APPLIED PROCESS MINING, RECOMMENDATION, AND VISUAL ANALYTICS

by

SEN YANG

A dissertation submitted to the

School of Graduate Studies

Rutgers, The State University of New Jersey

In partial fulfillment of the requirements

For the degree of

Doctor of Philosophy

Graduate Program in Electrical and Computer Engineering

Written under the direction of

Ivan Marsic

And approved by

_____

_____

_____

_____

New Brunswick, New Jersey

JANUARY, 2019

ABSTRACT OF THE DISSERTATION

APPLIED PROCESS MINING, RECOMMENDATION AND VISUAL ANALYTICS

by Sen Yang

Dissertation Director: Dr. Ivan Marsic

Process mining techniques attempt to extract non-trivial knowledge and insights from activity logs and use them for further analyses. The traditional process mining focuses on addressing three different problems: workflow discovery, conformance checking and model enhancement. Although many theoretical studies have been done in the process mining domain, studies that applying process mining on solving real-world problems are limited. This dissertation explores how process mining can be used in real-world process analysis to reveal process insights and help human decision making. Novel algorithms and frameworks were proposed to better model and address the real-world problems. In addition, we introduced the recommender system into the process mining domain to help build a data-driven decision support system. Specifically, this dissertation includes three main contributions: (1) application of process mining techniques in real-world medical process analysis; (2) two different process recommender systems; and (3) a process visual analytic tool.

First, we applied process mining techniques to real-world medical process analysis. To enhance the existing workflow discovery algorithm, we developed a splitting-based workflow discovery method. Our method is able to tackle the duplicate-activity problem by allowing the activity nodes in the model to further split. By comparing our discovered model to hand-made expert workflow model of the same process, we were able to find the discrepancies between work-as-done and work-as-imaged. To further quantify and analyze the discrepancies between work-as-done and work-as-imaged, we invented a framework for automatic process deviation detection. Our framework first compares the observed

process traces with knowledge-driven workflow models using a phase-based conformance checking algorithm. The discrepancies (process deviations) were analyzed and false alarms were identified. The false alarms were categorized into three types of causes: (1) model gaps or discrepancies between the model ("work as imagined") and actual practice ("work as done"), (2) errors in activity trace coding, and (3) algorithm limitations. The deviation detection system was then repaired according to the false alarms. With our framework, the deviation detection accuracy was improved from 66.6% to 98.5%. The output system was then applied on unseen datasets to automatically detect the deviations. We applied our framework to two different medical processes and discovered meaningful medical findings. In addition, to analyze the differences between the medical treatment procedures of different patients, we introduced a framework for analyzing the association between treatment procedures and patient cohorts. The framework works by learning weights of context attributes by best-first search, deciding patient cohorts using clustering algorithms, discovering treatment procedures (or patterns) with process mining techniques, and analyzing the cohort-vs.-procedure through statistical analysis.

Second, existing recommender systems have not been developed based on process mining. Our work presents such a bridge. We designed a data-driven process analysis and recommender system that can provide contemporaneous recommendations of process steps and help with retrospective analyses of the process. We first designed a prototype-based recommender system. This approach relies on mining historic data to uncover the potential association between the way of enacting a process and contextual attributes. If association tests are significant, we train a recommender system to output a prototypical enactment for the given context attributes. The system recommends all steps at once. Although it may not be feasible for the performers to study and follow a long list of steps, this recommendation can be used at runtime to automatically verify the process compliance and detect omitted steps and other process errors. Later, we proposed another recommender system that is able to provide step-by-step recommendations. The system was built on recurrent neural networks. The networks took both environmental and behavioral contextual information as input and output next-step suggestions.

Last, we implemented our methods into a visual analytic tool. The tool was named as VIT-PLA, which is short for Visual Interactive Tool for Process Log Analysis. In this tool,

we proposed a prototype-based process data visualization strategy. The strategy works by first clustering process data into clusters and then discovering the prototypical procedure from each cluster. Only such cluster prototypes were visualized and presented to the users. Our strategy can greatly reduce the data amount to visualize but preserve the characteristics of each cluster. Statistical analyses were followed and visualized to help analysts better understand their process data.

Acknowledgements

First and foremost, I would like to express my deepest gratitude to my advisor, Prof. Ivan Marsic, for his guidance and support in the past four years. He brought me to this amazing project and was dedicated to educating and directing me throughout my Ph.D. journey. His great encouragements, inspirations, advice and feedback lead me to a thorough understanding of my research. I also enjoyed the freedom he gave me and benefited from the independent research opportunities. I feel very fortunate to work with him.

Next, I would like to express my gratitude to Prof. Xiong Hui from Rutgers Business School. Since my research focuses on applied data mining and analytics, there are not many people with a similar background in our department. Ever since I took Prof. Xiong's data mining class, he has been a helpful tutor and friend. In addition, I would like to thank Dr. Randall S. Burd from Children's National Medical Center. Dr. Burd is very important to my Ph.D. study. His medical domain knowledge is valuable to me and to my papers. And, with the second major in math, he gave me countless guidance on statistics in my data analytics. He is an outstanding mentor that inspires me all the time. I am also grateful to Prof. Sarcevic Aleksandra from Drexel University, who I worked together in the past four years and helped me in the paper writing. I would also like to thank all the medical experts that I have collaborated with, Dr. Rachel Webman, JaeWon Yang, Richard A. Farneth, Dr. Omar Z. Ahmed, and Megan Cheng. They are insightful and knowledgeable. Without their help, I could not achieve this.

I would like to thank Prof. Anand D. Sarwate, Prof. Shantenu Jha, Prof. Desheng Zhang and Prof. Yingying Chen, for their time to serve on my committee and provide me with suggestions on improving the dissertation.

In addition, during my graduate school journal at Rutgers, I received tremendous help from my lab colleagues and friends. I would like to thank my lab colleagues, Xinyu Li, Yue Gu, Moliang Zhou, Shuhong Chen, Xin Dong, Jingyuan Li, Weiqing Ni, Haiyue Ma, and Fei Tao. I would also like to thank all friends who have accompanies me along the journey.

Finally, my special appreciation is dedicated to my family. Their love is the compass that guides me to greater heights.

Table of Contents

Part I

Introduction

Part I: Introduction

Chapter 1 Introduction	Chapter 2 Preliminaries

Part II: Applied Process Mining and Analysis

Chapter 3 Workflow Model Discovery	Chapter 4 Workflow Deviation Analysis	Chapter 5 Patient Cohorts Analysis

---

Part III: Process Recommender System

Chapter 6 Trace-level Recommendation	Chapter 7 Activity-level Recommendation

---

Part IV: Implementation and Conclusion

Chapter 8 VIT-PLA	Chapter 9 Conclusions

Chapter 1
Introduction

1.1 Introduction and Motivation

Process mining is a relatively new research field that sits between data mining and business process management. In process mining, specialized data mining algorithms are applied to activity or event logs to identify the insights and knowledge. Existing studies in the process mining have several gaps. This dissertation attempts to uncover these gaps and provides solutions. First, although many research and techniques have been conducted or developed in process mining filed in recent years, we found many limitations and challenges when applying the process mining techniques to real-world processes. The limitations mainly come from methods' accuracy, computational complexity, interpretation ability, robustness and generality. Hence we proposed our own solutions to address these limitations and evaluated them on real-world process datasets. Second, traditional process mining studies focus on process diagnosis, i.e., descriptive analysis. We extend the current process mining research to operational support level. We brought the predictive model and recommender systems into the process mining domain and contributed two different process recommender systems. Third, existing tools for process data visualization and analysis are limited. We developed a visual analytic tool for process data, providing several different visualization strategies and statistical analyses. Here is the detailed introduction of these three studies.

1) Applied Process Mining and Analysis

We developed novel process mining methods and applied them to real-world medical process analysis. First, to enhance the existing workflow discovery algorithm, we developed a splitting-based workflow discovery method. Our method is able to tackle the duplicate-activity problem by allowing the activity nodes in the model to further split. Second, to quantify and analyze the discrepancies between work-as-done and work-as-imaged, we invented a framework for automatic process deviation detection. This

framework provides a method for identifying repeated, omitted and out-of-sequence activities that can be included in the design of decision support systems for complex medical processes. Third, to analyze the differences between the medical treatment procedures of different patients, we introduced a framework for analyzing the association between treatment procedures and patient cohorts. The framework works by learning weights of context attributes by best-first search, deciding patient cohorts using clustering algorithms, discovering treatment procedures (or patterns) with process mining techniques, and analyzing the cohort-vs.-procedure through statistical analysis.

2) Process Recommender System

Existing recommender systems have not been developed based on process mining. Our work presents such a bridge. We designed a data-driven process analysis and recommender system that can provide contemporaneous recommendations of process steps and help with retrospective analyses of the process. We first designed a prototype-based recommender system. This approach relies on mining historic data to uncover the potential association between the way of enacting a process and contextual attributes. If association tests are significant, we train a recommender system to output a prototypical enactment for the given context attributes. The system recommends all steps at once. Although it may not be feasible for the performers to study and follow a long list of steps, this recommendation can be used at runtime to automatically verify the process compliance and detect omitted steps and other process errors. Later, we proposed another recommender system that is able to provide step-by-step recommendations. The system was built on recurrent neural networks. The networks took both environmental and behavioral contextual information as input and output next-step suggestions.

3) Visual Analytic Tool for Process Data Analysis

We implemented our methods into a visual analytic tool. The tool was named as VIT-PLA, which is short for Visual Interactive Tool for Process Log Analysis. In this tool, we proposed a prototype-based process data visualization strategy. The strategy works by first clustering process data into clusters and then discovering the prototypical procedure from each cluster. Only such cluster prototypes were visualized and presented to the users. Our strategy can greatly reduce the data amount to visualize but preserve the characteristics of

each cluster. Statistical analyses were followed and visualized to help analysts better understand their process data.

The presented approaches, frameworks and tools were evaluated with several real-world datasets. As we have a partnership with Children's National Medical Center, we were able to continuously access valuable domain knowledge and feedback on our methods. This research is supported by a 4-year NIH project, i.e., Smart Trauma Resuscitation Decision Support System. Existing studies have shown that critically-injured patients have up to a four-fold higher risk of death from errors than general hospital patients [12], with nearly half of these preventable deaths related to errors that occur during the initial resuscitation phase of treatment [13][14]. Although a standardized trauma resuscitation protocol has been shown to improve the care of injured patients, human errors are still commonly observed and can contribute to adverse outcomes. Hence we built a decision support system using process mining and data mining techniques for monitoring medical team (trauma resuscitation team) behaviors, extracting data-oriented insights, and providing real-time alerts or recommendations. With this system, we aimed to reduce medical team errors and improve patient outcomes.

