizing the differences between the vertical positions for the same thread at neighboring stages. These two terms are balanced by a parameter $\alpha \in [0,1]$. The overall optimization problem is solved using an iteratively procedure in which thread segments in a pair of succeeding stages (i.e., stages t and $t+1$, or t and $t-1$) are laid out in each iteration, stopping only when the energy term has converged.

Based on the y-positions calculated above, the threads are vertically placed in rows on the display as shown in Fig. 6(a). Users can further cluster the thread segments within the same stage into latent stage categories by interactively adjusting a distance threshold t via the slider shown Fig. 4(a). The neighboring threads whose vertical distances at a given stage are smaller than the threshold will be clustered together within the corresponding stage as shown in Fig. 6(b).

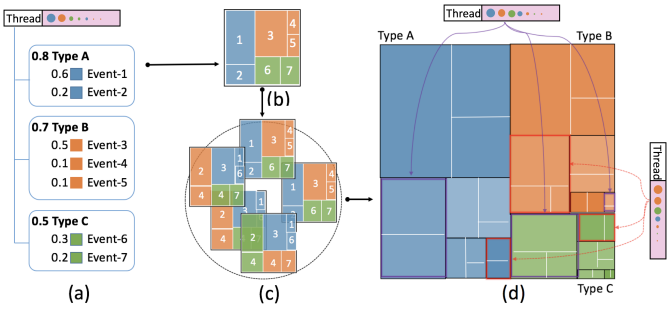

Fig. 7. The visual encoding of overview panel. (a) Each generated thread is described by its major events with corresponding occurrence probabilities and classified by event type. (b) Classified events are depicted by a small thread icon, which are then (c) packed together to generate the overview. (d) The final overview is created by splitting and re-grouping the thread icons by event type.

5.4 Context Views

The interface includes four contextual views including an overview and thread list view, an entity list view, an event list view, and an event flow view. The thread list and overview allow for comparisons of the events associated with each thread. The entity and event list views are used to examine raw, low-level data to aid in interpretation. The event flow view shows an aggregate representation of the raw event sequence data that scales to large number of event sequences. These views are coordinated with selections in the thread view, facilitating pattern inspection, comparison, and interpretation (**T3,T4**).

Overview and Thread List View. We employ the visual design of DICON [5] in the overview panel (as shown in Fig. 4(6)) to simplify the comparison of event occurrence probabilities across threads (**T3**). Each thread is described by a set of events and associated probabilities, which is given by the corresponding column vector in factor matrix **A**. We select the top 10 types of events which are most relevant to the thread (as shown in Fig. 7(a)), and normalize the range of probability to the interval [0,1] before gathering them into a treemap-like representation (as shown in Fig. 7(b)). The thread icons are further packed together as shown in Fig. 7(d), with the color opacity encoding the threads occurrence probability given by summing the values of the corresponding column vector in factor matrix **A**.

Furthermore, we display the tree-map icon of each individual thread (as shown in Fig. 7(b)) in thread list view (as shown in Fig. 4(5)). This allows users to compare threads across various stages. When users select thread segments in the thread view, the most relevant events are ranked by summing up the relevance degree of each event i in stage t, which is given by $(\mathbf{A}_{ik}\mathbf{B}_{tk})$. A full list of ranked events is also provided beside the icon of each thread.

Entity List. The entity list view provides users with easy access to entity profiles and raw event sequences (**T4**). As illustrated in Fig. 4(3), each list item contains the entity id and other basic information. An intuitive node-link representation of the raw event sequence data for the entity is also displayed, with node color indicating event types, and event name at the top of each node.

Event List. The event list (**T4**) provides even more details about the underlying event data behind the selected thread stages in the thread view. Events can be ranked by either the number of entities in which they appear, or the summed probabilities across the selected threads. A horizontal bar visually encodes the value to enable quick comparisons. An event search is also provided, allowing users to direct query for a specific event.

Event Flow. An EventFlow-based design [23] is used to provide an aggregate representation of the raw event sequences involved in the selected thread segments. As shown in Fig. 4(2), each horizontal line depicts one entity's event sequence during the selected stages. The colors of the line segments in this view are used to encode event types, and the length is proportional to the time span of the event sequence. In addition, we mark the beginning and the end of event type regions with a dark, vertical bar to help users identify event type transitions.

