Visual Progression Analysis of Event Sequence Data

Shunan Guo, Zhuochen Jin, David Gotz, Fan Du, Hongyuan Zha, and Nan Cao

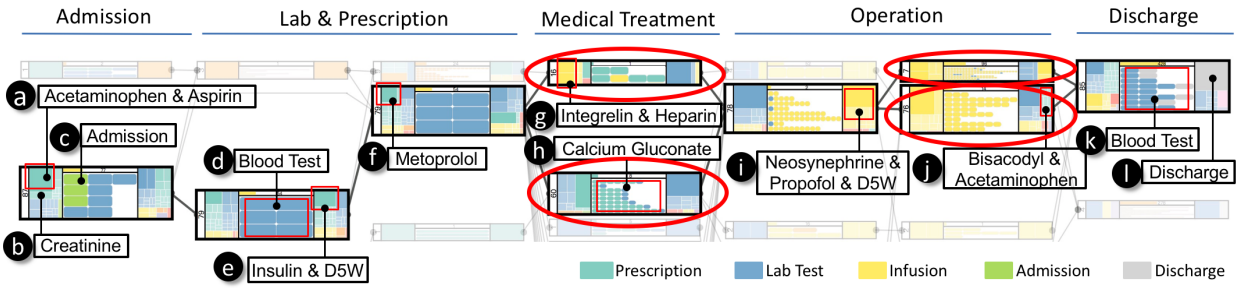

Fig. 1. The clinical progression of patients with cardiac diseases in MIMIC dataset. The system identified seven stages and the medical experts further grouped them into five phases according to their interpretation.

Abstract— Event sequence data is common to a broad range of application domains, from security to health care to scholarly communication. This form of data captures information about the progression of events for an individual entity (e.g., a computer network device; a patient; an author) in the form of a series of time-stamped observations. Moreover, each event is associated with an event type (e.g., a computer login attempt, or a hospital discharge). Analyses of event sequence data have been shown to help reveal important temporal patterns, such as clinical paths resulting in improved outcomes, or an understanding of common career trajectories for scholars. Moreover, recent research has demonstrated a variety of techniques designed to overcome methodological challenges such as large volumes of data and high dimensionality. However, the effective identification and analysis of latent stages of progression, which can allow for variation within different but similarly evolving event sequences, remain a significant challenge with important real-world motivations. In this paper, we propose an unsupervised stage analysis algorithm to identify semantically meaningful progression stages as well as the critical events which help define those stages. The algorithm follows three key steps: (1) event representation estimation, (2) event sequence warping and alignment, and (3) sequence segmentation. We also present a novel visualization system, ET^2, which interactively illustrates the results of the stage analysis algorithm to help reveal evolution patterns across stages. Finally, we report three forms of evaluation for ET^2: (1) case studies with two real-world datasets, (2) interviews with domain expert users, and (3) a performance evaluation on the progression analysis algorithm and the visualization design.

Index Terms—Progression Analysis, Visual Analysis, Event Sequence Data

1 Introduction

Across a broad range of application areas, data is frequently collected in the form of temporal event sequences. Such data typically includes large numbers of discrete timestamped events, which are then grouped by a common attribute and ordered to form a collection of event sequences. These sequences are often collected in large volumes and capture patterns of progression, showing the evolution of a focal entity over different stages. For example, in healthcare, electronic health records (EHRs) capture timestamped observations (e.g., diagnoses, medications, procedures) for each patient. An analysis of progression patterns in this domain might aim to uncover the evolution of a disease from mild to severe within a specific patient population.

Recognizing the importance and broad applicability of event sequence analysis, a wide range of visual analysis techniques have been developed in recent years. These advances have addressed a wide variety of algorithmic challenges related to the discovery and analysis of patterns within complex event sequence datasets. This includes scalability to large volumes of sequences (e.g., [11, 45]), the ability to handle high dimensionality (large numbers of event types, e.g., [15]), techniques for events with attributes [7], and the tight integration with a variety of pattern mining algorithms for guided exploration (e.g., [37]).

However, it remains difficult to use these techniques to understand progression patterns within complex event sequence data. One major challenge is to find similar high-level structures that appear across the collection of underlying event sequences. Such structures, which represent latent stages through which entities may evolve over the course of an event sequence, can be highly informative in terms of understanding patterns of progression over time.

Determining these latent stages and the patterns through which event sequences move through the stages, requires the ability to detect evolution patterns which cannot be easily defined using the rigid event ordering rules. For instance, in a medical use case, a patient may visit a lab for blood tests just before seeing a doctor, just after, or perhaps even a day or two later. Semantically, the exact order is not critical. The important feature in this case is that the events in the patient's medical record reflect a particular stage (e.g., outpatient monitoring of symptoms) in a progression that may lead to different types of events at a later time (e.g., surgical procedures in a hospital). Unfortunately, the body of existing work has focused nearly exclusively on the analysis of *hard patterns* [14]: exactly matching orderings of events in which the sequence of event types defining a pattern match an explicitly defined order. This makes them poor match tasks where detailed differences in event order as less important than understanding higher-level progression patterns across stages.

To meet this need, recent work has explored more flexible ap-

- *Shunan Guo and Hongyuan Zha are with East China Normal University. E-mail: g.shunan@gmail.com, zha@sei.ecnu.edu.cn*
- *Zhuochen Jin is with iDVX lab at Tongji University.E-mail: zhuochen.jin@tongji.edu.cn*
- *David Gotz is with University of North Carolina at Chapel Hill. Email: gotz@unc.edu*
- *Fan Du is with University of Maryland. Email: fan@cs.umd.edu*
- *Nan Cao is with iDVX lab at Tongji University and is the corresponding author. Email: nan.cao@tongji.edu.cn*

Manuscript received xx xxx. 201x; accepted xx xxx. 201x. Date of Publication xx xxx. 201x; date of current version xx xxx. 201x. For information on obtaining reprints of this article, please send e-mail to: reprints@ieee.org. Digital Object Identifier: xx.xxxx/TVCG.201x.xxxxxxx

proaches. Closest to our work is EventThread [18], which attempted to tackle the stage identification problem by algorithmically segmenting groups of event sequences into fixed-width time intervals, then grouping similar event sequence segments into clusters (i.e., soft patterns) within each interval. Each cluster can then be summarized statistically to characterize a series of latent stages of event sequence progression. The EventThread approach was shown to be quite useful in identifying stage progression patterns within evolving event sequences. However, it has a key limitation: the reliance upon fixed-width time intervals. For example, EventThread might split medical data into yearly or monthly segments before stages are identified. However, in medical data, different patients will have differing speeds of disease progression. For example, one patient may progress from mild to severe heart disease over multiple years, while another may progress over a period of weeks. It is not possible for EventThread to identify the occurrence of similar stages (e.g., mild heart disease vs. severe heart disease) that occur across differing time scales as in this example.

This paper introduces EventThread 2, or ET^2, an entirely new visual progression analysis technique and system designed to overcome the time scale limitation in previous work. More specifically, the primary contributions of this paper are as follows:

- **Stage Analysis Algorithm.** We introduce a novel unsupervised algorithm for aligning and segmenting collections of temporal event sequences to identify latent stages of progression. It extracts meaningful latent stages of heterogeneous duration to better reflect the non-linear progression patterns exhibited in many real-world applications.
- **System for Visual Query and Interrogation.** We introduce a dynamic event sequence visual analysis system which combines temporal queries with a thread-based design to show the event sequence evolution patterns across the mined stages at three different granularities: low-level sequences of individual raw events, aggregate sequences showing frequent and rare stage transitions, and a high-level summarization highlighting the more significant stage transition patterns. Interactions are provided for users to explore these views and navigate between them.
- **Evaluation.** We demonstrate the utility of the proposed approach in three different ways. First, we review the application of ET^2 to a variety of real-world event sequence datasets. Second, we report feedback from an interview with two expert users from medical domain. Third, we demonstrate the performance of our algorithm through a scalability experiment and the usability of our visualization design through an extra performance interview.

2 Related Work

Event sequence analysis has emerged as an important topic within a wide range of application areas (e.g., electronic transactions [46], clickstreams [49], medical [32]). Given this broad applicability, a variety of visualization and computational methods for the analysis of event sequence data have been developed in recent years. These efforts have aimed to address the problem from a range of perspectives and analytic goals, including statistical description, event summarization, classification, pattern discovery, decision making, and behavior prediction (e.g., [10, 17, 26, 47]). This section provides an overview of prior work in a subset of these topics, focusing on the areas that are most relevant to the new methods outlined in this paper.