1.2 Dissertation Outline

The rest of the dissertation is organized as follows. In Chapter 2 of Part I, we introduce the background knowledge, two real-world medical processes and process mining techniques. In Part II, we applied and evaluated our process mining methods and frameworks on real-world processes. Specifically, in Chapter 3, we propose a novel workflow mining algorithm. In Chapter 4, we describe our framework to identify the process deviations. In Chapter 5, we bridge the gap between patient cohort analysis and process mining analysis. In Part III, we propose two different recommender frameworks, the prototype-based framework (Chapter 6) and the recurrent neural network based framework (Chapter 7). In the last Part (IV), we show the implementations of our visual analytic tool, VIT-PLA.

Chapter 2
Preliminaries

In this chapter, we give preliminary knowledge that highly relevant to this dissertation. First, we provide a description of the two medical processes, trauma resuscitation and intubation, that frequently used in our study. Then we introduce the process mining and related techniques.

2.1 Related Medical Processes

Two real-world medical process datasets were used throughout my Ph.D. project funded by NIH. Both of the two datasets were collected in Children's National Medical Center (CNMC), a level 1 trauma center in Washington D.C. Since these two datasets were used heavily in this dissertation, we would like to provide in this section the data descriptions, data exploration results and data characteristics analysis.

2.1.1 Trauma Resuscitation Process

Trauma resuscitation [15][16] is the process that the trauma team works together to assess and treat the patients who are severely injured (Figure 2.1). The trauma team includes bedside physicians, bedside nurses (left nurse, right nurse and charge nurse), and other team roles (e.g., surgical coordinator, respiratory therapist). A junior reside or nurse practitioner usually takes the role of bedside physician, depending on provider availability. The entire trauma resuscitation process mainly includes two medical phases, primary survey and secondary survey. The primary survey consists airway (assessing airway patency), breathing (assessing breath sounds and adequate oxygenation), circulation (assessing extremity pulse and managing blood loss), disability (assessing neurological status), and exposure (removing all clothes and managing hypothermia). The secondary survey consists of a multi-step, head-to-toe physical examination of the patient's body.

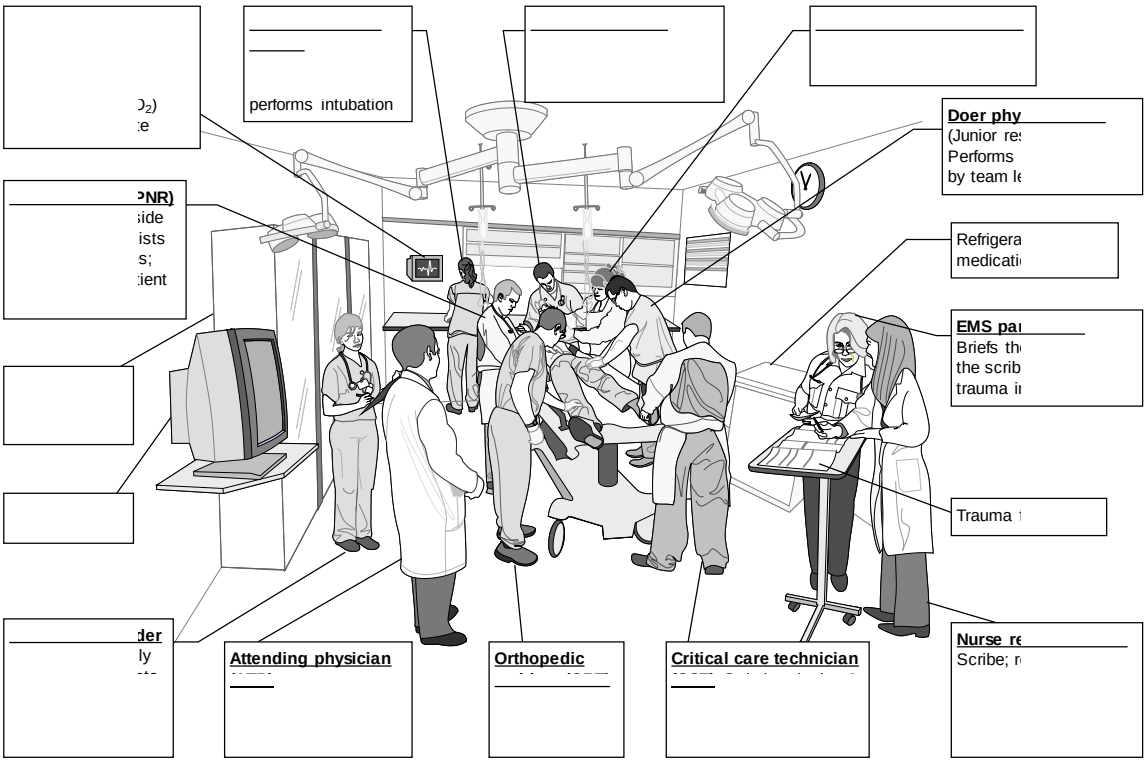


Figure 2.1 The trauma team in the trauma bay. Source of this figure (Fig. 1. in our paper "Teamwork errors in trauma resuscitation" [17]).

We started collecting the trauma resuscitation process data from August 2014. We performed selective sampling by including only the cases with patients who were admitted to the hospital following the resuscitation when errors and error management were more likely to have an impact on patient outcome. The process data was coded using retrospective video review of the sampled trauma resuscitations. The videos were recorded using the surveillance cameras installed in the trauma bays. The use of video recordings has been approved for use for research purposes by the Institutional Review Board at Children's National Medical Center. The videos were manually reviewed in Studiocode[1] (a video analysis tool) to identify the set of activities performed during the resuscitation, start and end times for each activity, the role performing the activity. Our studies (Chapter

[1] https://vosaic.com/support/category/studiocode

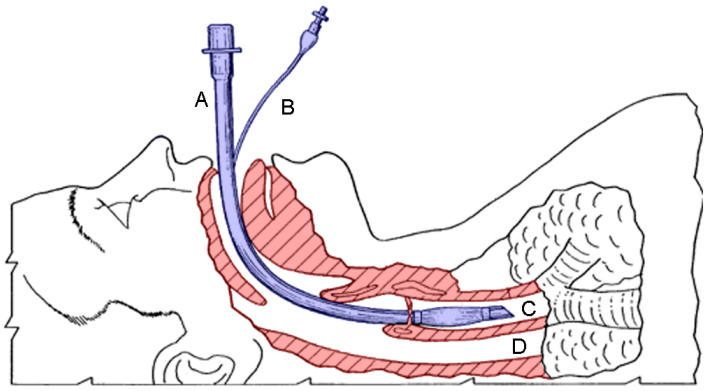


Figure 2.2 Diagram of an endotracheal tube that has been inserted into the patient's trachea. Source of this figure (https://en.wikipedia.org/wiki/Tracheal_intubation).

3 – Chapter 8) were conducted at different times of the data collection. Hence I report in my dissertation the trauma resuscitation datasets of different size. The details of the datasets used are described in each work below accordingly.

2.1.2 Tracheal Intubation Process

Endotracheal intubation is a procedure by which an endotracheal tube is inserted through the mouth down into the patient trachea (Figure 2.2). This is done because the patient cannot maintain their airway, cannot breathe on their own without assistance, or both. This may be because they are given anesthesia or they may be severely injured.

We collected our intubation dataset between February 2014 and February 2016. We included all patients less than 21 years old undergoing RSI in the emergency department resuscitation rooms. Although the intubation process may occur as part of the trauma resuscitation process, our intubation dataset excluded trauma patients because the trauma team in CNMC is a different hospital-based team with intubations managed by anesthesiology.

2.2 Process Mining and Related Techniques

Process mining [18], an interdisciplinary research field between data mining and business process management, aims to discover, monitor and improve real-world processes by extracting knowledge from activity (event) logs. An “activity log” is a collection of process cases, which contains a trace of “activities”. An activity is a well-defined action or step in the process. It is usually denoted with the activity type, start time and end time. In this dissertation, we prefer the term “activity” to “event” because intuitively “event” only emphasizes the “occurrence” and it does not have a duration. On the other hand, “activity” not only emphasizes the occurrence of an action but also indicates it may last for a period of time.

The process mining techniques attempt to tackle three problems, process discovery, process conformance checking and process enhancement (Figure 2.3). Process discovery techniques take activity or event logs as inputs and produce workflow models without using any prior information. Process conformance checking algorithm compares workflow models with activity or event logs to measure the level of compliance and identify process deviations. Process enhancement aims to improve the a-priori workflow models with observed activity or event logs.

Process discover (a.k.a. workflow discovery) is the most essential process mining task. Based on the activity (or event) log, a process model is constructed to capture the behavior observed in the log. The problem of automated process discovery has been intensively researched in recent years. Despite a rich set of proposals [19], the process discovery methods suffer from two major deficiencies when applied to real-world processes [20]: (1) they produce complex and spaghetti-like models; and (2) they produce models that either poorly fit the event log or over-generalize it. In addition, since process mining has half of its DNAs from business process management. Business process modeling languages (i.e., graphical representations for specifying business processes in a business process model, such as Petri nets [21], Declarative models [22], Business Process Model and Notation (BPMN) [23]) were used for workflow representation. Such notations can be confusing and uncomfortable for people outside this domain. Hence, in the computer science domain, we can simply interpret the process models as “(probabilistic) graphical models”.