5.5 Interaction

To allow users to explore the data from different perspectives, we augmented the visualizations within EventThread with several interactive capabilities.

Query and Filter. Users can explore different data sets through the query box, and apply filters to focus on specific event types by clicking on the event type buttons as shown in Fig. 4(c). The Stage slider (as shown in Fig. 4(b)) can be used to focus an analysis on select stages of interest.

Highlights and Tooltips. When the mouse hovers over a thread node, the corresponding column in the latent category is also visually emphasized (as shown in Fig. 5(f)) to help users identify the event name. At the same time, all other occurrences of same event (the event associated with the hovered node) will also be highlighted regardless of where they appear. This helps users uncover events that repeat across different sections of the thread view (as shown in Fig. 5(j)). Similarly, when users hover a thread or the thread label in one stage, the entire thread across all stages will be highlighted (as shown by second thread from the top in Fig. 5).

Users can also choose to overlay grids on the threads to help separate nodes to during visual scanning and inspection (as shown in Fig. 9). Finally, hovering over any visual element in a visualization triggers the display of tooltips with detailed descriptive information about the underlying events (as shown in Fig. 4(g)). This feature is essential for helping users decode complex datasets which contain many thousands of coded event types (**T4**).

Brush and Zoom. Users can brush several threads segments in the thread view (as shown in Fig. 4(f)), and in response the coordinated contextual view panels will be updated to display corresponding information. The information will be based on the entities and raw event sequence data associated with the selected segments. Users can also zoom in on the thread view by selecting a range of stages in the interactive timeline (as shown in Fig. 4(e)).

Adjust Clustering Threshold Level. The amount of clustering applied to the threads at each stage can be adjusted through cluster slider on the top of the thread view (as shown in Fig. 4(a)). Users would typically adjust the slider to find the most meaningful category groupings among the threads, or dynamically move the slider to understand the sensitivity of the thread groupings.

6 Evaluation

The EventThread design provides a unique approach to event sequence summarization, using tensor analysis to help extract latent high-level structures. We evaluate this approach in multiple ways. First, we report results from three usage scenarios where EventThread was applied to three real-world datasets in three distinct application domains: a cohort of electronic medical records, a corpus of car maintenance records, and a dataset recording progress of professor's acedemic behaviors. We then report qualitative feedback on our system gathered from an expert user in the medical domain.

6.1 Usage Scenarios

In this section, we report the results from three real usage scenarios to demonstrate the ways in which EventThread can help users find patterns in real-world event sequence datasets. Each scenario is performed in a distinct domain using datasets that have very different properties (numbers of events, types of events, length of sequences). Yet in all three scenarios, EventThread was able to help users identify domain-relevant insights. In each scenario, we manually selected the key parameters such as the number of threads and stages in the data preprocessing phase to best reveal innate sequential event patterns of the underlying data.

6.1.1 Usage Scenario I: Analysis of COPD Cohort

In our first usage scenario, we used EventThread to analyze event sequence data representing the medical records for a cohort of 5,804 chronic obstructive pulmonary disease (COPD) patients. The dataset included timestamped events representing diagnoses, procedures, and encounters (i.e., hospital admission and discharge events). The data