2.1 Sequential Event Pattern Analysis

Mining methods designed to discover sequential event patterns have been widely studied and often focus on the development of efficient algorithms for determining the longest and most frequent patterns in a fully-automated fashion (e.g., [4]). One challenge is that important patterns can be diffused across large numbers of permutations of similar but distinct patterns. To address the issue, some methods extend frequent pattern matching to dealing with longer sequences by supporting the identification of key events [8, 12, 13]. Other methods apply supervised learning (e.g., [33]) which hierarchically group events into a higher level of abstraction based on predefined event taxonomies. Unsupervised cluster-based methods have also been explored [2,5,24,25,30], which aggregate similar event sequences to determine the frequency of patterns within these latent groups.

The above computational methods provide fully automated solutions, and are highly useful in identifying large numbers of relatively short recurring event patterns. However, communicating the variety of patterns identified by the algorithms (and their relative frequencies) to human analysts is essential for interpretation and can be difficult to do effectively [20, 29, 31], which motivates many visual analysis based approaches discussed below.

2.2 Visual Summarization of Event Sequences

Recently, a wide range of visual analysis techniques for event sequence data have been developed. For instance, Lifelines [34] focused on summarizing individual patient records, while a later version Lifeline2 [40] improved the scalability for their approach by developing a tree-based aggregate representation to display the permutations of event occurrences and their proportions for groups of patients records. Outflow [44] similarly captured all permutations of event occurrence based on a graph-based model. EventAction [9] summarized event data occurring within a common time period, and explored potential outcomes for behavior prediction and recommendation.

These techniques help communicate various forms of sequential patterns, but rely on permutations in the order of event type occurrence to build the visual representations (i.e., "hard patterns" [14]). This limits their usage in many real-world applications where a large number of event types will lead to a combinatorial explosion in permutations of event order. In comparison, this paper introduces novel stage-based analysis and visualization methods aimed at summarizing event sequence progression through a set of latent stages over time. These stages are then used to summarize aggregate progression over time, using higher-level representations (i.e., the stages) which gather into the same stage similar but distinct event subsequences.

2.3 Stage Analysis

Stage analysis in event sequence data help reveal the progression of the event development. For example, when dealing with the EHR data, many traditional disease progression models such as [21,36,38,39] are developed based on clinician's experiences and their subjective assessments, but most of them produce imprecise results that can hardly used in real scenarios. Recent work has leverage advanced machine learning techniques, thus producing more precise results. For example, Wang et al. [41] introduced an unsupervised disease progression model, which is composed of a three-layer network and based on continuous-time Markov models and the standard EM algorithm. Yang et al. [48] developed a novel statistical model to classify event sequences and infer progression stages within each sequence category. These approaches perform well in identifying latent stages within a single event sequence, but are not capable of aligning and segmenting event sequence collections. In addition, the results of these methods are difficult to interpret. In contrast, our proposed algorithm can automatically segment and align a collection of event sequences with similar progression stages, and generate different levels of summarization to support interpretation.

2.4 Visual Analysis of Progression

Paralleling the recent interest in stage analysis and representations of progression that exist at a higher level than event permutations, visualization techniques have also been proposed to support interactive progression analysis. This includes work such as DecisionFlow [15] and related projects [16, 17] which aggregate event sequences into stages via manually identified milestones.

Rather than rely on manual milestone selection, the visual analytics method proposed in this paper includes a novel stage analysis algorithm to determine stages more systematically. In this sense, the most closely related work is EventThread [18], a recently published visual analytics system which incorporates automatic progression analysis methods. However, in EventThread, the length of a stage is predefined using a fixed size. This is a critical limitation which poorly reflects real-world scenarios as differences in time scale across stages can be critical, which is a major motivation in the work proposed in this paper.

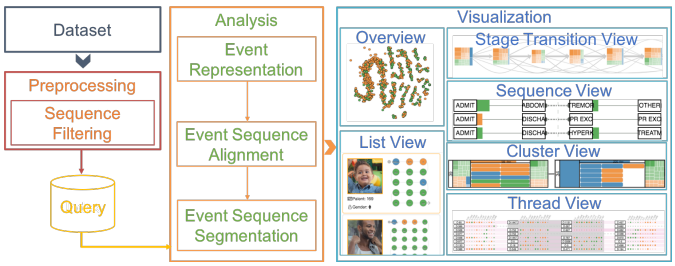

Fig. 2. The ET^2 system contains three key modules: a preprocessing module, an analysis module, and a visualization module. Together, these modules support a comprehensive and interactive analysis of the latent progression stages within a collection of event sequences

3 System Overview

Our system is designed to support interactive progression analysis of event sequence data. The development of ET^2 began with the identification of a set of key design requirements based on (1) the authors' own first-hand experiences working with users analyzing this form of data, and (2) a review of both the features and limitations of existing tools such as EventThread [18]. These design requirements are summarized as follows:

- **R1 Query capabilities to focus the analysis of large-scale event sequence datasets.** Large collections of real-world event sequence data typically contain very diverse sequences, often with little overlap in observed events. This heterogeneity can make it difficult, and less useful, to view all sequences at once. Instead, users often wish to apply initial queries against the full collection to focus an analysis on specific sets of sequences. Such a query capability allows the user to restrict an analysis to sequences that meet a set of inclusion/exclusion criteria. For example, in a medical analysis the user may query to see all sequences related to a specific diagnosis or treatment rather than "all patients."
- **R2 Stage-based progression summaries to facilitate interpretation and reasoning.** Complex event sequence datasets can contain vast numbers of event types, meaning that even very similar sequences will vary significantly in both the events observed and the order of occurrence when viewed at the level of individual events. To facilitate high-level understanding, an analysis system should be able to organize groups of semantically related events into latent stages which capture the overall progression of a subset of sequences, such as stages in disease progression over time.
- **R3 Time-varying stages to reflect non-linear speed of progression.** Many real-world event sequence datasets capture information about progression patterns which evolve at different rates between and within sequences. For example, medical conditions may progress at different rates between patients, and at different points in the treatment of a single patient. Thus, the stage-based summary should reflect this time-varying requirement.
- **R4 Multi-granular summarization to reveal progression patterns at different levels.** The system should be able to provide both low-level information (i.e., patterns of raw event) and high-level information (i.e., patterns of stages over time) about how event sequences progress. In this way, the system will provide both low-level interpretations of the progression patterns, as well as a higher-level understanding of sequence progression.
- **R5 Interactive analysis environment to explore results from multiple perspectives.** The system should provide a unified interface to meet the above requirements. This experience should be intuitive and integrated, allowing analysts to link findings at different granularities and time scales, to compare findings, and to understand both individual stages and the transitions between them.

Based on the above requirements, we developed the architecture of the ET^2 system as shown in Fig 2. This design includes three major modules: (1) the preprocessing module, which transforms the raw sequence data into a database for subsequent query and analysis (**R1**); (2) the analysis module, which extracts progression patterns from the raw event sequence data (**R2,R3,R4**); and (3) the visualization module, which provides multiple coordinated views to support result interpretation and interactive stage analysis (**R5**).

4 Progression Analysis

This section formalizes the problem of progression analysis and introduces the analysis pipeline developed to address this problem.

4.1 Algorithm Overview

The goal of progression analysis is to automatically infer the latent stages underlying a set of event sequences. We develop an algorithmic approach that is designed based on three intuitive assumptions. (1) The analyzed event sequences should have a similar progression process. For example, the medical records of patients with the same type of disease. This assumption requires a pre-filtering (**R1**) of the data to retrieve similar sequences. (2) The sequences may vary in number of events, duration, and pattern of progression through different stages. For example, patients with the same medical conditions may respond differently to treatments and their symptoms may progress at different rates of speed. (3) The progression process is irreversible.

Based on these assumptions, we propose an unsupervised progression analysis algorithm to segment event sequences into stages through three major steps: (1) estimation of event representation, (2) event sequence alignment, and (3) sequence segmentation. The first step employs a neuron network model to convert each event into a vector representation that captures event co-occurrence probabilities. The second step performs time shifting and warping to align and aggregate event sequences. In the third step, the aggregated sequences are segmented into latent stages using an optimization-based algorithm. This process is illustrated in Fig. 3 and described in detail below.

4.2 Estimation of Event Representation

The process of segmenting an event sequence into stages is similar to the task of segmenting a sentence into thematically consistent sub-phrases in text mining. The key is to identify the expected likelihood of event pairs co-occurring in close proximity. In text mining, word embedding methods based on neural-network models (e.g., skip-gram [27]) have been proposed to address similar problem. These methods generates a latent vector representation for each word in a text corpus such that the distances between vectors indicate the co-occurrence likelihood between words. Following a similar idea, we develop an event embedding algorithm which converts each event into a vector derived from the context of surrounding events within a given group of sequences. Thus, the co-occurrence likelihood of the events is also captured by the distance between their vectors. Here, each event sequence is analogous to a sentence, in which each event is analogous to a word.