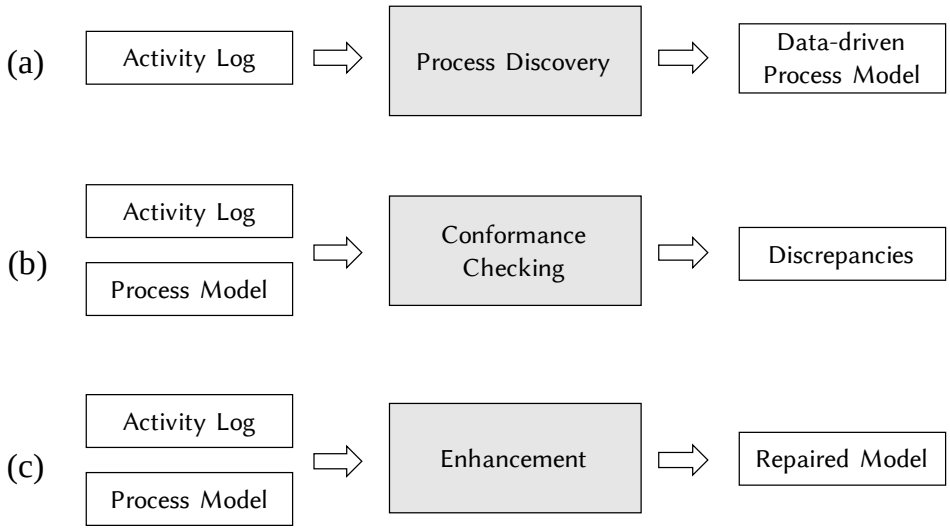


Figure 2.3 Three major tasks in process mining: (a) process model discovery, (b) process conformance checking, and (c) process model enhancement.

Conformance checking [24] is another essential process mining task that aims to analyze whether reality, as recorded in an activity log, conforms to the given workflow model and vice versa. The goal is to detect the discrepancies and quantify such discrepancies. The measure used to quantify the degree of discrepancies (or compliance) between the activity log and model is called fitness score, ranging from 0 (fully non-compliant) to 1 (fully compliant). The fitness score (Eq. 2.1) is defined as one minus the ratio between the number of deviations N_d from the expert model and the number of tasks N_t in a process trace.

$$\text{fitness} = 1 - \frac{N_d}{N_t} \tag{2.1}$$

Model enhancement [25] aims to extend and improve a process model using information extracted from the activity log. The model enhancement is actually duplicate with conformance checking. Because discrepancies between the activity logs and the corresponding process model can be discovered after conformance checking. The model can then be repaired with the discovered discrepancies.

Part II

Applied Process Mining and Analysis

Part I: Introduction

Chapter 1
Introduction

Chapter 2
Preliminaries

Part II: Applied Process Mining and Analysis

Chapter 3
Workflow Model Discovery

Chapter 4
Workflow Deviation Analysis

Chapter 5
Patient Cohorts Analysis

Part III: Process Recommender System

Chapter 6
Trace-level Recommendation

Chapter 7
Activity-level Recommendation

Part IV: Implementation and Conclusion

Chapter 8
VIT-PLA

Chapter 9
Conclusions

Chapter 3
Medical Workflow Modeling Using Alignment-Guided State-Splitting HMM

This chapter on Medical Workflow Modeling with Alignment-Guided State-Splitting HMM is based on our paper [4][5]. Process mining techniques have been used to discover and analyze workflows in various fields, ranging from business management [18] to healthcare [51]. Much of this research, however, has overlooked the potential of hidden Markov models (HMMs) for workflow discovery. We present a novel alignment-guided state-splitting HMM inference algorithm (AGSS) for discovering workflow models based on observed traces of process executions. We compared the AGSS to existing methods (ML-SSS [34], MDL [36], heuristic approach [28], and STACT [37]) using four real-world medical workflow datasets and a more detailed case study on one of them. Our numerical results show that AGSS not only generates more accurate workflow models, but also better represents the underlying process. In addition, with trace alignment to guide state splitting, AGSS is significantly more efficient (by a factor of O(n)) than previous HMM inference algorithms. Our case study results show that our approach produces a more readable and accurate workflow model that existing algorithms. Comparing the discovered model to the hand-made expert model of the same process, we found three discrepancies. The discrepancies were recognized as mismatches between “work as done” (actual practice described in the discovered model) and “work as imagined” (hand-made expert model). These three discrepancies were reconsidered by medical experts and used for enhancing the expert model.

3.1 Introduction

The application of workflow discovery and analysis in the medical field has the potential to improve patient outcomes. In the past, medical experts carefully designed medical workflow models, but actual practice often deviates from a perceived ideal process [26].

Models discovered from real process data provide information about the actual executions, and are critical for understanding process errors, e.g., omitted and duplicate activities. In addition, many clinical workflows do not have a predetermined workflow model. Process model extraction is then essential for discovering workflows more representative of their actual executions. We present a novel hidden Markov model (HMM) inference algorithm derived from existing work [27][28] to efficiently discover representative workflow models from medical processes.

Existing workflow discovery algorithms cannot provide the optimal workflow models. These methods assume that duplicate activities in a process trace are equivalent. Based on our analysis of trauma resuscitation workflow and previous work, each occurrence of an activity may have different underlying "intentions". For instance, over the course of a single trauma resuscitation, the trauma team may check the patient's eyes at two different points in time for different reasons. During the primary survey they assess the patient's pupillary response for neurological disability. During the secondary survey the team may examine the eyes in more detail, looking for injuries to the cornea, sclera and eyelids. To discover an accurate workflow model, an algorithm should be able to distinguish the first eye check from the second one, despite their identical labels. To our knowledge, no existing workflow mining algorithm can properly model such duplicate activities. Recent work has presented a strategy to refine duplicate activity labels, but only during preprocessing [29]. We address this problem by modeling duplicate activities as different hidden states in an HMM.

While HMMs are well-studied in speech, handwriting, and bioinformatics, they have been overlooked in process mining for several reasons. First, HMM inference is computationally demanding due to the iterative maximization procedures (i.e., Baum-Welch (BW) algorithm). Second, classical BW HMM inference depends heavily on subjective and labor-intensive parameter initialization (i.e. finding the number of hidden states, observation vectors, transition matrices). Finally, the resulting HMM may be too complex for knowledge acquisition purposes; the interpretation of hidden states can be challenging even for simple processes.

We also studied the balance between model accuracy and generality. If a workflow discovery algorithm only pursues model accuracy, it may overfit the observed process data,

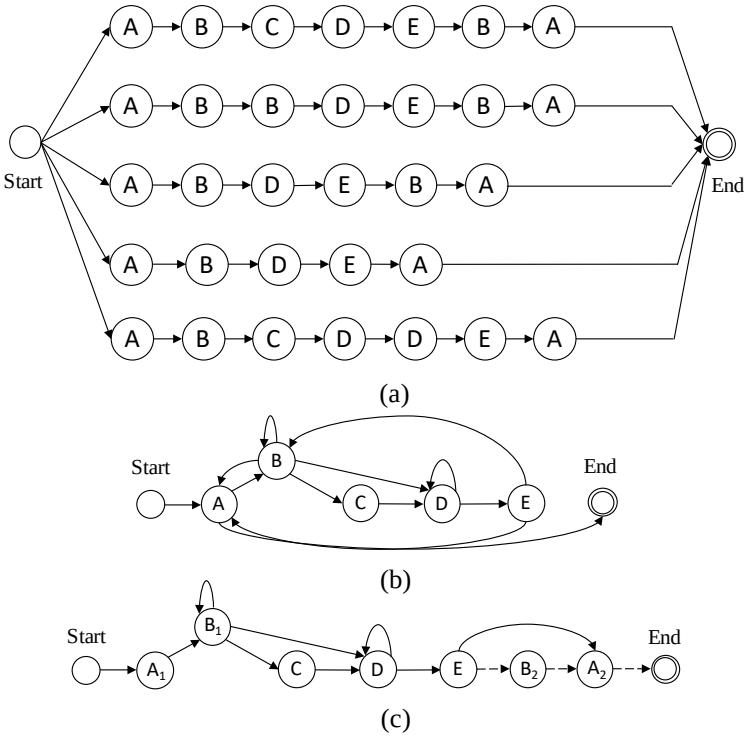


Figure 3.1 (a) The most specific workflow model of process log *O*; (b) a typical workflow model discovered based on existing Markov chain modeling methods (probabilities not shown); (c) the possible underlying workflow model that generated *O*.

making the model too specific. If the algorithm does not represent specific features of the process, the model may underfit the observed data. In both cases, the derived models are biased from the ground-truth process that generates the observed data. Consider a simple process log with five observed traces where letters represent activities performed in the shown order: ***O*** = {ABCDEBA, ABBDEBA, ABDEBA, ABDEA, ABCDDEA}. Three different representative workflow models may be discovered based on ***O*** (Figure 3.1). The most specific workflow model (a), a very general workflow model (b), and the ground-truth workflow model that was used to generate the example traces (c). Although model (a) describes the observed process traces very accurately, it lacks generality to represent unseen traces generated by the same process. Model (b) would be discovered by existing process mining algorithms [18]. It represents each activity by a single workflow node and overlooks potentially different "intentions" of the repeated activities. Our empirical study

based on medical knowledge shows the activities of the same name but occurring at different locations (i.e., sequential order in the execution) are usually performed due to different intentions. This model (b) allows the existence of sequences that are improbable in the underlying model (c). For example, activity A may go directly to "End", or ABCDE may loop many times. On the other hand, the underlying model (c) used to generate the traces has repeated activities A (A_1, A_2) and B (B_1, B_2). A_2 and B_2 occur at the end of process executions, rather than in the loop shown in model (b). We present an algorithm that uncovers workflows like (c) from a process log.

Our AGSS algorithm uses trace alignment—a data-driven algorithm—to guide HMM model inference. Trace alignment can discover the consensus sequence, or the backbone procedure of the observed process traces [30]. The trace alignment algorithm alone, however, is not sufficient for model mining because it only models sequential relationships between activities, and cannot handle parallelism. This limitation is not an issue for HMMs, a graphical model that can easily represent branches. Instead, HMMs face problems with model parameter initialization and computational complexity. Since trace alignment is able to find the distribution of the activities, incorporating alignment into HMM inference can help avoid subjective parameter initialization, boost inference speed, and produce better HMMs. Our contributions in this study are:

- A novel alignment-guided state-splitting HMM inference algorithm (AGSS) for real-world workflow discovery. Taking advantage of alignment, AGSS can efficiently find which states to split and how many states to split. This way, the complexity of the induced HMM is controlled by both the activity distribution in the alignment matrix and the dataset size (number of observed process traces). We compared AGSS to existing state-splitting-based HMM inference algorithms. The performance was evaluated using data from four real-world medical processes, in three different aspects: (1) How representative the induced model is of the observed process traces. (2) How close the induced model is to the underlying model that generated the observed process traces. (3) The computational complexity of HMM inference.
- An HMM simplification algorithm that reduces the number of insignificant transitions and extracts a fully connected backbone workflow model from a

spaghetti-like HMM model. The simplified model is more readable and allows for easier knowledge extraction.