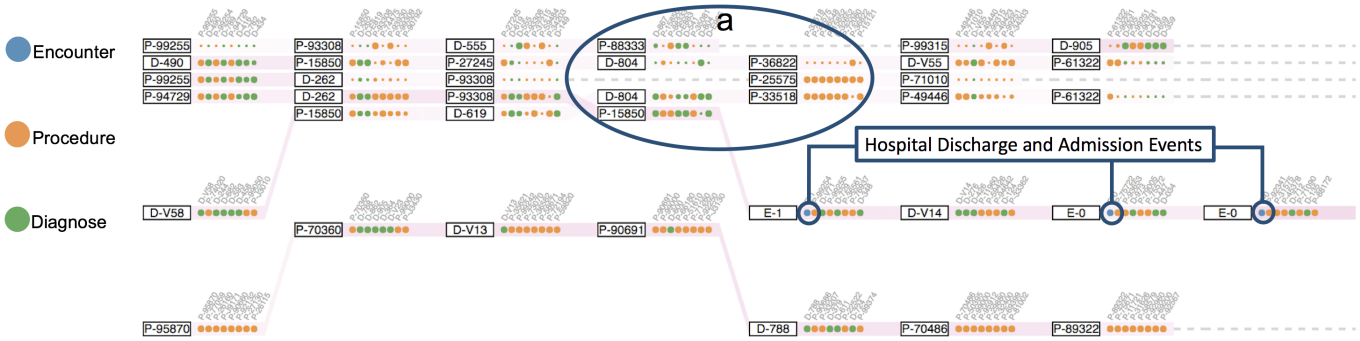

Fig. 8. The thread view of EventThread being used to explore medical event data for a cohort of patients suffering from chronic obstructive pulmonary disease (COPD). There are 6 threads generated, which are divided into 8 stages. A branch with several encounter events (hospital admissions and discharges) takes one thread away from the first latent stage category, indicating a group of patients that develop a high risk of increased hospital admissions later in the disease progression.

included seven years of longitudinal medical data, captured between 2008 and 2014. Each patient's individual timeline, however, varied in length and time span within those seven years.[1]

Visualization and results. We applied EventThread to the medical data described above, after first aligning all patients' event data to a common $t = 0$ time period determined by the patient's first hospitalization. We then processed the data to create 6 threads based on annual stages,with each thread represents a typical clinical pathway of a certain cohort. Even without any clustering across threads, the data exhibited a few interesting trends. First, the number of event sequences for each thread reduced in volume over time, reflecting patients either leaving the health care system or succumbing to their conditions. In addition, it was clear that some threads captured relatively rare progressions while others represented the majority of the cohort.

Adjusting the clustering to group similar threads, we explored the order in which the threads merged with each other at different stages. Eventually, settling on a clustering level of 0.16, we discovered a single main cluster containing 5 threads with a single outlier thread capturing more unusual progressions. Most interestingly, the main cluster was grouped together only for stage two through four, before a branch appeared taking one cluster away from the group at stage 5 and continuing (Fig. 8(a)).

The stage 5 branch contained several encounter events (hospital admit and discharge), suggesting that this was a group at high risk of increased hospital admissions. As described in our Expert Interview, this finding was among the most clinically interesting found in our analysis of the medical data, and has potential for real-world impact.

6.1.2 Usage Scenario II: Analysis of Vehicle Maintenance

In our second usage scenario, we applied EventThread to a corpus of vehicle maintenance records. This dataset contained nearly 5,000 maintenance records from 1,112 cars. Each record consisted of a car's id, a maintenance type, a specific maintenance item, and a description of the item. There was a total of six maintenance types which can be found in the legend of Fig. 1.

Visualization and results. Different from COPD data, we aligned the sequences of vehicle dataset by the number of maintenance that was performed, considering that different people may have diverse usage habits. After data processing, we generated seven latent threads representing various car owner's maintenance habits and each stage representing another round of maintenance on a car.

We explored the latent stage categories, and finally settled on a clustering level of 0.2, where the initial stage was generally divided into three latent categories. This included one major category with five threads characterized by *beautification* events, and two smaller threads exhibiting large numbers of *repair* and *parts replacement* events. As illustrated in Fig. 1, the data revealed several interesting patterns. First, all threads tended to have more *repairs* and *parts replacements* in later stages. Second, a branch that split from the second category in stage 2 contained several *diagnostics* events (e.g., performing diagnostic tests). The vehicles in this branch were generally in better condition, with lower levels of *repair* and *parts replacement* events in later stages compared to other threads. This shows, perhaps, the benefit of increased diagnostic testing.

Furthermore, we found a very large difference between thread 1 and 7. These two groups of cars used different types of engine oil. In thread 1 *semi synthetic oil* was a regularly occurring pattern, while for thread 7 it was *full synthetic oil* that was found with high probability. From Fig. 1, we can see thread 7 merged with the third category, a group of cars using *premium synthetic oil*, in stage 6, and threads in this latent category gradually integrated to one pattern (as shown in Fig. 1(b)). To thoroughly observe the different conditions between cars with distinct engine oils in later stages, we expanded the event features of the last stage, and an obvious contrast is revealed. Cars using *full synthetic oil* and *premium synthetic oil* (as shown in Fig. 1(b)) showed much less risk of car repair and parts replacements compared to cars using *semi-synthetic oil* (as shown in Fig. 1(a)). This finding indicates that *full synthetic oil* and *premium synthetic oil* are a better choice when selecting engine oils.