When applied to event sequence data, we use skip-gram to estimate the occurrence likelihood $P(e_j|e_i)$ of an event e_j given the occurrence of event e_i. This model is a three layer neural network (as shown in Fig. 3(1)). The input layer takes an event e_i as input with each neuron in the layer corresponding to a field of e_i's one-hot vector [42]. (2) A hidden layer compresses the sparse input one-hot vector into a dense latent vector with a much lower dimensionality. The embedding matrix that connects the input and hidden layer is learned during the training process. Each row of the matrix is a vector that captures the neighborhood context of an event in the dataset and is the vector that we aim to compute. (3) The output layer calculates $P(e_j|e_i)$ based on a softmax function [43].

We train the above model based on the entire dataset with the goal of maximizing $P(e_j|e_i)$ for each input event e_i (i.e., find the set of events that are most likely to co-occurred with e_i), which is equivalent to minimizing the following loss function:

$$J = -\sum_{m=1}^{M}\sum_{i=1}^{n}\sum_{j=1}^{n}\log(\omega_{ij}\cdot P(s_m^{(j)}|s_m^{(i)})), |t_i - t_j| \leq T \qquad (1)$$

where M indicates the number of event sequences in the training set; n is the length of an event sequence; $s_m^{(i)} = (e_i, t_i)$ indicates the i-th event e_i occurred at time t_i inside the m-th sequence s_m; $P(\cdot)$ is the

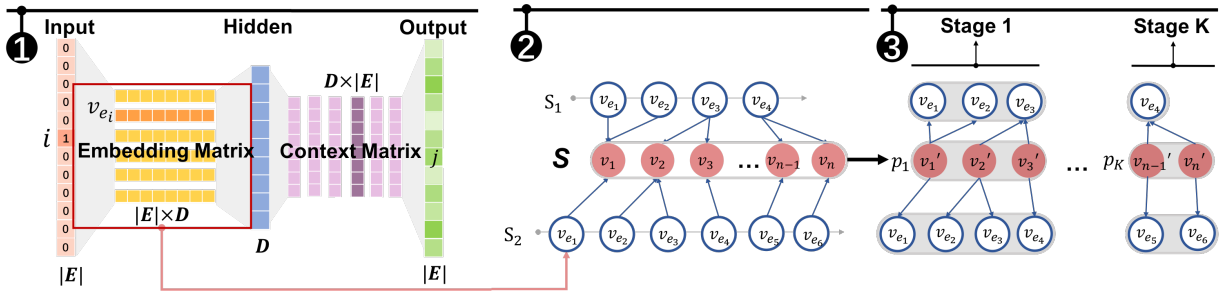

Fig. 3. Progression analysis includes three major steps: (1) event representation estimation, (2) sequence alignment, and (3) sequence segmentation.

aforementioned probability to be maximized. ω_{ij} is a weight which controls how the training samples are generated. In text mining, the training samples of the above model are given in the form of word pairs (the input word and a nearby word), which is generated by sampling within a fix-sized sliding window. However, unlike words in a sentence, event sequences capture both order and time intervals between events. For instance, two adjacent events occurring on the same day tend to have a stronger connection than adjacent events separated by a longer duration. Thus, instead of sampling training data using a fix-width window, we give different weights ω_{ij} for different training samples $\langle (e_i,t_i),(e_j,t_j)\rangle$ according to their time interval. This is defined as:

$$\omega\langle (e_i,t_i),(e_j,t_j)\rangle = \begin{cases} exp(-|t_i-t_j|/\theta) & |t_i-t_j| \le T, \\ 0 & \text{otherwise} \end{cases}$$

where (e_i,t_i) indicates the i-th event e_i and its timestamp t_i in a sequence, θ is the decay coefficient, and T is the maximum time span. This idea is inspired by [19], which summarized the temporal relation between events into a compact graph and generated synthetic samples for training. In our method, however, event pairs are sampled from real data and we directly multiply each sample by its corresponding weight in the loss function. In this case, event pairs with shorter time intervals will be assigned a higher weight to imply stronger correlation and thus they will have a greater impact on the final regression.

The computation of the event vectors costs $O(|E|^2)$, where $|E|$ is the number of total events. This step is calculated in an offline procedure which will not effect the performance of the online system.

4.3 Alignment

Event sequences captured in real-world settings, even when recording multiple instances of the same process, can vary dramatically in length and exact event orders. For example, multiple patients suffering from the same disease may have doctor visits at different times, they may have symptoms appear or recede at different speeds, and they may have distinct sets of co-morbidities. These differences will result in electronic health records of different length and with different events that nonetheless represent similar disease progressions.

In these cases, simply aligning sequences based on the first event or an arbitrary alignment [18,28] will not allow the aggregate progression patterns of interest to emerge. Instead, it is necessary to align event sequences based on their implicit semantics. To this end, we employ Dynamic Time Warping (DTW) [23] to align a group of event sequences with variable lengths and event orders (as shown in Fig. 3(2)).

In particular, our algorithm employs an iterative alignment process starts by constructing a mean-sequence $S = \langle (\varepsilon_1,t1),\ldots,(\varepsilon_n,t_n)\rangle$ initialized by the longest sequence in the dataset with the vector representation denoted as $V = (v_1,v_2,\ldots,v_n)$. In each iteration, an input event sequence is matched and merged to the mean-sequence via DTW, and the corresponding mean event vectors are also computed. Events in a sequence are treated as equidistant points sampled along the time axis, and the distance between a pair of events is determined by the Euclidean distance between the corresponding event vectors. DTW is thus applied based on these settings. In our implementation, FastDTW [35] is used for computational efficiency. The algorithm continues until all sequences have been merged. The resulting mean vector representation is noted as $\bar{V} = (\bar{v}_1,\bar{v}_2,\ldots,\bar{v}_n)$. The overall time complexity of this alignment step is $O(M(n-1))$, where M and n indicate the number of sequences and the length of the mean sequence, respectively.

4.4 Segmentation

In the final step, the mean-sequence S is split into segments (i.e., latent stages), which are then unpacked to derive the segmentation for each individual sequences (as shown in Fig. 3(3)). We borrows the idea of text segmentation and employs the Content Vector Segmentation (CVS) algorithm [1], which was originally introduced to divide document into coherent sections. Here, we treat the mean-sequence S as a non-fragment document, and each aggregate event as a word. Thus we are able to use algorithm to divide the sequence into coherent segments.

Intuitively, the design rationale of the algorithm is based on the assumption that events within the same segment (i.e., stage) should share similar semantics. Thus, a latent vector c_k is introduced to capture the semantic context of each segment p_k. Based on this assumption, an event e_i will be assigned into segment p_k if it has a high coherence with the context c_k which is given by $v_i \cdot c_k$, where v_i is the vector representation of e_i. Therefore, the goal of the algorithm is to learn an optimal c_k^* for each putative segment p_k which maximizes the overall coherence scores within each segment:

$$c_k^* = \arg\max \sum_i v_i^k \cdot c_k$$

where v_i^k is the vector representation of the i-th event in the k-th segment; c_k is the semantic context vector of the k-th segment with each field representing a latent topic [3]. For each segment p_k, the context vector c_k is calculated field by field as follows:

$$c_k(d) = \text{sign}(\sum_{i \in p_k} v_i^k(d))$$

where $c_k(d)$ is the d-th field in vector c_k; $v_i^k(d)$ refers to the value in d-th field of e_i's vector representation v_i^k.

A greedy algorithm has been used to approximately, but efficiently, solve the above optimization problem. In each iteration, it splits S or a subsequence of S into two parts at the place which maximizes the sum of c_k over all k segments. The algorithm stops automatically when the best available split results in a total score increase that is smaller than a threshold. We refer to the final number of segments after the greedy algorithm completes as K and the length of the mean sequence as n. The time complexity of the algorithm is $O(n^2K)$, correspondingly. Finally, we unpack S into individual sequences inheriting the stage information from S, and the segmentation result of each individual sequence is delivered to the visualization module for visual analysis. Note that if an event is aligned with multiple elements fell into different stages on the mean sequence, it will follow the element with highest similarity and get assigned to the corresponding stage.

5 Visualization

This section presents the ET^2 visualization and interaction design. It enumerates a set of design tasks that motivate the system's design choices, then describes the system's key views and user interactions.