3.2 Related Work

Most HMM inference methods are used for supervised classification, with heuristic initialization $\lambda_0 = (\pi_0, A_0, B_0, n, m)$. Given an approximate HMM topology, these algorithms fine-tune the HMM with the Baum-Welch algorithm [31]. Without appropriate initial parameter, the inferred HMM can be heavily biased from the ground truth. Other HMM inference algorithms implement either state-merging or state-splitting to address the initial parameter problem [27][32].

The state-merging approach was first proposed by Stolcke, et al. [32]. It begins the inference with the most specific topology, containing one path for each observed trace (as in Figure 3.1 (a)), and iteratively merges states to maximize the posteriori (MAP) [32]. The state-merging approach however is computationally expensive [28][33] because of two reasons. First, as initializing the HMM with all observed traces usually requires many nodes, calculating observation probability $P(\boldsymbol{O}|\lambda)$ can be expensive. Second, the merging method suffers from a lack of search direction and may stop too early, achieving local convergence far from the global optima.

The state-splitting approach, proposed by Takami and Sagayama [27], infers HMMs from the opposite direction. It begins with a general HMM and successively splits states until convergence. Compared to merging, the splitting approach is faster. Because the initial model is small, it is generally much closer to the final desired model [28][34][35]. Given observations $\boldsymbol{O}$, current state-splitting algorithms can be generalized into two steps. Step 1: determine the best state to split in each iteration *j*:

$$\lambda_j = \underset{\lambda_{j-1}^s \in \boldsymbol{\mathcal{M}_{j-1}}}{\arg\max}\ \text{Score}(\lambda_{j-1}^s, \boldsymbol{O}) \tag{3.1}$$

Step 2: determine when to stop splitting:

$$\text{Score}(\lambda_j, \boldsymbol{O}) \leq \text{Score}(\lambda_{j-1}, \boldsymbol{O}) \tag{3.2}$$

where λ_{j-1}^s is a candidate model for splitting state *s* at iteration *j*–1, $\boldsymbol{\mathcal{M}}_{j-1}$ includes all the candidate models at iteration *j*–1, and the scoring function Score(λ, $\boldsymbol{O}$) quantifies how well

the model λ fits the observations $\boldsymbol{O}$, as well as the model penalty. In other words, greedy splitting is done on all possible splits, and stops when further splitting does not increase the score. Both steps are dependent on the scoring function. It is difficult to analyze greedy splitting's effect on the resulting model but the stopping mechanism clearly influences the model's accuracy and generality. Splitting too much will overfit, sacrificing representativeness for accuracy. Splitting too little will underfit, leading to a model with poor accuracy. Maximum likelihood successive state-splitting (ML-SSS) [34] terminates splitting either when the likelihood $P(\boldsymbol{O}|\lambda)$ saturates, or when the model size reaches a certain threshold. ML-SSS does not penalize model complexity, so it could theoretically keep splitting down to the most specific model. Different complexity penalties have been used in previous research. Mavromatis [36] used minimum description length (MDL). Herbst [28] used a heuristic score function. Siddiqi et al. [37] used Bayesian information criterion (BIC) for simultaneous temporal and contextual splitting (STACS). Despite the variety of scoring metrics and algorithms, none of this research is about deriving workflow models from complex workflow data. The previous works used state-splitting method to infer the HMMs. But the final goal is to use the trained HMMs for their classification tasks. My goal of using HMMs on workflow data is to derive descriptive models so that analysts can extract knowledge from them. Although these previous works can also be used for the same purpose, we are looking for an efficient algorithm that can produce more accurate and more interpretable workflow models.

3.3 Alignment Guided State-splitting HMM

3.3.1 Definitions and Data Formalization

We first define the terms and notations used later:

Definition 1: A **process log** $\boldsymbol{O}$ is a set of process traces composed of activity executions. A log with T process traces is denoted $\boldsymbol{O} = \{\boldsymbol{O}_1, \dots, \boldsymbol{O}_j, \dots, \boldsymbol{O}_T\}$. A process trace $\boldsymbol{O}_j$ is represented by $\boldsymbol{O}_j = \{a^{start}, a_1, \dots, a_i, \dots, a_l, a^{end}\}$, where a_i denotes the i-th activity in the trace and ℓ denotes the trace length. "start" (a^{start}) and "end" (a^{end}) symbols are added to mark the beginning and end of each trace.

Definition 2: Given a process log $\boldsymbol{O}$, an **alignment algorithm** $\mathcal{A}(\boldsymbol{O})$ forms an alignment matrix with the traces in $\boldsymbol{O}$ as rows and activities of the same type as columns. If for a given trace a matching activity cannot be found, a gap symbol "-" is inserted in the corresponding column (Figure 3.2 (a)). Alignment $\mathcal{A}(\boldsymbol{O})$ returns the consensus sequence $\boldsymbol{CS}$, which we use to guide the selection of splitting candidates $\boldsymbol{SC}$. When describing alignment algorithm computational complexity, we use L to denote the average length of alignment matrices in a pairwise alignment. Alignment has previously been used for workflow and activity pattern analysis [1][30].

Definition 3: A **consensus sequence** $\boldsymbol{CS}$ is generated from the alignment matrix. It contains the activities in alignment matrix columns. $\boldsymbol{CS}$ can be considered the "average", or "backbone" of the original traces. Due to variations in the execution of complex processes, some alignment columns contain only a small number of non-gap elements. We define "column frequency" as the fraction of non-gap elements (Figure 3.2 (a)), which correspond to how frequently the activity in this column was performed across different process executions. We apply a column frequency threshold ε to filter out columns with rare activities. We use $\boldsymbol{CS'}$ to denote the consensus sequence after the filtering. Splitting candidates $\boldsymbol{SC}$ are taken from $\boldsymbol{CS'}$. Note that we use a "relaxed" consensus sequence that considers the surrounding columns within a time window when calculating the frequency of a specific column. This is because parallel activities in the workflow may result in different alignment orderings across nearby columns, but should still be considered the same "intention".

Definition 4: An **HMM** λ is a statistical model for modeling temporal sequences with unobserved or hidden states. An HMM is denoted by a quintuplet $\lambda = (\boldsymbol{A}, \boldsymbol{B}, \boldsymbol{\pi}, n, m)$, where matrix $\boldsymbol{A}$ records the state transition probabilities, where $\boldsymbol{A}(i, j)$ represents the transition probability between state i and state j; $\boldsymbol{B}$ records the observation probability distribution in each state; $\boldsymbol{\pi}$ is a vector consisting initial state distribution; n is the number of hidden states and m is the number of distinct observation symbols per state. The **observation probability** $P(\boldsymbol{O}|\lambda)$ is the probability that the HMM λ will emit the set of observed traces $\boldsymbol{O}$. **Log-likelihood** (a.k.a. log-likelihood) $\log P(\boldsymbol{O}|\lambda)$ is usually used in implementations to avoid arithmetic underflow problems.

Definition 5: The **splitting candidates** $\boldsymbol{SC}$ comprise a set of activities eligible for state splitting during the topology inferencing of HMM λ. We use a_i in $\boldsymbol{SC}$ to denote a candidate activity to split in λ. The HMMs obtained by state-splitting are placed in the set **candidate models** $\boldsymbol{\mathcal{M}}$. *N* denotes the total number of splits done during the inferencing of λ.

3.3.2 Alignment Guided State-splitting HMM

3.3.2.1 AGSS algorithm

AGSS (see Figure 3.2) first initializes a general Markov chain λ_0. Unlike most state-splitting algorithms that start with only one state [27][35][37], AGSS begins with a Markov chain composed of one state per activity (step 1 in Alg.3.1), a strategy used in Herbst's heuristic state-splitting approach [28].

After initialization, AGSS determines two factors: which states to split and when to stop splitting. AGSS determines the splitting candidates from the alignment matrix and orders them by column frequency. Specifically, AGSS uses the following intuitions about the consensus sequence (step 2 in Alg.3.1): (1) activities of different columns of the alignment matrix likely have different purposes or intentions regardless of activity type; and (2) each column's frequency measures the importance or distinctiveness of the activity's intention. Having two different columns with the same activity type and frequencies above the ε threshold strongly indicates that this activity (e.g., "A and B" in Figure 3.2 (a)) has multiple intentions, qualifying the associated state for splitting candidacy (step 3 in Alg.3.1). It also intuitively follows, and our experiments have shown, that the candidates with higher column frequencies (e.g. "A" in last column of Figure 3.2 (a)) should be tested for splitting before those with lower frequencies (e.g. "B" in the column before last in Figure 3.2 (a)). Therefore, we sorted the candidates $\boldsymbol{\mathcal{CS}'}$ by descending frequency (step 4). Because each activity is already assigned a state in the initial model λ_0, there is no need to split the first occurrence of each activity in $\boldsymbol{\mathcal{CS}'}$. Therefore, we revoke the candidacy of first occurrences and return the final list $\boldsymbol{SC}$ (step 4).