6.1.3 Usage Scenario III: Analysis of Academic Behaviors

For our third usage scenario, we analyzed a dataset with career progression events for group of professors. The dataset contained sequences of timestamped milestone events in 10 different categories such as earning new degrees, being assigned new job titles, or publishing (e.g., books, journal articles, and conference papers). The dataset included records of 40 individuals spanning 23 years. We manually classified the events into 3 high-level types including training, publishing, and promotion.

Visualization and results. We aligned all sequences to the first occurring event, then applied EventThread to create 4 threads based on annual stages. Each thread stands for a career path of a group of scholars. After adjusting the clustering level, interesting patterns were revealed. We captured the result from stage 8 to stage 15 in Fig. 9. As shown in the figure, stage 8 was generally categorized into three groups, indicating people with different titles: Associate Professors were representative of thread 1, Professors were represented in thread 2, and Assistant Professor were most represented in threads 3 and 4. We found that in thread 2, full professors were more likely to have the co-occurring event of publishing conference papers. The assistant professor cluster, however, contained a group publishing newspaper articles and conference papers (thread 3), and another publishing in journals (thread 4). The difference between these two threads became more significant at stage 12, as evidenced by the split which occurs in Fig. 9(b). As for thread 1, all publishing events occurred with high

[1]The discretization process that divides event data into stages is done by calendar year in our prototype. Because event dates in the medical dataset are perturbed by a random offset for each patient to preserve privacy, the time window may not align with the change in calendar year. This produces a time range of 8 calendar years, as evidenced by the 8 stages in Fig. 8.

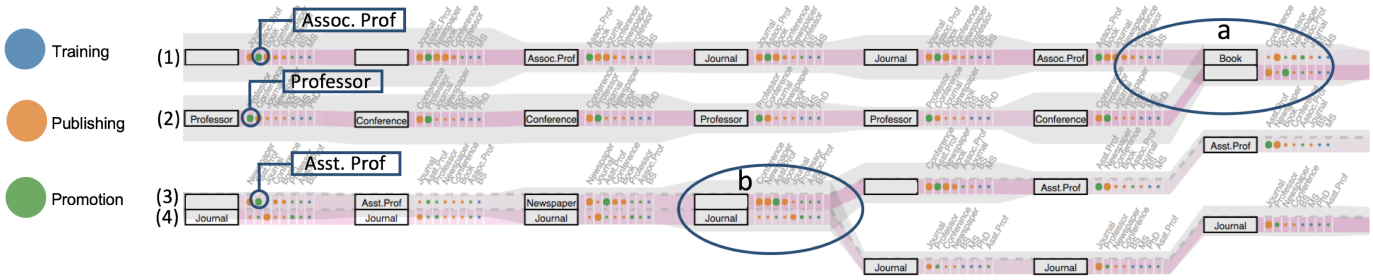

Fig. 9. An analysis result of users' academic behaviors from stage 8 to stage 15. Threads are generally grouped by users with different academic titles. The behavior patterns in each thread are significantly different. Similarities between threads are illustrated through branch merges and splits.

probabilities in various stages, and the thread merged with thread 2 at stage stage 15 (as shown in Fig. 9(a). This finding indicates an increasing similarity between professors' and associate professors' behaviors as those associate professors gain seniority.

6.2 Expert Interview

To gather qualitative feedback about the system, we interviewed a medical expert with domain knowledge of the COPD usage scenario outlined above. We interviewed an Assistant Professor of Medicine at the University of North Carolina School of Medicine. In addition to her clinical training, the Medical Doctor also has a Masters in Healthcare Administration giving her strong perspectives on a variety of medical use cases ranging from population-based policies to point-of-care treatment decisions. She also has experience with health informatics, and is generally receptive to the potential of technology to address healthcare challenges.

We interviewed the doctor for approximately 35 minutes. The interview began with a brief 5 minute introduction of the EventThread, including its visual and interaction design. The remaining 30 minutes were spent visually exploring data and discussing the implications of system's analytical features.

The short introduction was sufficient for the doctor to understand the main idea behind the system's design, including the most significant visual encoding used to represent the data. She asked periodic questions throughout the interview session to clarify more detailed aspects of the visual design, and was able to quickly apply the moderator's answer to derive new findings from the visualization. Interaction with the system was primarily performed by the moderator in response to the doctor's verbal queries. This shows that the short introduction was sufficient to bootstrap the doctor's analysis, but that a more comprehensive tutorial would be required for a user to be independently capable of using the system to its full potential.