5.1 Design Tasks

The visual and interaction design for the ET^2 system has been developed to support a set of motivating design tasks. The tasks were defined in response to our prior experience developing event sequence analysis techniques, lessons learned during ongoing collaborations with users who analyze temporal event sequence data, and the system requirements outlined in Section 3.

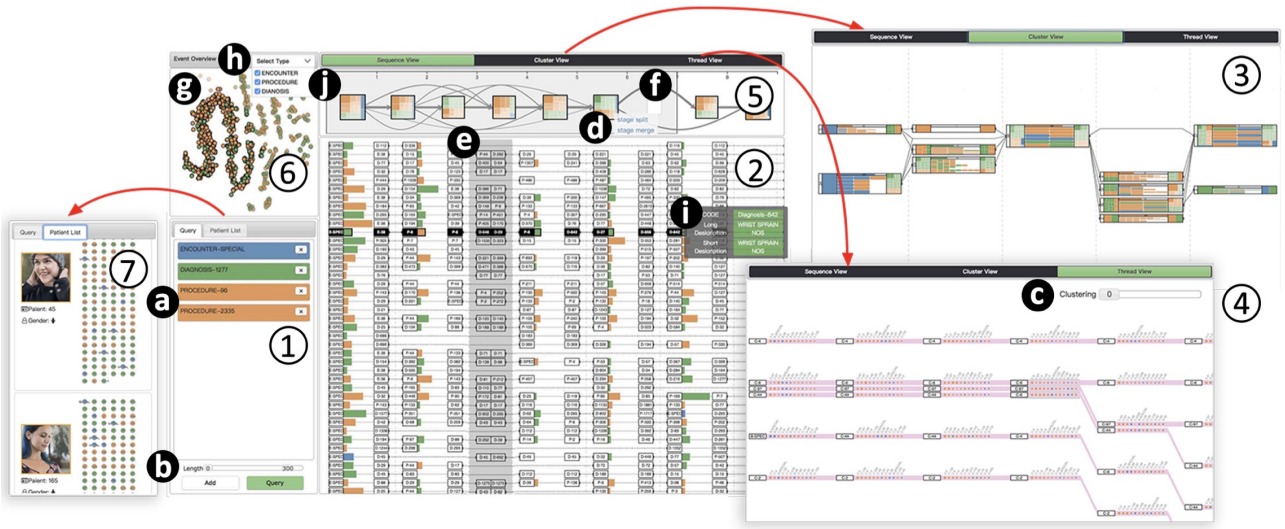

Fig. 4. The user interface of ET^2 consists of seven coordinated views: (1) query view, (2) sequence view, (3) cluster view, (4) thread view, (5) stage transition view, (6) event overview, and (7) entity list view.

T1 **Allow custom queries of the event sequence data.** Real-world event sequence datasets may have little in common. To help users focus an analysis by pre-filtering to a relevant set of event sequences, the visualization system should support queries against the sequence database with time-based constraints.

T2 **Display progression by stage.** To help users understand progression patterns, the visualization should clearly identify the boundaries between distinct stages and the order in which they occur. The visualization should also provide information about boundary events to help users understand what contributes to the beginning or end of a stage.

T3 **Provide detailed views of sequence segments aggregated within a given stage.** Due to the scale and diversity of the event subsequences that contribute to the definition of a stage, the visualization should be able to communicate both common and exceptional patterns and to allow comparison.

T4 **Indicate the key events which characterize a stage, and provide an overview of transition patterns between different stages.** To allow users to semantically interpret the meaning of each stage, it is important to provide overviews of the key events which contribute to a stage's definition. In addition, the visualization should communicate the overall state-to-stage progression patterns found within a dataset.

T5 **Facilitate visual data exploration and comparison.** Given the proposed data-driven approach to progression analysis, it is critically important to provide users with a way of exploring and comparing the extracted stages and evolution patterns across different levels of aggregation. The visualization should support this need through rich user interactions and coordinated views.

5.2 User Interface

Guided by the aforementioned design tasks, we developed the ET^2 visualization system as shown in Fig. 4. The user interface consists of seven key views, beginning with the *query view* (Fig. 4(1)) which employs a milestone-based query method that allows users to query the sequence database with time-based constraints **(T1)**.

Four coordinated views were designed to display stage analysis results at different levels of summarization once the query has been executed, including: a *sequence view* (Fig. 4(2)), a *cluster view* (Fig. 4(3)), a *thread view* (Fig. 4(4)) organized as tabs across the top of the interface, and a *stage transition view* (Fig. 4(5)). The *sequence view* displays the segmentation results for each individual sequence in a scrollable list, with additional details such as the duration, starting event, and end event for each segment **(T2)**. The *cluster view* aggregates segments within each stage into clusters and presents the evolution pattern between clusters across the different stages. An overview of the starting and ending events and the frequent subsequences is provided as the context of each cluster **(T3)**. The *stage transition view* further aggregates the clusters in the same stage, showing the event distribution and transition patterns across different stages **(T4)**. The *thread view* displays the latent clusters of similarly evolving sequences identified through tensor analysis [18] **(T3)** and provides the highest level of summarization with the maximal visual scalability. The *event overview* (Fig. 4(6)) provides a t-SNE projection of event vectors, which allows users to compare how events co-occur within the given stage **(T5)**. Finally, the *entity list view* (Fig. 4(7)) presents a detailed profile of individual entities as determined by the current selection. This provides users with easy access to the raw underlying events within a given sequence **(T5)**. Brushing and highlighting techniques are applied throughout the views to facilitate exploration across these coordinated views.

5.3 Sequence View

The *sequence view* aims to help users explore raw event sequences within the progression context **(T2)**. For example, a doctor may want to investigate the overall disease progression of a patient while inspecting the key diagnosis events and comparing the patients with similar progression patterns. To this end, the design should balance between the specific event details and the overall progression stages. To facilitate comparison, the design should also be able to align different sequences based on their progression process.

Following the above requirements, each sequence (Fig. 5(a)) is organized into columns aligned with K stages (Fig. 5(b)). This design choice facilitates the comparison of multiple sequences within the same progression stage. Adjacent stages are visually separated using vertical dashed-lines to highlight the stage boundaries. Each stage segment is depicted using a pair of labels (Fig. 5(c)) indicating the first and last events in the sequence segment. This design helps highlight key events which may signal a stage transition. The label pairs are connected by a solid light-grey line, which is overlaid with one of two distinct visual representations: a duration bar (Fig. 5(e)), or a detailed event timeline (Fig. 5(d)). The length of the duration bar, shown by default, encodes the duration of the segment within the sequence. The bar is color-coded with the event category that occurs most frequently within the segment to communicate a high-level summary. The detailed event timeline provides details-on-demand by encoding each individual event as a color-coded (by event category) rectangle whose x-axis position is determined by the event's relative time of occurrence. Users can switch between these two visual representations by clicking on any bar. When a sequence has no events within a particular stage, the segment is visualized as a dashed-line across the given stage (e.g., Fig. 5(f)). This design helps visually distinguish segments that were absent (e.g., a skipped disease stage for a patient) within a given sequence.

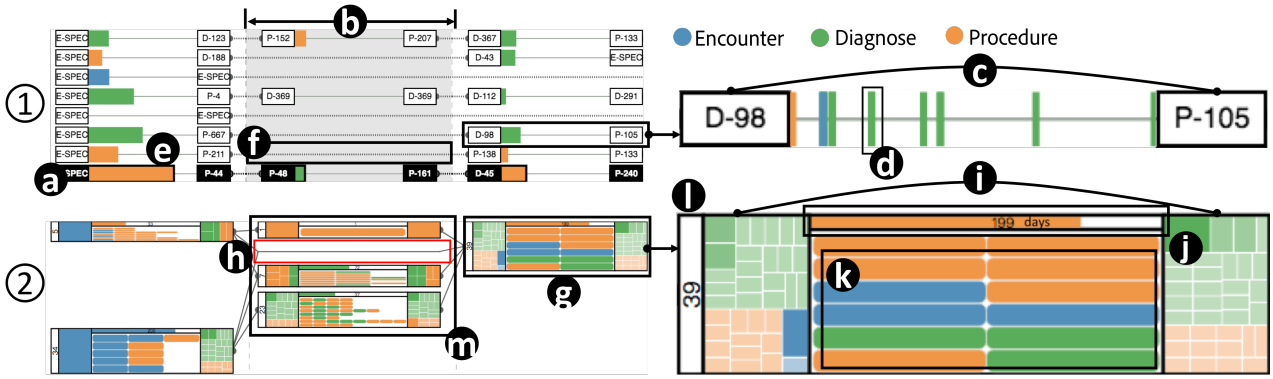

Fig. 5. The visual design of the (1)*sequence view* and (2)*cluster view*. The *sequence view* shows the progression analysis results for individual event sequences. Sequence segments in different stages are grouped into clusters and represented using rectangular nodes in the *cluster view*.