After calculating the splitting candidates, AGSS performs iterative splitting. In iteration *j*, given a splitting candidate activity $a_i \in \boldsymbol{SC}$, we find states $\boldsymbol{S}$, in the model from last iteration λ_{j-1}, which have observations a_i (e.g., S_1 in Figure 3.2 (b). Step 7 in Alg.3.1). For

Algorithm 3.1. AGSS HMM

Input: $\boldsymbol{O}$, ε
Output: λ
Step 1. Initialize HMM topology as Markov chain λ_0;
/* Obtain candidate activities for splitting */
Step 2. Compute relaxed consensus sequence $\boldsymbol{CS} = \mathcal{A}(\boldsymbol{O})$;
Step 3. Filter consensus sequence $\boldsymbol{CS}' = \text{Filter}(\boldsymbol{CS}, \varepsilon)$;
Step 4. Find splitting candidates $\boldsymbol{SC}$ by sorting $\boldsymbol{CS}'$: $\boldsymbol{\mathcal{D}} = \text{Sort}(\boldsymbol{CS}')$, and removing the first occurrences of all activities: $\boldsymbol{SC} = \text{RemoveFirstOcurrences}(\boldsymbol{\mathcal{D}})$;
/* Perform state splitting */
Step 5. **for** activity a_i in $\boldsymbol{SC}$, **do**
Step 6. Create a temporal model $\lambda' = \lambda_{j-1}$;
Step 7. Find states $\boldsymbol{S}$ that have observation a_i from λ';
Step 8. **for** s_k in $\boldsymbol{S}$, **do**
Step 9. Split s_k to $s_k'$and s_k'' in λ';
Step10. Initialize observation and transitions of $s_k'$and s_k'';
Step11. Train with Baum-Welch algorithm: $\lambda' = \text{Train}(\lambda', \boldsymbol{O})$;
Step12. $\boldsymbol{\mathcal{M}} = \boldsymbol{\mathcal{M}} \cup \{\lambda'\}$;
Step13. **end for**
Step14. $\lambda_j = \arg\max_{\lambda' \in \boldsymbol{\mathcal{M}}} P(\boldsymbol{O}|\lambda')$;
Step15. **if** $\log P(\boldsymbol{O}|\lambda_j) < \log P(\boldsymbol{O}|\lambda_{j-1})$;
Step16. $\lambda_j = \lambda_{j-1}$;
Step17. **end if**
Step18. $j = j + 1$;
Step19. **end for**
Step20. $\lambda = \lambda_{j-1}$;
Step21. return λ;

each state s_k in $\boldsymbol{S}$, we split it into two new states s_k' and s_k'' (step 9 in Alg.3.1). The newly-split states (s_k' and s_k'') are assigned the same observation as their predecessor's state s_k. The transitions (i.e., transition probabilities from other states to s_k', s_k'' and from s_k', s_k'' to other states) can be initialized either randomly or by estimating the distribution of surrounding activities in the alignment matrix (step 10 in Alg.3.1). After initializing the parameters of $s_k'$and s_k'', λ' is trained with the Baum-Welch Algorithm to optimize the HMM parameters (step 11 in Alg.3.1). λ' that maximizes log-likelihood $P(\boldsymbol{O}|\lambda')$ is assigned λ_j (step 14 in Alg.3.1). Afterwards, λ_j is compared to λ_{j-1}; λ_j is kept only if it improves model log-likelihood (steps 15-17 in Alg.3.1).

Compared with existing state-splitting HMM inference algorithms [34][35][37], AGSS has the following advantages. First, as we associate each state with a single activity type and assign newly-split states their predecessor's activity, each state in the inferred HMM has only one associated activity. This way, the discovered models will have easily-

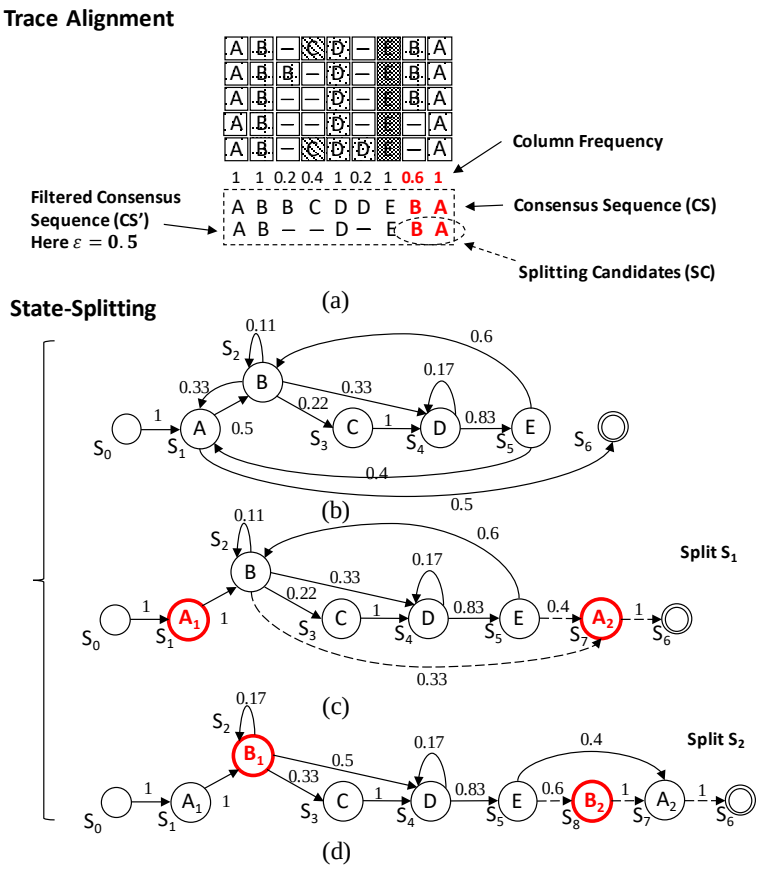


Figure 3.2. Illustration of the AGSS algorithm. (a) Splitting candidates $\boldsymbol{C}$ = {A, B} are selected from the consensus sequence. (b) The splitting starts from the initial model. (c) Activity A is split first because the column in which A is repeated has higher frequency (1.0) than that of activity B (0.6). (d) Activity B is split after A. The splitting stops when all splitting candidates are split.

interpretable topologies. Second, because we calculate all splitting candidates based on alignment, AGSS does not need to search the parameter hyperspace of all possible HMMs from all possible splits. Third, as the alignment results and consensus sequence selection are tailored to the characteristics of the data, our alignment-based strategy better captures the activity distribution. With these advantages, AGSS not only runs faster but is also more robust.

Algorithm 3.2. Estimating frequency threshold ε

Input: $\boldsymbol{O}$
Output: ε
Step 1.Randomly select half of the traces from $\boldsymbol{O}$, denoted as $\boldsymbol{O}_{half}$;
Step 2.Calculate consensus sequences: $\boldsymbol{CS} = \mathcal{A}(\boldsymbol{O})$; $\boldsymbol{CS}_{half} = \mathcal{A}(\boldsymbol{O}_{half})$;
Step 3.**for** $\varepsilon = 0 : r : 1$, **do**
Step 4. $\boldsymbol{CS}' = \text{Filter}(\boldsymbol{CS}, \varepsilon)$; $\boldsymbol{CS}'_{half} = \text{Filter}(\boldsymbol{CS}_{half}, \varepsilon)$;
Step 5. **break if** $ED(\boldsymbol{CS}', \boldsymbol{CS}'_{half})$ converges
Step 6.end for
Step 7.return ε

3.3.2.2 Estimating the ε Threshold

The frequency threshold ε is important for selecting the splitting candidates which, in turn, determine when splitting will terminate. The threshold ε for the frequency of non-gap symbols in columns ranges between 0 and 1. As it approaches 0, more activities will be allowed onto the candidate list; as it approaches 1, fewer states are candidates. We propose two threshold ε selection methods. The first method finds ε by running the inference algorithm with random ε values, and then picks the threshold yielding the best model (model evaluation discussed later). The apparent limitation is the long running time required to infer multiple models. The second method estimates ε from the data size (num. of traces) using Alg.3.2, where ED is the edit distance (a.k.a. Levenshtein distance [38]), between two filtered consensus sequences and r is an increment of ε. Alg.3.2 is based on the law of large numbers [39], which implies that as the data size increases (more process traces), the average of the data (consensus sequence) should converge to an expected value. In other words, the elements in consensus sequences will stabilize with more data.

To evaluate whether we have enough data for producing a stable consensus sequence $\boldsymbol{CS}$, we randomly select half of the input traces (step 1 in Alg.3.2) and calculate their consensus sequence $\boldsymbol{CS}_{half}$ (step 2 in Alg.3.2). $\boldsymbol{CS}$ and $\boldsymbol{CS}_{half}$ are filtered using the threshold ε (step 4 in Alg.3.2). The edit distance between $\boldsymbol{CS}'_{half}$ and $\boldsymbol{CS}'$ will approach 0 when data size is sufficient. The convergence occurs faster for larger ε (Figure 3.3(a)). Given the observed data, Alg.3.2 returns the smallest ε at which the edit distance converges. The convergence threshold is set to 0.05, which is the smallest normalized edit distance (i.e., edit distance normalized by the sum of trace lengths) between $\boldsymbol{CS}'_{half}$ and $\boldsymbol{CS}'$. In

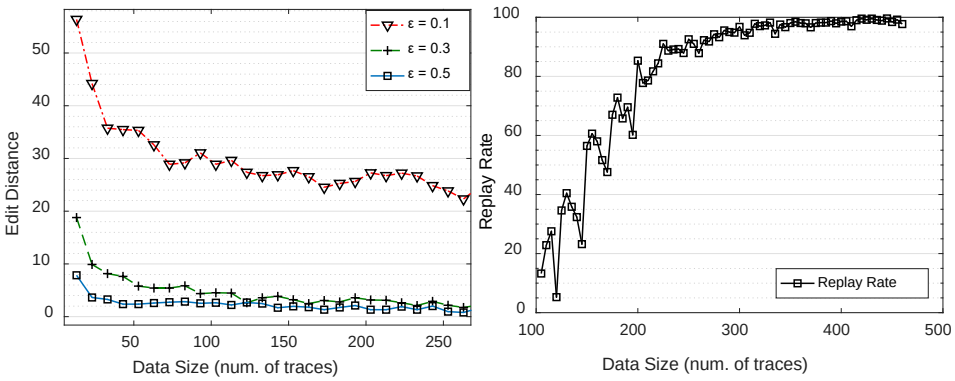


Figure 3.3. (a)Training dataset size plotted against edit distance between $\mathcal{CS}_{half}$ and $\mathcal{CS}$ using different frequency thresholds ε; (b) replay rate on test data. The results are averaged over 30 runs using the Dutch hospital data [40].

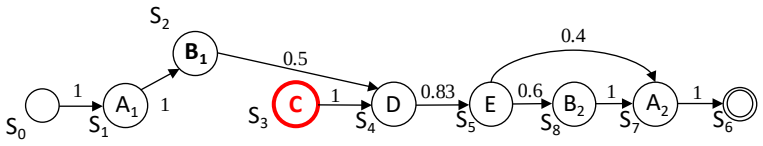


Figure 3.4. Flawed simplified workflow model with a dangling node S_3. This model is simplified from (d) in Figure 3.2 with an arbitrary transition cutoff as 0.35.

other words, convergence occurs when over 95% of the two consensus sequences are equal. The choosing of ε based on Alg.3.2. is automatic.