The most common refrain from the doctor was "Oh, interesting!" Many times this was followed by "I wouldn't have thought of that" or something similar. For example, she was most interested in the finding outlined in our medical use case: the group of COPD patients which in later stages broke away from the main group and experienced much higher rates of hospitalization. "The's the only path that has some blue" she pointed out quickly, latching on to the color coding distinguishing between types of events.

Moreover, the doctor felt that the finding was exactly the sort of information that health care systems would be interesting in discovering. "Everyone's trying to build predictive models for risk reduction and value-based care," she began. "I could maybe see something like this being used to cluster high-risk patients... we don't really know a lot about how they cluster out" and "if we knew, that might be helpful." Asking to have more time than our interview allowed, she stated, "It would be nice to look around [more]."

The doctor also spent a significant amount of time asking for clarification about exactly what events we were using. She stated that this was due, in part, to inconsistencies in coding between doctors and the desire to know what doctors meant when using a specific code. This supports the approach taken in EventThread in which similar event sequences are grouped together rather than requiring an exact sequence match as often done in the past.

The doctor also talked about the benefits of the companion views. "The individual view is interesting" because it provides "a look-back at what happened to the individual patients" who are represented by the main clustered overview. She also stated that "people would be interested in [the treemap view] as well" because it gives a "quick, easy way to see" what events are common.

Finally, the doctor also identified some aspects of the design that were confusing. One especially challenging point of feedback was that the reliance on codes (of which there are tens of thousands) is difficult for interpretation. Clinicians do not have the codes memorized. Textual descriptors are much more meaningful. However, these descriptions can be quite long and are not directly suitable for use as labels within a dense visualization. The doctor felt that our hover-activated tooltips were very useful, but wondered about better solutions. This is a challenge to consider in future work. A second challenge identified by the doctor is the presence of "uninteresting" medical events within the visualization. There were times that she ignored events that she felt were clinically irrelevant. The system currently allows users to turn on or off categories of events. However, the interview showed that more fine grained control would be useful.

7 Conclusion and Discussions

We have presented EventThread, a technique designed to support visual summarization and latent stage analysis of large scale and high-dimensional event sequence data. Based on event sequence clustering of high-level structures via tensor decomposition, EventThread incorporates a robust layout algorithm to promote latent thread comparisons as well as dynamic exploration of latent stage categories. We proposed a novel visualization design of threads and multiple coordinated views with rich interactions to comprehensively assist users with data exploration and analysis. We evaluated our system via real-world event sequence datasets, and conducted an interview with an expert from the health-care domain. These results demonstrate that our design can be used to identify semantically interesting patterns in highly summarized event sequence data and facilitate latent stage analysis. However, there are several key limitations: (1) stages are created with a static, constant duration which can be improved to fit many real-world scenarios in which stages can differ in length as entities may have different progression rates [36]. Second, the number of desired latent sequential patterns is currently manually defined. We intend to explore adaptive tensor decomposition algorithms to help automatically identify optimal values for this parameter for a given dataset. Moreover, we also plan to conduct formal experimental user studies to gain more valuable insights regarding the usability of the system.

Acknowledgments

We would like to thank all the reviewers for their constructive comments. We also would like to thank all the users and domain experts who participated our user study. This work is a part of the research supported from NSFC Grants 61602306, STCSM 15JC1401700, the National Grants for the Thousand Young Talents in China, and the NSFC-Zhejiang Joint Fund for the Integration of Industrialization and Information under Grant No. U1609220.

REFERENCES

[1] W. Aigner, S. Miksch, H. Schumann, and C. Tominski. *Visualization of time-oriented data*. Springer Science & Business Media, 2011.

[2] W. Aigner, S. Miksch, B. Thurnher, and S. Biffl. Planninglines: novel glyphs for representing temporal uncertainties and their evaluation. In *Proceedings of International Conference on Information Visualisation*, pp. 457–463. IEEE, 2005.

[3] J. Alon, S. Sclaroff, G. Kollios, and V. Pavlovic. Discovering clusters in motion time-series data. In *Proceedings of IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, vol. 1, pp. I–I. IEEE, 2003.

[4] I. Cadez, D. Heckerman, C. Meek, P. Smyth, and S. White. Model-based clustering and visualization of navigation patterns on a web site. *Data Mining and Knowledge Discovery*, 7(4):399–424, 2003.