5.4 Cluster View

The *cluster view* (Fig. 5(2)) provides a higher-level summary of progression patterns of multiple event sequences by grouping similar segments at each stage. This view is critical for summarization, illustrating the most common sequential patterns (e.g., typical care plans within a patient cohort) and helping to identify segments that are more distinct (**T3**). To facilitate interpretation of the summary, the visual design integrates high-level progression patterns with low level event details within each progression stage.

5.4.1 Visual Design and Encoding

As illustrated in Fig. 5(2), each cluster of event sequence segments is displayed as a node (see Fig. 5(g)) which provides a visual summary of the cluster. Cluster nodes are arranged from left to right in the order of occurrence, and linked with edges to show transitions between clusters from stage to stage. The height of each cluster node is proportional to the number of event sequence segments contained in the cluster, with the label on the left side of each node providing the exact number (Fig. 5(l)). This node-link visual design was chosen to provide a flow-graph-like representation of the pathways through which sequences progress from stage to-stage over time, as well as the number of sequences which pass through each stage.

To help with interpretation, each node illustrates the frequencies of starting and ending events within a cluster (which often signal a transition in stage) as well as the frequent event patterns found within each cluster. The frequencies of starting and ending events are displayed at the left and right end of the node, respectively (Fig. 5(i)). We use treemaps to encode this 1-D data because it is spatially compact and allows for fast comparison across nodes [6]. Frequent sequential patterns are identified within each clustered stage based on VMSP (Vertical Mining of Maximal Sequential Patterns) [12], and visualized in the interior of the cluster node beneath the duration bar (Fig. 5(k)). Each row represents a unique sequential pattern, with color-coded bars representing key events within the pattern. Finally, the average duration of the segments within a cluster is encoded using the length of the bar located at the top of each stage (Fig. 5(j)). This overall design is intended to communicate high-level patterns (size and progression across stages) and support the semantic interpretation of individual stages (event types, frequency, and duration).

5.4.2 Layout Algorithm

Given the visual design outlined above, a critical and challenging aspect of the *cluster view* is the spatial layout of the cluster nodes. This section describes the novel layout algorithm used by ET^2 to address this challenge. The algorithm has two steps: (1) the insertion of intermediate nodes, and (2) the positioning of cluster nodes.

Insertion of intermediate nodes. Transitions from cluster to cluster (represented as edges between cluster nodes) can be categorized into two types: (1) regular transitions between adjacent stages, and (2) spanning transitions which extend across multiple stages. For regular transitions, clusters can be linked in straightforward fashion using a straight edge. For spanning transitions, which skip over columns in the visualization, naively drawing straight edges may result in overlaps between non-connected edges and nodes. To address this challenge, we first insert implicit "intermediate nodes" to convert spanning transitions into a series of regular transitions that connect the real nodes through a series of intermediate nodes. These intermediate nodes will not be rendered, but are used to aid in the layout process. To further simplify the visualization, when multiple intermediate nodes are inserted into a given section, they are clustered using the same Mean Shift Clustering algorithm used to cluster the regular nodes. The clustering algorithm is applied to intermediate feature vectors that are calculated by interpolating between the average feature vectors for the starting and ending clusters of the corresponding spanning transitions. This method helps produce simplifying groups of spanning edges (Fig. 5(h)).

Positioning of cluster nodes. The layout algorithm first sets the x-position of all cluster and intermediate nodes according to the stage of the corresponding segments, which aims to keep x-axis positions aligned between the *sequence view* and *cluster view*. It then calculates the y-position for each node based on a pairwise similarity score applied to nodes within each stage. Formally, the layout constraints are formulated as an optimization problem, which aims to minimize the following layout energy:

$$\sum_{t=0}^{T}(\alpha\sum_{i<j}\omega_{ij}^{t}\|y_i^t-y_j^t\|^2+(1-\alpha)\sum_i\sum_k\|y_i^t-\theta_{ik}\cdot y_{ik}^{t-1}\|^2)$$

where ω_{ij} is the inverse quadratic Euclidean Distance of the corresponding cluster centers of nodes i and j: $\omega_{ij}=1/\|v_i-v_j\|^2$, which is proportional to the similarity between node i and j, and y_i^t is the vertical position of a node i at stage t. θ_{ik} indicates the proportion of entities that flow through the k^{th} edge destined to node i. Intuitively, the first term minimizes the distance between nodes with higher similarity, while the second term preserves the layout position of each node across the transition flow. These two terms are balanced by a parameter $\alpha\in[0,1]$. The energy is minimized in an iterative procedure, with the adjacent stage pairs reordered (i.e., stage t and $t+1$, or stage t and $t-1$) at each iteration until the energy convergences.

To help users make a better comparison of the similarity between different stages, we further organize the nodes within the same stage into groups based on the y-position calculated using the procedure outlined above. This is achieved by grouping neighboring cluster nodes for which differences in y-axis positions are below a minimum threshold(as shown in Fig. 5(m)). The resulting groupings are used to structure the layout with vertical white space.

5.5 Other Views

A variety of other linked contextual views are also included in the system to enable a comprehensive representation of the data (Section 5.6), which in turn support users in flexible data exploration, pattern inspection, and comparison (**T5**).

Query View. The *query view* employs a milestone-based query capability [15] which allows users to select specific sequences that match a

user's analytical interests. Users can specify an ordered list of milestone events (Fig. 4(a)) and a range of sequence length length (Fig. 4(b)). In response, the system will retrieve only the event sequences which have the specific events appearing in the specified order for analysis.

Event Overview. The *event overview* illustrates the contextual information of each event through t-SNE projections of the event vectors defined in Section 4. Each event is represented using a color-coded circle based on the event category. The distance between two events reflects their contextual correlation, meaning that events that appear closer in this view have a higher probability of co-occurrence.

Stage Transition View. The *stage transition view* provides a very high-level summarization of events within each stage and the progression paths through which entities flow over time **(T4)**. Each stage is represented by a treemap which shows the frequency of events associated with a specific stage. The stages are linked with directed edges showing various progression paths that entities follow as they transit into and out of the stage. The width of each edge is proportional to the number of entities that follow the corresponding transition.

Thread View. The *thread view* displays the event sequence progression using the visual method first proposed in the EventThread [18] system. Here, each thread represents a latent evolution pattern, while nodes on each thread convey the events that are most strongly associated with a specific evolution path (Fig. 4(c)). We overcome a critical limitation of the original EventThread system (the use of fixed time intervals to segment event sequences) by incorporating the stage analysis results of the algorithm described in Section 4 to segment sequences to derive more meaningful latent evolution patterns.

Entity List View. The *entity list view*, as shown in Fig. 4(7), provides users with detailed information about individual entities, including the raw low-level event sequence data that serves as the original input for the ET^2 system .

5.6 Interactions

The ET^2 design includes a variety of user interactions which together support a variety of exploratory analysis tasks.

Dynamic stage manipulation. The ET^2 system allows users to manipulate the stage analysis results through three different operations: stage *merging*, stage *splitting*, and stage *folding*. Users can perform a merge on neighboring stages by selecting the stages to be merged in the *transition view*, then right clicking to raise a context menu (Fig. 4(d)). Users can also use the *transition view* to split a stage into two via the operations on a context menu. In a process that is similar to how split points are determined for the full event sequence, the stage analysis algorithm will automatically find the best possible alignment and segmentation of the stage to be split. Finally, users can *fold* a stage that they decide is unimportant for an analysis. This doesn't change the segmentation, but reduces the amount of information displayed to the interface as shown in Fig. 4(e). These interactions allow users to interactively adjusting the segmentation process and the resulting stages, with the goal of creating more meaningful latent patterns in both *cluster* and *thread views*.

Filtering. The system supports two different types of filtering: *entity filtering* and *event filtering*. First, users can filter out a group of one or more entities by clicking on a stage node or transition edge in the *stage transition view*. For example, patients moving through stage seven is filtered out in Fig. 4(f). As a result of this filter operation, the entities that remain in the visualization all jump over the filtered-out stage. Second, users can filter out specific types of events for analysis by brushing event circles in the *event overview* (Fig. 4(g)), or by interacting with the event drop-down list shown in Fig. 4(h).