This ε is desirable because it selects the number of splitting candidates proportional to the size of the data. For smaller datasets, it would be inappropriate to split too much and overfit (i.e., states split to capture characteristics or patterns that only exist in specific observed traces). If the process log $\boldsymbol{O}$ is small, the two consensus sequences would be unstable, forcing a higher ε that preserves only the most common activities in the consensus sequence. This high ε produces a stable but sparse consensus sequence, therefore disqualifying most states from being split. On the other hand, for large datasets ε would be small, allowing the model to split more. The resulting model would be more detailed, which is preferable when training with sufficient data. AGSS then adaptively adjusts the number of potential splits to the amount of data available.

3.3.2.3 Simplifying and Pruning the Discovered Model

Data-derived workflow models can be spaghetti-like [18], i.e., complex graphs with a large amount of nodes and directed arcs (transitions). This finding is also true for the AGSS-induced HMM. When used for classification, spaghetti-like models are acceptable black boxes, because we only focus on classification accuracy and do not extract insights from the model structure. In workflow discovery, finding a descriptive and interpretable model is more important. This goal can be accomplished by model simplification.

Model simplification and pruning are techniques to reduce the size and complexity of workflow models by omitting insignificant information. The obvious solution is to apply transition frequency cutoffs, but this may produce a flawed workflow model with "dangling" process fragments (C in Figure 3.4). These dangling fragments represent impossible executions when put in the context of workflows. Removing insignificant transitions while keeping the graph a plausible workflow is a form of the NP-hard minimal spanning strong subgraph problem (where we artificially place a transition with frequency one from the end to the start). In the context of process mining, however, it is more reasonable to use heuristic methods.

In this work, we propose a heuristic HMM simplification algorithm (Alg.3.3), which can extract the skeleton (the backbone structure) of the HMM. The strategy preserves the dominating incoming and outgoing transitions for each state, reduces the number of insignificant transitions, and does not produce single dangling activities. It is, however, possible for a group of states to form a subprocess isolated from the main process (i.e., the process that includes the start state s^{start} and end state s^{end}). For example, a pair of states can construct a loop where the most significant in-and-out transitions go to each other. We address this problem by first discovering isolated subprocesses (steps 9-18 in Alg.3.3) and then reconnecting the subprocess to the main one through the entire subprocess's most significant in-and-out transitions (steps 19-25 in Alg.3.3). The output λ^s from algorithm 3 is the backbone model of observed processes $\boldsymbol{O}$.

Algorithm 3.3. HMM Simplification

Input: $\lambda = (\boldsymbol{A}, \boldsymbol{B}, \boldsymbol{\pi}, n, m)$
Output: λ^s
/* Preserve the dominating in-and-out transitions of each state */
Step 1. Initialize the simplified transition matrix $\boldsymbol{A}^s = \emptyset$;
Step 2. **for** $i = 0{:}1{:}n$, **do**
Step 3. | **for** $j = 0{:}1{:}n$, **do**
Step 4. | | **if** $\boldsymbol{A}(i,j)$=max($\boldsymbol{A}(i,:)$), $\boldsymbol{A^s}(i,j) = \boldsymbol{A}(i,j)$;
Step 5. | | **if** $\boldsymbol{A}(i,j)$=max($\boldsymbol{A}(:,j)$), $\boldsymbol{A^s}(i,j) = \boldsymbol{A}(i,j)$;
Step 6. | | **if** $i = j$, $\boldsymbol{A^s}(i,j) = \boldsymbol{A}(i,j)$; /* keep self-transition */
Step 7. | **end for**
Step 8. **end for**
/* Discover states isolated from the main process */
Step 9. Initialize a set with all states in λ except s^{start}: $\boldsymbol{S} = \{s_1, \ldots s_n\}$. Initialize a stack $\boldsymbol{S}^{(m)} = \{s^{\text{start}}\}$; /* s^{start} is s_0 */
Step 10. **while** $\boldsymbol{S}^{(m)} \neq \emptyset$
Step 11. | s_i=$\boldsymbol{S}^{(m)}$.pop();
Step 12. | **for** $j = 0{:}1{:}n$, **do**
Step 13. | | **if** $\boldsymbol{A}^s(i,j) \neq 0$ && $s_j \in \boldsymbol{S}$ /* i is the index of s_i */
Step 14. | | | $\boldsymbol{S}^{(m)}$.push(s_j);
Step 15. | | | $\boldsymbol{S}$.remove(s_j);
Step 16. | | **end if**
Step 17. | **end for**
Step 18. **end while**
/* Connect isolated subprocesses to the main process */
Step 19. **while** $\boldsymbol{S} \neq \emptyset$ /* $\boldsymbol{S}$ includes all states isolated from main process */
Step 20. | $s = \boldsymbol{S}$.get(0);
Step 21. | According to $\boldsymbol{A^s}$, find all states in $\boldsymbol{S}$ directly or indirectly connected to s and insert them to $\boldsymbol{S}^{(s)} = \{\ldots, s, \ldots\}$;
Step 22. | Find the most preceding state s_j or states $\boldsymbol{s}_j$ (when loop exists) in $\boldsymbol{S}^{(s)}$, $\boldsymbol{A^s}(i,j) = \underset{i,j}{\text{argmax}}\, \boldsymbol{A}(i,j)$, find the most succeeding state s_k or states $\boldsymbol{s}_k$ (when loop exists) in $\boldsymbol{S}^{(s)}$, $\boldsymbol{A^s}(i,k) = \underset{i,k}{\text{argmax}}\, \boldsymbol{A}(i,k)$;
Step 23. | Remove the states in $\boldsymbol{S}^{(s)}$ from $\boldsymbol{S}$;
Step 24. **end while**
Step 25. **return** $\lambda^s = (\boldsymbol{A}^s, \boldsymbol{B}, \boldsymbol{\pi}, n, m)$

3.4 Experiments

We compared AGSS to existing baseline state-splitting HMM inference methods, such as ML-SSS [34], MDL [36], heuristic approach [28], and STACT [37]. The performance was measured based on (1) model's quality and (2) the algorithm's computational efficiency. Both perspectives were quantitative (the induced HMMs were not interpreted by humans), so model simplification or pruning (Section IV.3.3.2.3) was not applied in this section. Experiments were run on a Dell desktop (Win 10, Intel Xeon 3.7GHz CPU, 48G RAM).

Table 3-1. Statistics of our four medical process datasets.

Stats \ Dataset	Intubation	Primary	Secondary	Dutch
Num. Patient Records	74	186	122	833
Num. Total Acts	900	1291	3057	24,550
Num. Act Types	15	8	17	18
Longest Trace (Num. Acts)	13	8	51	229
Shortest Trace (Num. Acts)	5	4	7	2
Avg. Num. Acts in Trace	12.16	6.94	25.06	29.47

3.4.1 Real World Medical Process Datasets

During the evaluation, we used activity logs from four real-world medical processes (). Obtaining and using the medical data for this study was approved by the Institutional Review Board at Children's National Medical Center in Washington, DC. Three medical logs were coded from surveillance videos of trauma resuscitations: (1) The endotracheal intubation (breathing tube insertion). (2) The initial evaluation phase (i.e., primary survey) of the trauma resuscitation [41], where the team looks for immediately life-threatening injuries. (3) The head to toe examination phase (i.e., secondary survey) of the trauma resuscitation. The fourth dataset was collected at a Dutch hospital and published by 4TU (http://researchdata.4tu.nl/home/) [40]. After removing outliers and rare activities, there remained 24,550 activities of 18 types in the Dutch hospital data.

3.4.2 Measuring Quality of Induced HMM

The challenge for HMM inferencing in workflow mining is its unsupervised nature; labeled ground truth data is not available to test the induced model. Previous state-splitting algorithms working with supervised classification problems had labeled data to test the performance [37]. Our goal in this study was to induce a descriptive process model, as opposed to a predictive one. This descriptive model can be evaluated by how well it represents observed and unobserved data [42]. To quantify the model quality, we adopted two concepts, *model fidelity* and *model confidence*, from previous research [43]. Model fidelity (or accuracy) measures the agreement between a given workflow model and the observed process traces, i.e., the log likelihood of generating the observed process traces using the given model. The value is high if the model structure accurately describes the

Algorithm 3.4. Calculating model confidence M_c

Input: $\boldsymbol{O}$
Output: M_c
Step 1. Initialize $M_c = 0$;
Step 2. Partition the observed traces $\boldsymbol{O}$ into 10 folds, $\boldsymbol{O}$={$\boldsymbol{O}_1$,…, $\boldsymbol{O}_{10}$};
Step 3. **for** $i = 1{:}\,1{:}\,10$, **do**
Step 4. Infer λ_i' with 9 folds except $\boldsymbol{O}_i$
Step 5. Replay $\boldsymbol{O}_i$ in λ_i' and calculate log-likelihood log$P(\boldsymbol{O}_i \mid \lambda_i')$
Step 6 Sum up log-likelihood M_c += log$P(\boldsymbol{O}_i \mid \lambda_i')$
Step 7. end for
Step 8. return M_c

observed process traces. Model confidence measures how well a workflow model represents the underlying process that generates the observed process traces. High model confidence is achieved if the model describes not only the observed traces, but also the unobserved realizations of the underlying process. Model confidence is difficult to achieve with insufficient training data, because the model may overfit the small dataset.

Given observed traces $\boldsymbol{O}$ and inferred model λ, model fidelity (M_f) is defined as the log-likelihood of the observed traces:

$$M_f = \log P(\boldsymbol{O}|\lambda) = \sum_{i\in\{1,2,\ldots,\ell\}} \log P(\boldsymbol{O}_i|\lambda) \tag{3.3}$$

where $P(\boldsymbol{O}_i \mid \lambda)$ represents the probability that the observed trace $\boldsymbol{O}_i$ is produced by model λ and ℓ is the average trace length.

The model confidence is hard to quantify because we have neither the underlying ground truth model, nor the entire set of possible process traces. Model confidence (M_c), can be estimated by partitioning the data into training and test sets. Let λ' denote the inferred model from training data $\boldsymbol{O}_{training}$. We define model confidence as log$P(\boldsymbol{O}_{test} \mid \lambda')$. In practice, we chose to use 10-fold cross validation (Alg.3.4) to reduce result variance. Real-world process logs are usually small, since they need to be coded manually by domain experts.