[5] N. Cao, D. Gotz, J. Sun, and H. Qu. Dicon: Interactive visual analysis of multidimensional clusters. *IEEE Transactions on Visualization and Computer Graphics*, 17(12):2581–2590, 2011.

[6] G. Chowdhury. *Introduction to Modern Information Retrieval*. Facet publishing, 2010.

[7] M. J. Cohen, A. D. Grossman, D. Morabito, M. M. Knudson, A. J. Butte, and G. T. Manley. Identification of complex metabolic states in critically injured patients using bioinformatic cluster analysis. *Critical Care*, 14(1):R10, 2010.

[8] S. B. Cousins and M. G. Kahn. The visual display of temporal information. *Artificial Intelligence in Medicine*, 3(6):341–357, 1991.

[9] B. C. Dickerson, A. Bakkour, D. H. Salat, E. Feczko, J. Pacheco, D. N. Greve, F. Grodstein, C. I. Wright, D. Blacker, H. D. Rosas, et al. The cortical signature of alzheimer's disease: regionally specific cortical thinning relates to symptom severity in very mild to mild ad dementia and is detectable in asymptomatic amyloid-positive individuals. *Cerebral Cortex*, 19(3):497–510, 2009.

[10] H. M. Fonteijn, M. Modat, M. J. Clarkson, J. Barnes, M. Lehmann, N. Z. Hobbs, R. I. Scahill, S. J. Tabrizi, S. Ourselin, N. C. Fox, et al. An event-based model for disease progression and its application in familial alzheimer's disease and huntington's disease. *NeuroImage*, 60(3):1880–1889, 2012.

[11] D. Gotz. Soft patterns: moving beyond explicit sequential patterns during visual analysis of longitudinal event datasets. In *Proceedings of IEEE VIS Workshop on Temporal and Sequential Event Analysis*, 2016.

[12] D. Gotz and H. Stavropoulos. Decisionflow: visual analytics for high-dimensional temporal event sequence data. *IEEE transactions on Visualization and Computer Graphics*, 20(12):1783–1792, 2014.

[13] R. A. Harshman. Foundations of the parafac procedure: Models and conditions for an" explanatory" multi-modal factor analysis. 1970.

[14] Z. Huang, X. Lu, H. Duan, and W. Fan. Summarizing clinical pathways from event logs. *Journal of Biomedical Informatics*, 46(1):111–127, 2013.

[15] C. R. Jack, D. S. Knopman, W. J. Jagust, L. M. Shaw, P. S. Aisen, M. W. Weiner, R. C. Petersen, and J. Q. Trojanowski. Hypothetical model of dynamic biomarkers of the alzheimer's pathological cascade. *The Lancet Neurology*, 9(1):119–128, 2010.

[16] C. H. Jackson, L. D. Sharples, S. G. Thompson, S. W. Duffy, and E. Couto. Multistate markov models for disease progression with classification error. *Journal of the Royal Statistical Society: Series D (The Statistician)*, 52(2):193–209, 2003.

[17] G. M. Karam. Visualization using timelines. In *Proceedings of ACM SIGSOFT International Symposium on Software Testing and Analysis*, pp. 125–137. ACM, 1994.

[18] J. Kiernan and E. Terzi. Constructing comprehensive summaries of large event sequences. *ACM Transactions on Knowledge Discovery from Data*, 3(4):21, 2009.

[19] J. Kiernan and E. Terzi. Eventsummarizer: a tool for summarizing large event sequences. In *Proceedings of International Conference on Extending Database Technology: Advances in Database Technology*, pp. 1136–1139. ACM, 2009.

[20] Y. Koren and L. Carmel. Visualization of labeled data using linear transformations. In *InfoVis*, 2003.

[21] M. Krstajic, E. Bertini, and D. Keim. Cloudlines: compact display of event episodes in multiple time-series. *IEEE Transactions on Visualization and Computer Graphics*, 17(12):2432–2439, 2011.

[22] C. Li and G. Biswas. A bayesian approach to temporal data clustering using hidden markov models. In *ICML*, pp. 543–550, 2000.

[23] M. Monroe, R. Lan, H. Lee, C. Plaisant, and B. Shneiderman. Temporal event sequence simplification. *IEEE Transactions on Visualization and Computer Graphics*, 19(12):2227–2236, 2013.