Highlights and Tooltips. Linked-highlighting is used widely within various ET^2 views to help users identify corresponding elements such as events, stages, and transitions during interactive analysis. For example, when users click on a sequence in the *sequence view*, the entire sequence will be highlighted to help users locate corresponding segments in different stages. Similarly, when users hover over a cluster node in the *cluster view* or a stage node in the *transition view*, all of the transition paths for the corresponding entities will be highlighted to help users capture the associated progression patterns. Informative tooltips (Fig. 4(i)) are provided in all views to provide users with additional information in place as they navigate the visualization.

Details-on-Demand. A variety of user interactions support access to details-on-demand for different types of structures. As mentioned above, tooltips are available for many graphical marks. In addition, users can click on the duration bars in the *sequence view* to display a more detailed event-based timeline view of the segments (as mentioned in Section 5.3). In the *cluster view*, users can select individual cluster node to switch back to the *sequence view* to view more detailed information about the corresponding sequences. Users can also zoom into particular stages through brushing on the *stage transition view* (Fig. 4(j)). Moreover, the *event overview* and *entity list view* are updated interactively based on selections in other views to provide additional details about the selected items.

6 Evaluation

We evaluate the effectiveness of the ET^2 system via case studies, follow-up interviews with expert users, and a performance evaluation focused on scalability of the system.

6.1 Case Studies

We demonstrate the capabilities of ET^2 by analyzing real-world data from two domains: medicine and career advancement.

Case Study 1: Clinical Progression. We applied ET^2 to a public critical care dataset, MIMIC [22], which contains de-identified electronic health records of 46,521 patients who were admitted to the intensive care unit (ICU). Those records contained over 11,000 types of timestamped events, organized into seven categories, including hospital admission and discharge, death, ICU admission and discharge, prescriptions, infusions, laboratory tests, and microbiological test. We invited two experts to help us evaluate the usability of ET^2 system. The first expert (**E1**) was a cardiologist with 12 years of clinical experience, and the other one (**E2**) was his PhD student. During the study session, the experts explored the visual analysis result and provided feedback on its usability and usefulness. The process was recorded for subsequent analysis and discussed in Section 6.2.

After a brief introduction, the experts started the exploration by querying the dataset to retrieve a group of 145 patients diagnosed with cardiovascular disease. The system automatically analyzed the progression of this group and identified seven stages (Fig. 1). The experts briefly inspected the raw event sequences displayed in the *sequence view* before switching their focus to the *cluster view*. The major cluster nodes in each stage where most of the patients were involved immediately caught the experts' attention. They clicked on several of the nodes to highlight a major progression path and inspected the treemaps and frequent patterns. Next, they drilled down to explore each of the stages to identify semantically meaningful phases.

By reviewing the frequent patterns, the experts found the first stage captured events about hospital admission. They also noticed two medicines (Acetaminophen and Aspirin) were used for many patients for pain relief and the easing the cardiac infarction. A lab test for kidney function, Creatinine, was also commonly performed. "These are necessary actions for preventing renal injury when treating patients with cardiovascular diseases" the cardiologist said. Both of the experts were impressed by our design and believed that "the cluster view was very informative." They also confirmed that the details shown in treemap and the frequent patterns shown inside the stage nodes and the corresponding tooltips were very useful for allowing them to interpret the meaning of a stage.

After checking the event details shown inside stages two and three, the experts believed that these two latent stages belong to the same phase, in which the prescriptions such as Metoprolo (for reducing heart failure risk after myocardial infarction) and Insulin & Dextrose 5% in Water(D) (for decreasing potassium levels in blood serum) were made based on results of a set of routine lab examinations such as blood tests.

The fourth stage splits into two groups that attracted the experts' attention. They investigated and comparing these two groups, discovering that they illustrated two different treatment plans for different patients. "The patients in the upper group have acute coronary syndromes" said

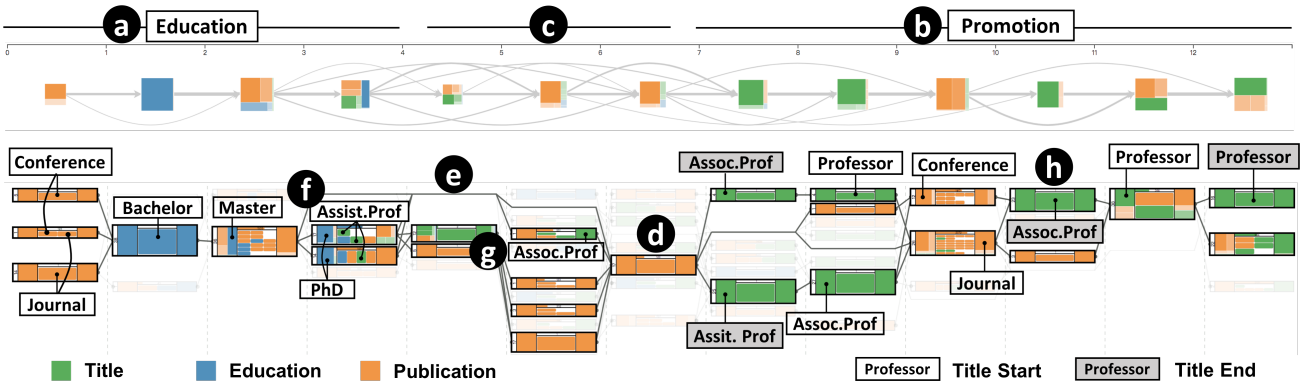

Fig. 6. The progression analysis result of scholars' academic behaviors. ET^2 produced 13 progression stages, from which three higher-level phases can be derived: (a) education, (b) promotion, and (c) transition period.

by the doctor, "this is why Integrelin and Heparin are used to reduce the risk of death." They also identified the corresponding key events for the group. In comparison, patients in the lower group took Calcium Gluconate, which is used to treat Hyperkalemia. The doctor confirmed the usefulness of the stage analysis algorithm, commenting that "the automatic stage analysis results are very meaningful" and "it will be very useful and interesting to demonstrate the disease progression [in ET^2] when [our own] data is available." The other expert stated that "it is impressive and interesting to actually see these paths (the different treatment plans) [emerge from] the view."

The experts believed that the fifth and sixth stages were both related to operations after using the visualization to see that patients in these stages commonly received infusion treatments (e.g., Propofol and D5W). These medications are mainly used to maintain the state of general anesthesia and vital signs during surgeries. The medications Bisacodyl and Acetaminophen (used to relieve common postoperative symptoms such as constipation and fever) were also observed.

Finally, they labeled last stage as the "Discharge" phase after noticing many laboratory tests used to preparing patients for home. These sequences often end with a discharge event. In general, both experts appreciated ET^2's ability to automatic identify progression patterns, and confirmed that the visual designs of the *sequence* and *cluster views* effectively supported clinical interpretation and comparison of the stages.

Case Study 2: Academic Career Paths. We also tested ET^2 on an academic dataset consisting of career path milestone events for 39 university professors over 23 years. The data consists of 10 event types such as obtaining a degree, changing the job title, and publishing conference/journal papers. These were classified into three high-level categories: education, publication, and promotion. ET^2 generated 13 progression stages for this dataset (Fig. 6). A senior PhD student major in computer science was invited to participate in this study. We observed his behaviors and report the findings as follows.

The *stage transition view* immediately caught the user's attention, from which he successfully identified the overall trajectory of professors' academic career paths. "This view intuitively shows people are educated at the early stages and get promoted later." He also found publication events were evenly distributed throughout the entire process: "It seems all the professors are productive at any stage of their career." These observations highlight the usefulness of the *stage transition view*.

A detailed inspection on the cluster nodes inside each stage help the user further interpret the stage analysis results. Specifically, he found that stages 2-4 correspond to the events when people get their degrees, and that they occurred after they published a set of papers at stage 1 (Fig. 6(a)). The user confirmed that "the *cluster view* has provided a direct and clear [guidance]." Following a similar procedure, the user found that stages 5-7 indicate the phase in which people increase their publications and gradually work toward their first faculty position (Fig. 6(c)). Some outliers skipped this phase, jumping directly into the next phase (Fig. 6(e)). A detailed inspection of these career paths in the *sequence view* found that these outliers obtained assistant professorships right after graduation. This pattern was also visible among frequent patterns displayed in the cluster node at stage 4 (Fig. 6(f)).