Some test data $\boldsymbol{O}_i$ may not be replay-able on the trained HMM λ' (i.e., probability $P(\boldsymbol{O}_i \mid \lambda')$=0). This problem usually occurs with sparse training data, where an activity or transition present in the test data is not in the training data. The probability $P(\boldsymbol{O}_i \mid \lambda')$ of such a trace would then be zero, despite missing only one activity or transition. This issue is

Table 3-2. Comparison of AGSS with existing state-splitting algorithms on four real world medical processes. Model fidelity (M_f) and model confidence (M_c) are scaled by the number of process traces in each process.

	Intubation			Primary Survey			Secondary Survey			Dutch Hospital		
	M_f	M_c	R_r	M_f	M_c	R_r	M_f	M_c	R_r	M_f	M_c	R_r
MC[1]	-12.8	-12.9	84.3%	-10.00	-10.38	96.6%	-47.59	-44.09	65.8%	-38.8	-37.3	97.6%
AGSS	-12.8	-12.9	84.3%	-9.98	-10.16	96.1%	-45.22	-44.13	65.8%	-32.76	-34.03	97.6%
ML-SSS[2]	-10.56	-22.05	98.6%	-9.05	-10.66	100%	-48.79	-60.52	100%	-52.39	-54.66	100%
Heuristic	-11.8	-15.8	84.3%	-8.50	-10.95	96.1%	-47.59	-44.10	65.8%	-38.6	-36.9	97.6%
MDL	-19.45	-19.78	100%	-12.37	-12.97	100%	-64.44	-65.37	100%	-81.34	-82.03	100%
STACT	-12.32	-14.23	98.6%	-9.32	-10.19	100%	-49.22	-58.08	99.2%	-44.86	-47.04	100%

[1]MC stands for Markov Chain

[2]The convergence threshold of ML-SSS is 0.01

usually addressed by smoothing the model parameters, a process of flattening the probability distribution so that all sequences can occur with some probability. Smoothing can improve HMM performance in a classification problem, but will not help mine workflow models. For model mining, adding unseen activities and transitions would only make the model. Instead of smoothing, we propose a replay rate metric R_r, the percent of test data can be replayed in λ'.

$$R_r = \frac{\sum_i \delta(P(\boldsymbol{O}_i|\lambda'))}{n} \times 100\% \tag{3.4}$$

where $\delta(x) = 1$ if $x \neq 0$, and $\delta(x) = 0$ otherwise; and n is the number of traces in the test set. R_r is strongly related to data size (Figure 3.3(b)), with R_r increasing as data size increases.

We computed model fidelity (M_f), model confidence (M_c), and replay rate (R_r) to evaluate the HMM inducing algorithms on different datasets (Table 3-2). As described above (Section 3.3.2), ML-SSS, MDL, and STACT were designed to start splitting from one node and allow different activity types to be observed at the same state. On the other hand, Heuristic and AGSS were designed to be initialized with a Markov chain. We also used tests on synthetic data to set a convergence threshold of 0.01 for ML-SSS to prevent too many splits [34] (splitting is terminated if the next iteration's log likelihood is less than 1.01 times that of the previous).

The results (Table 3-2) show that while AGSS does not achieve the best model fidelity on intubation and primary survey, it achieves the best model confidence on all datasets. Compared with its initial Markov chain topology, AGSS induces a model with better fidelity and confidence, indicating that the splitting improves the model. The heuristic approach, on the other hand, has higher model fidelity (-11.8 on intubation and -8.5 on primary survey) but lower model confidence (-15.8 on intubation and -10.95 on primary survey) than Markov chain. This finding indicates that the heuristic approach made unnecessary splits and overfit to the training data. Starting from a Markov chain and using the "one observation per state" constraint, both AGSS and the heuristic approach suffer from low replay rates. From a model mining perspective, however, these constraints become advantages in producing easy-interpretable workflow models. ML-SSS achieves high model fidelity using the Intubation dataset (-10.56) and Secondary Survey dataset (-48.79) but has low model confidence (-22.05 and -60.52), indicating it also overfit the data. MDL and STACT have lower model fidelity and confidence when compared to AGSS and Heuristic, indicating they penalize model complexity too severely and terminate the splitting too early. This is in agreement with the fact that MDL and STACT assume a data size much larger than the number of model parameters [36][37].

3.4.3 Computational Complexity Comparison

Existing state-splitting algorithms (e.g. ML-SSS, MDL, Heuristic, STACT) are computationally complex because the search for splitting candidates at each iteration is expensive. Each state is considered a candidate and is therefore tested. AGSS avoids this problem by initially discovering all splitting candidates from trace alignment. The overall computational complexity of AGSS is $\mathcal{O}(Tn^2N)$, consisting of trace alignment and splitting candidate evaluation. Trace alignment requires $\mathcal{O}(T^2\ell^2 + T^2\,log(T))$ to construct the guide tree and $\mathcal{O}(T^2L + TL^2)$ for progressive alignment [1][10]. State splitting requires $\mathcal{O}(Tn^2N)$ for N total state splits with $\mathcal{O}(Tn^2)$ for Baum-Welch parameter learning [31] and likelihood evaluation ("forward-backward") at each split. In comparison, other state-splitting algorithms generally take $\mathcal{O}(Tn^3N)$, where the number of splits N depends on the convergence rate of the score function. The computational complexities of

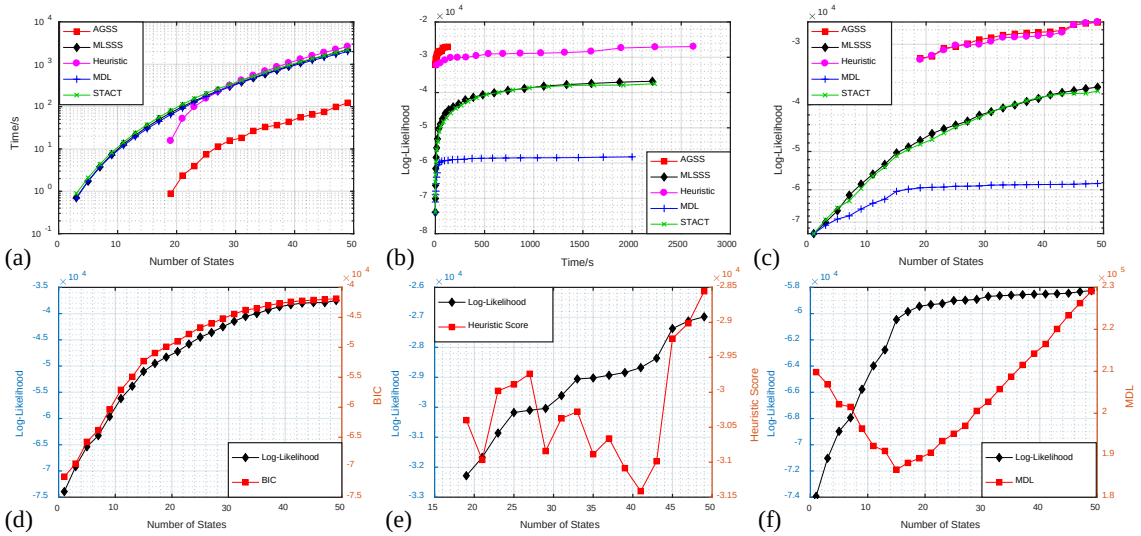


Figure 3.5. (a) The number of states vs. computation time in seconds; (b) the computation time vs. log-likelihood (i.e., log-likelihood); (c) the number of states vs. log-likelihood; (d) BIC and log-likelihood change as state splitting using STACT; (e) Heuristic score and log-likelihood change as state splitting using Heuristic approach; (f) MDL and log-likelihood change as state splitting using MDL. Experiments were run on Dutch hospital dataset [40].

AGSS and other state-splitting algorithms strongly depend on the average number of hidden states *n* because of its raised power.

When comparing the number of successful splits versus time on a logarithmic scale (Figure 3.5(a)), we ignored the terminating criterion for each algorithm and continued splitting until a given number of states *N'* (50 is used in Figure 3.5) was reached. AGSS showed a clear advantage in speed, using only 135.22 seconds. Other baseline algorithms took more than 2,000 seconds. AGSS and Heuristic have a slight advantage in that they start splitting from 17 states as opposed to one, but this is insignificant because it only takes less than 100 seconds for MLSSS, MDL, and STACT to split to 17 states. In the computation time vs. log-likelihood comparison (Figure 3.5(b)) , AGSS also outperformed baseline algorithms in log-likelihood (model fidelity) from the start and maintained the advantage through 50 states. Also, by starting from a Markov chain, AGSS and heuristic have initial log-likelihood -32,367, while the others start from -73,936. The algorithms starting from one node did not achieve the initial log-likelihood of AGSS and heuristic even after splitting 49 times. Another observation is that the splitting speed slows (Figure

3.5(b)) and the log-likelihood gains per split diminish (Figure 3.5(c)) as the number of states increase. Although log-likelihood kept increasing, model complexity penalties behaved differently (Figure 3.5(d)(e)(f)). BIC stayed close to the log-likelihood values, indicating a small penalty on the model complexity (Figure 3.5(d)). This small penalty leads to large HMMs. The heuristic score starts decreasing when the number of states arrives at 27 before beginning to oscillate (Figure 3.5(e)), indicating that further splitting could potentially produce a better model. MDL is a positive value and smaller values are rewarded. It first decreases, but and then starts climbing after 15 states (Figure 3.5(f)). MDL converges faster than STACT and heuristic, meaning that it more heavily penalizes model complexity. Compared with baseline methods, AGSS shows its unique boosting effect on HMM inference, saving much computational time (Figure 3.5) while providing a higher model quality (Table 3-2).

3.5 Case Study: Trauma Workflow Mining

AGSS is designed to discover workflow models without any prior information. To evaluate the workflow model discovered by AGSS (Figure 3.6 (d)), we compared it to models inferred from STACT (Figure 3.6 (c)), from Disco (Figure 3.6 (b), a process mining tool), and by medical experts using domain knowledge (Figure 3.6 (a)). The model discovery algorithm used in Disco (https://fluxicon.com/disco/) is based on a fuzzy workflow mining algorithm [44], rather than an HMM. Comparing the AGSS model to the Disco one shows that AGSS can find better workflow models than other process mining algorithms, regardless of whether they use HMMs. We tested different process mining algorithms (e.g., alpha, heuristic, and genetic miners [45]) and chose Disco as the baseline because it is the most widely used in process mining applications. Models discovered from highly-variable real-world processes are usually uninterpretable spaghetti-like diagrams. Models were simplified using Algorithm 3 before comparisons were made. Differences between the models have implications on both medical and model engineering interpretations of the underlying process.