[24] T. Mori, A. Takada, Y. Iwamura, and T. Sato. Automatic human life summarization system in sensory living space. In *IEEE International Conference on Systems, Man and Cybernetics*, vol. 2, pp. 1583–1588. IEEE, 2004.

[25] T. Oates, L. Firoiu, and P. R. Cohen. Using dynamic time warping to bootstrap hmm-based clustering of time series. In *Sequence Learning*, pp. 35–52. Springer, 2000.

[26] N. Osato, M. Itoh, H. Konno, S. Kondo, K. Shibata, P. Carninci, T. Shiraki, A. Shinagawa, T. Arakawa, S. Kikuchi, et al. A computer-based method of selecting clones for a full-length cdna project: simultaneous collection of negligibly redundant and variant cdnas. *Genome Research*, 12(7):1127–1134, 2002.

[27] A. Perer and J. Sun. Matrixflow: temporal network visual analytics to track symptom evolution during disease progression. In *AMIA Annual Symposium Proceedings*, vol. 2012, p. 716. American Medical Informatics Association, 2012.

[28] Q.-K. Pham, G. Raschia, N. Mouaddib, R. Saint-Paul, and B. Benatallah. Time sequence summarization to scale up applications. In *Proceedings of ACM Conference on Information and Knowledge Management*, pp. 1137–1146. ACM, 2009.

[29] C. Plaisant, B. Milash, A. Rose, S. Widoff, and B. Shneiderman. Lifelines: visualizing personal histories. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, pp. 221–227. ACM, 1996.

[30] R. I. Scahill, J. M. Schott, J. M. Stevens, M. N. Rossor, and N. C. Fox. Mapping the evolution of regional atrophy in alzheimer's disease: unbiased analysis of fluid-registered serial mri. *Proceedings of the National Academy of Sciences*, 99(7):4703–4707, 2002.

[31] P. Smyth et al. Clustering sequences with hidden markov models. *Advances in Neural Information Processing Systems*, pp. 648–654, 1997.

[32] P. M. Thompson, K. M. Hayashi, G. De Zubicaray, A. L. Janke, S. E. Rose, J. Semple, D. Herman, M. S. Hong, S. S. Dittmer, D. M. Doddrell, et al. Dynamics of gray matter loss in alzheimer's disease. *Journal of Neuroscience*, 23(3):994–1005, 2003.

[33] P. M. Thompson, M. S. Mega, R. P. Woods, C. I. Zoumalan, C. J. Lindshield, R. E. Blanton, J. Moussai, C. J. Holmes, J. L. Cummings, and A. W. Toga. Cortical change in alzheimer's disease detected with a disease-specific population-based brain atlas. *Cerebral Cortex*, 11(1):1–16, 2001.

[34] K. Wongsuphasawat and D. Gotz. Outflow: visualizing patient flow by symptoms and outcome. In *IEEE VisWeek Workshop on Visual Analytics in Healthcare, Providence, Rhode Island, USA*, pp. 25–28. American Medical Informatics Association, 2011.

[35] K. Wongsuphasawat, J. A. Guerra Gómez, C. Plaisant, T. D. Wang, M. Taieb-Maimon, and B. Shneiderman. Lifeflow: visualizing an overview of event sequences. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, pp. 1747–1756. ACM, 2011.

[36] J. Yang, J. McAuley, J. Leskovec, P. LePendu, and N. Shah. Finding progression stages in time-evolving event sequences. In *Proceedings of International Conference on World Wide Web*, pp. 783–794. ACM, 2014.

[37] J. Zhao, M. Glueck, F. Chevalier, Y. Wu, and A. Khan. Egocentric analysis of dynamic networks with egolines. In *Proceedings of CHI Conference on Human Factors in Computing Systems*, pp. 5003–5014. ACM, 2016.

[38] J. Zhao, Z. Liu, M. Dontcheva, A. Hertzmann, and A. Wilson. Matrixwave: visual comparison of event sequence data. In *Proceedings of Annual ACM Conference on Human Factors in Computing Systems*, pp. 259–268. ACM, 2015.

[39] J. Zhou, J. Liu, V. A. Narayan, J. Ye, A. D. N. Initiative, et al. Modeling disease progression via multi-task learning. *NeuroImage*, 78:233–248, 2013.