He then noticed that the paths merged into one cluster at stage 7 and then split again into two branches, representing two different promotion paths. The top branch showed promotions from Associate Professor to Full Professor and the bottom branch represented promotions from Assistant Professor to Associate Professor. Moreover, the user found people progressing along the lower branch commonly received a subsequent promotion at stage 11 and became Full Professors at stage 12. He also observed a gap between their first and second promotions at stage 10 where a large number of publication events occurred, indicating a process of publication accumulation. The user was impressed by our system. He said, "although it takes time to read the diagram, the rich findings are quite impressive" and "the *[cluster] view* is very powerful in terms of illustrating many details in the progression context."

6.2 Expert Interviews

Expert users from the medical (**E1**,**E2**) and career path (**E3**) case studies took part in follow-up interviews to provide additional feedback. Their comments are summarized below.

Automatic Stage Analysis. All three experts commented generally on the usefulness of the automatic stage analysis supported in our system. In particular, **E1** believed our technique is useful for illustrating the disease progression process. **E2** mentioned that "we used to compare historical records of patients manually, which is laborious and time-consuming. This system can be really helpful in automatically finding out and revealing common patterns of the patients at each clinical stage". Both **E1** and **E2** suggested that "the technique will be more useful if it can be used to analyze a particular chronic disease such as COPD" (in contrast to the heterogeneous ICU data from MIMIC), and **E3** felt the stage analysis algorithm was able to produce "precise and meaningful results".

Visualization Design. All of the experts felt that the *cluster view* was highly effective. **E1** commented that "it is the most informative view" in the system and **E2** agreed. Both **E1** and **E3** felt that the design of the cluster nodes were very useful for interpretation. **E1** said: "the information displayed in the cluster node is very comprehensive...It can help us differentiate different treatment plans at each stage." **E3** also mentioned the frequent patterns shown inside the cluster nodes are "very meaningful", are "good summarizations of complex sequences", and can avoid "showing too much unnecessary details". When asked about the *sequence view*, **E1,2** felt the design successfully illustrated events in a progression context.

System. The experts felt that ET^2 is useful, and that it is generally easy to use after a short training period. **E1,2** were particularly interested in using ET^2 to explore their own data and suggested that we make the system available online. **E3** also suggested to use the system for analyzing and comparing careers paths in different fields. All the experts felt the "query mechanism" in our system is an "essential part" of the system. Despite these positive comments, however, the experts at times also felt slightly overwhelmed by ET^2's user interface as it contained "too many views" (**E1,3**), or contained "too much graphical information with too little text" (**E1,2**). **E1** explained: "I have to explore and interpret a large number of visual elements and event types,

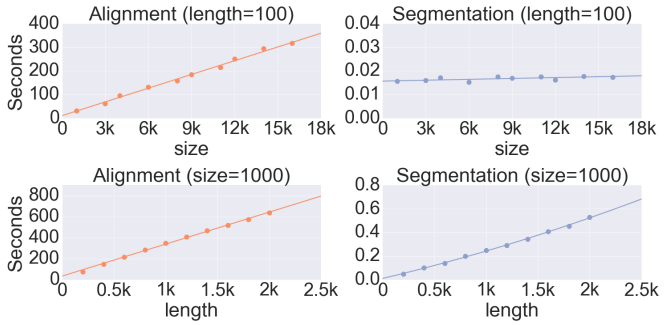

Fig. 7. The performance of two key algorithm steps when the size of the dataset and the averaged length of the sequences were changed.

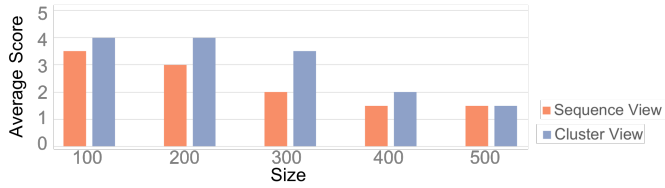

Fig. 8. The readability rating by the expert users of the *sequence view* and *cluster view* when showing datasets with different sizes.

..., I am more familiar with textual representations". However, they also agreed that after learning the design and encoding scheme, the interface became "more meaningful and easier to interpret". They also suggested to removing the *event overview* as it was often too dense or too sparse to be effectively used.

6.3 Performance Evaluation

We evaluated the scalability of ET^2 via both a quantitative evaluation of the algorithms and a qualitative evaluation of the visual designs. The datasets we used are all subsets collected from the MIMIC dataset.

Scalability of the Algorithm. To evaluate the stage analysis algorithm's scalalbity, we tested the performance of its key steps on a series of different sized event sequence collections (varying both the number of sequences and the average sequence length) sampled from the MIMIC dataset. Two key computation steps in the algorithm, alignment and segmentation, were tested on a four-core (Intel Core i5 CPU@3.2GHz) iMac computer with 8GB memory. The results, summarized in Fig.7, suggest that the performance bottleneck is the sequence alignment as DTW is computationally complex. Its performance depends strongly on the number of events, which is in turn determined by both the collection size and the average sequence length.

Scalablility of the Visual Designs. We examined the scalablility of the *sequence* and *cluster views* through a qualitative evaluation with the two medical expert users **E1,2**. The users were asked to examine results via the interface for event sequence collections of different sizes (from 100 to 500 with the step of 100) sampled from the MIMIC dataset. The averaged sequence length was around 800. The expert users were then asked to rate the readability of the views on a scale of 1 to 5 with 1 meaning most difficult to read (i.e., cannot find any patterns and cannot be interpreted at all) and a score of 5 meaning very easy to read (i.e., the pattern is clearly shown and easily interpreted). The results of these rating exercises are shown in Fig. 8 (mean value were reported).

In general, the *cluster view* was more highly rated when compared to the *sequence view*. However, the ratings for both views showed large drops when the data size reached 400. Follow-up discussions provided some explanations for these scores. **E1** commented that "over 300 sequences, there is no big difference of the results as both views contain too much information and are harder to interpret." When asked for detailed reasons, both **E1,2** mentioned the starting/ending treemaps inside each event node contained too many small elements to be viewed when the dataset is large.

The ratings for the *sequence view* were similar to the *cluster view* when the data size was small, and similarly but earlier drop as the data volume increased. **E2** explained that "exploring a small number of sequences is easy, as the visualization provides a good summary for each stage segment. But when the list of individual event sequences becomes long, it is very time-consuming to explore, especially when the sequences have large number of events." Both **E1,2** found it is very helpful to incorporate two views together for inspection. "The switch mechanism between two views is very powerful," **E1** commented, "It allows us to select only the clusters or paths of interest and inspect details in *sequence view*" to manage the scale.

7 Discussions

This section includes discussion regarding the generalizability and limitations of the proposed approach.

Generalization. The progression analysis algorithm can be easily generalized under the assumptions discussed in Section 4.1. Specifically, it is designed to analyze any event sequence dataset that follows a progression process as outlined in the paper, or any continuous temporal dataset which can be discretized into such kinds of sequences. There is no fundamental limitation on the sequence length, number of event types, intervals of time between events, or application domain. The visualization views are also designed based on the above assumptions and are specifically designed to visualize the stage analysis results output from our algorithm. However, the views are domain independent as shown by the two case studies reported above.

Scalability. The major limitation of ET^2 system is the scalability of the proposed algorithm and visualization views. Our experiments showed that both the algorithm and the primary visualization views can handle a few hundreds of event sequences within a reasonable response time and with a reasonable usability rating. To address this issue, a query mechanism is included in our system, which is designed to help users narrow down an analysis to focus on a subset of event sequences following a similar progression path within a larger dataset.

Validation. Similar to most unsupervised learning algorithms, the quality of the analysis results relies heavily on the design of the objective function and the quality of the optimization process. However, the lack of ground truth measurements for stage progression analysis makes the accuracy difficult to quantify. This paper, therefore, quantifies the computational performance of the algorithm and presents users' subjective feedback. However, it is important in future work to further assess the algorithm's accuracy. To this end, we plan to collect benchmark datasets with known ground-truths that would enable more quantitative accuracy measurements.

8 Conclusion

This paper presented ET^2, a visual analysis technique designed to support interactive progression analysis of event sequence data. ET^2 incorporates a novel progression analysis algorithm to identify semantically meaningful progression stages with variable time intervals. We also proposed a novel multi-view visualization design along with a set of rich interactions which allow users to explore and interpret the extracted stages, their progression over time, and the underlying even data. We evaluated the effectiveness of our approach in multiple ways: (1) we applied ET^2 to real-world data in multiple domains; (2) we accurately identified the evolution through stages of a real-world dataset with a known ground truth, and (3) we reported qualitative feedback gathered through domain expert interviews. While these results are promising, there are a number of important directions for future work. We plan to evaluate the usability of our system via a formal user study. We also plan to observe domain experts using our system in their daily work to discover ways to improve the designs of our system, and to better understand their analytical needs. Finally, we plan to explore methods of incorporating users' feedback into the progression analysis algorithm with the aim of further improving the accuracy of the results.

Acknowledgments

Nan Cao is the corresponding author. We would like to thank all users and domain experts who participated in our study, and all reviewers for their valuable comments. This work is a part of the research supported from NSFC Grants 61602306, Fundamental Research Funds for the Central Universities, NSFC Grants 61672231, STCSM 15JC1401700, and the NSFC-Zhejiang Joint Fund for the Integration of Industrialization and Information under Grant No. U1609220.

REFERENCES

[1] A. A. Alemi and P. Ginsparg. Text segmentation based on semantic word embeddings. *arXiv:1503.05543*, 2015.

[2] J. Alon, S. Sclaroff, G. Kollios, and V. Pavlovic. Discovering clusters in motion time-series data. In *IEEE CVPR*, vol. 1, pp. I–I, 2003.

[3] S. Arora, Y. Li, Y. Liang, T. Ma, and A. Risteski. Random walks on context spaces: Towards an explanation of the mysteries of semantic word embeddings. *arXiv preprint arXiv:1502.03520*, 2015.

[4] J. Ayres, J. Flannick, J. Gehrke, and T. Yiu. Sequential pattern mining using a bitmap representation. In *ACM SIGKDD*, pp. 429–435, 2002.

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

[6] N. Cao, D. Gotz, J. Sun, and H. Qu. Dicon: Interactive visual analysis of multidimensional clusters. *IEEE TVCG*, 17(12):2581–2590, 2011.

[7] B. C. M. Cappers and J. J. van Wijk. Exploring multivariate event sequences using rules, aggregations, and selections. *IEEE TVCG*, 24(1):532–541, 2018.

[8] Y. Chen, P. Xu, and L. Ren. Sequence synopsis: Optimize visual summary of temporal event data. *IEEE TVCG*, 24(1):45–55, 2018.

[9] F. Du, C. Plaisant, N. Spring, and B. Shneiderman. EventAction: Visual analytics for temporal event sequence recommendation. In *IEEE VAST*, pp. 61–70, 2016.

[10] F. Du, C. Plaisant, N. Spring, and B. Shneiderman. Finding similar people to guide life choices: Challenge, design, and evaluation. In *ACM SIGCHI*, pp. 5498–5544, 2017.

[11] F. Du, B. Shneiderman, C. Plaisant, S. Malik, and A. Perer. Coping with volume and variety in temporal event sequences: Strategies for sharpening analytic focus. *IEEE TVCG*, 23(6):1636–1649, 2017.

[12] P. Fournier-Viger, C.-W. Wu, A. Gomariz, and V. S. Tseng. Vmsp: Efficient vertical mining of maximal sequential patterns. In *Canadian AI*, pp. 83–94. Springer, 2014.

[13] P. Fournier-Viger, C.-W. Wu, and V. S. Tseng. Mining maximal sequential patterns without candidate maintenance. In *ADMA*, pp. 169–180, 2013.

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

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

[16] D. Gotz, S. Sun, and N. Cao. Adaptive contextualization: Combating bias during high-dimensional visualization and data selection. In *ACM IUI*, pp. 85–95, 2016.

[17] D. Gotz, F. Wang, and A. Perer. A methodology for interactive mining and visual analysis of clinical event patterns using electronic health record data. *Journal of Biomedical Informatics*, 48:148–159, 2014.

[18] S. Guo, K. Xu, R. Zhao, D. Gotz, H. Zha, and N. Cao. Eventthread: Visual summarization and stage analysis of event sequence data. *IEEE TVCG*, 24(1):56–65, Jan 2018.

[19] S. Hong, M. Wu, H. Li, and Z. Wu. Event2vec: Learning representations of events on temporal sequences. In *APWeb and WAIM Joint Conference on Web and Big Data*, pp. 33–47. Springer, 2017.

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

[21] 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.

[22] A. E. Johnson, T. J. Pollard, L. Shen, H. L. Li-wei, M. Feng, M. Ghassemi, B. Moody, P. Szolovits, L. A. Celi, and R. G. Mark. Mimic-iii, a freely accessible critical care database. *Scientific Data*, 3:160035, 2016.

[23] E. Keogh and C. A. Ratanamahatana. Exact indexing of dynamic time warping. *KAIS*, 7(3):358–386, 2005.

[24] J. Kiernan and E. Terzi. Constructing comprehensive summaries of large event sequences. *ACM TKDD*, 3(4):21, 2009.

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

[26] A. Marcus, M. S. Bernstein, O. Badar, D. R. Karger, S. Madden, and R. C. Miller. Twitinfo: aggregating and visualizing microblogs for event exploration. In *ACM SIGCHI*, pp. 227–236, 2011.

[27] T. Mikolov, K. Chen, G. Corrado, and J. Dean. Efficient estimation of word representations in vector space. *arXiv:1301.3781*, 2013.

[28] M. Monroe, K. Wongsuphasawat, C. Plaisant, B. Shneiderman, J. Millstein, and S. Gold. Exploring point and interval event patterns: Display methods and interactive visual query. *University of Maryland Technical Report*, 2012.

[29] T. Mori, A. Takada, Y. Iwamura, and T. Sato. Automatic human life summarization system in sensory living space. In *IEEE SMC*, vol. 2, pp. 1583–1588, 2004.

[30] 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.

[31] 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.

[32] A. Perer and F. Wang. Frequence: Interactive mining and visualization of temporal frequent event sequences. In *ACM IUI*, pp. 153–162, 2014.

[33] Q.-K. Pham, G. Raschia, N. Mouaddib, R. Saint-Paul, and B. Benatallah. Time sequence summarization to scale up chronology-dependent applications. In *ACM CIKM*, pp. 1137–1146, 2009.

[34] C. Plaisant, R. Mushlin, A. Snyder, J. Li, D. Heller, and B. Shneiderman. Lifelines: using visualization to enhance navigation and analysis of patient records. In *The Craft of Information Visualization*, pp. 308–312. Elsevier, 2003.

[35] S. Salvador and P. Chan. Toward accurate dynamic time warping in linear time and space. *Intelligent Data Analysis*, 11(5):561–580, 2007.

[36] 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. *The National Academy of Sciences*, 99(7):4703–4707, 2002.

[37] C. D. Stolper, A. Perer, and D. Gotz. Progressive visual analytics: User-driven visual exploration of in-progress analytics. *IEEE TVCG*, 20(12):1653–1662, Dec 2014.

[38] 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.

[39] 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.

[40] T. D. Wang, C. Plaisant, B. Shneiderman, N. Spring, D. Roseman, G. Marchand, V. Mukherjee, and M. Smith. Temporal summaries: Supporting temporal categorical searching, aggregation and comparison. *IEEE TVCG*, 15(6), 2009.

[41] X. Wang, D. Sontag, and F. Wang. Unsupervised learning of disease progression models. In *ACM SIGKDD*, pp. 85–94, 2014.

[42] Wikipedia contributors. One-hot — Wikipedia, the free encyclopedia, 2018. [Online; accessed 9-July-2018].

[43] Wikipedia contributors. Softmax function — Wikipedia, the free encyclopedia. `https://en.wikipedia.org/w/index.php?title=Softmax_function&oldid=848046651`, 2018. [Online; accessed 9-July-2018].

[44] K. Wongsuphasawat and D. Gotz. Outflow: visualizing patient flow by symptoms and outcome. In *IEEE VisWeek Workshop on VAHC*, pp. 25–28. American Medical Informatics Association, 2011.

[45] 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 *ACM SIGCHI*, pp. 1747–1756, 2011.

[46] C. Xie, W. Chen, X. Huang, Y. Hu, S. Barlowe, and J. Yang. Vaet: A visual analytics approach for e-transactions time-series. *IEEE TVCG*, 20(12):1743–1752, 2014.

[47] H. Xu, W. Wu, S. Nemati, and H. Zha. Patient flow prediction via discriminative learning of mutually-correcting processes. In *Data Engineering (ICDE), 2017 IEEE 33rd International Conference on*, pp. 37–38. IEEE, 2017.

[48] J. Yang, J. McAuley, J. Leskovec, P. LePendu, and N. Shah. Finding progression stages in time-evolving event sequences. In *ACM WWW*, pp. 783–794, 2014.

[49] J. Zhao, Z. Liu, M. Dontcheva, A. Hertzmann, and A. Wilson. Matrixwave: visual comparison of event sequence data. In *ACM SIGCHI*, pp. 259–268, 2015.