First, we compare general readability. The Disco model (Figure 3.6 (b)) is simple, with transitions between nodes with one activity per node. The reverse transition from *Rectal-Back* to *Palpation-Head* can be confusing, because the secondary survey ends after the

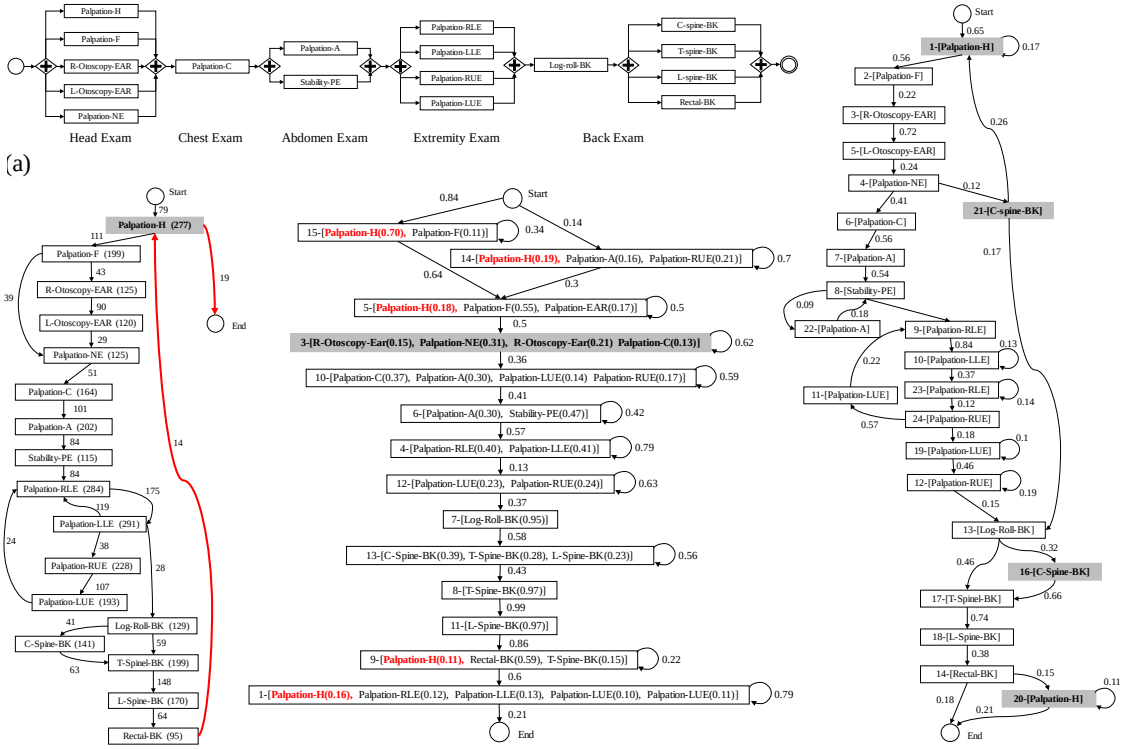


Figure 3.6. (a) Expert model designed by medical experts. (b) Simplified model discovered by Disco (redrawn for clearer view). The number in the node denotes the frequency of activity and the number on the edge denotes the frequency of the transition. (c) HMM induced by STACT where the transitions were simplified by Algorithm 3 and state observations with probability less than 0.1 were filtered. (d) HMM induced by AGSS and simplified by Algorithm 3.

back examination. This reverse transition can be misinterpreted as a start-over of the secondary survey, which is not true in practice. The linear structure of the model inferred by STACT is easy to follow and appears like a forward-moving workflow. The linearity comes from STACT allowing multiple observations in the same state. For example, an activity that occurs out of order, either early or late, can be simply represented by allowing it to occur at the “wrong” state. In Disco models, as each activity type is uniquely mapped to one node, such out-of-order activities are handled by adding new transitions to the “wrong” state. In AGSS models, these types of activities are handled differently based on their occurrence distribution. If one activity frequently occurs at multiple locations, then new states will be split to better describe those occurrences. Infrequent repeated

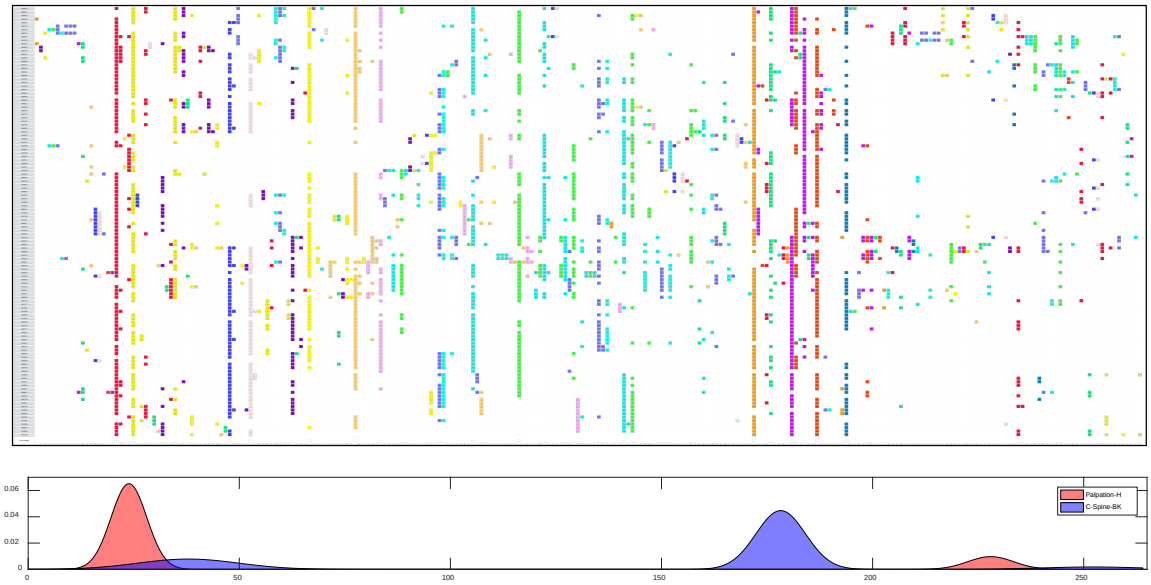


Figure 3.7. Trace alignment and distribution of activity “Palpation-H” and “C-Spine-BK”. In the alignment matrix, each row represents a different trace and each color represents an activity type. In the distribution plot (under the alignment), x-axis represents the column number (can be treated as “logic time”) of the alignment matrix and y-axis represents the probability. The trace alignment result is generated using our process visualization tool, VIT-PLA [2].

occurrences would be considered outliers, and would not be granted new states. The disadvantage of the STACT model is the difficulty of interpreting multiple activity types per state. For example, state #3 in Figure 3.6 (c) had more than four different activities. Another limitation is the large possibility of state self-transitions (i.e., transition from a state to itself), making the model implicit. AGSS explicitly describes the relationship between different activities. It also addresses the multiple occurrence problem not considered in Disco (i.e. *Palpation-H* was described using two different states, #1 and #20 (Figure 3.6 (d))).

Second, we discover more insights after comparing data-derived models to the expert model. Aside from the reverse transitions (from *Rectal-BK* to *Palpation-H*), the Disco model follows the expert model well. This finding, however, does not mean the Disco model is better than the other two data-driven models. Subjectively created by humans, the expert model may not reflect practice. For example, both the expert model and Disco failed

to capture the multiple occurrences of *Palpation-H* and *C-Spine-BK* (Figure 3.7). Unlike the others, STACT and AGSS captured multiple occurrences. For example, *Palpation-H* can be observed in states #1, #5, #9, #14, and #15 in the STACT model, as well as in states #1 and #20 of the AGSS model. While both STACT and AGSS were able to capture the multiple occurrences of activities, they do so very differently. AGSS balances the significance of activities' multiple occurrences before assigning a new node for them, while the STACT model simply creates new nodes for all multiple occurrences without considering complexity-vs.-accuracy tradeoffs. This explains why we can see a large number of activity recurrences in the STACT model (Figure 3.6 **(**c)). In comparison, AGSS added only six states (#19-#24 in Figure 3.6 (d)). As an example, state #20 (*Palpation-H*) is split from state #1, and state #21 (*C-Spine-BK*) is split from #16. The splitting is conformant to the distribution of these two activities (Figure 3.7), as both *Palpation-H* and *C-Spine-BK* occur at different times in the secondary survey. State #1 corresponds to the first cohort of *Palpation-H* while state #20 corresponds to the one close to the end. State #16 maps to the main cohort (between time 150 to 200) of *C-Spine-BK* and state #21 maps to the secondary cohort (between time 0 to 100). A small cohort of *C-Spine-BK* occurs at the end. It is however so small that AGSS did not split it when taking model complexity into account. This way, AGSS identified several discrepancies between practice and the expert model (Table 3-3). These discrepancies highlighted differences between what was being taught and what is being practiced during trauma resuscitations. These deviations from the expert model may be due to errors committed by the medical team, or may be necessary based on clinical circumstance. AGSS can determine whether deviations are part of a pattern of behaviors or isolated instances. This knowledge helps determine if these differences are acceptable, or if an enhancement of the expert model is needed.

Table 3-3. Discrepancies between expert model and practical procedures. Medical explanation and model enhancement.

Discrepancies	Medical Explanation	Model Enhancement
C-Spine-Back not only occurs during the back examination but also may occur during the head examination.	To determine whether the patient needs cervical spine (c-spine) support, it may be easier to palpate the c-spine and the neck simultaneously. From a clinical perspective, it is acceptable to perform c-spine assessment at either point in the secondary survey.	C-Spine-BK should be completed during head exam or during back exam. If it occurs at head exam, it is not mandatory to repeat again at back exam.
Palpation-Head not only occurs during the head examination but may also occur during the back examination.	The medical team logrolls the patient to perform the back exam without stressing the spinal column. To perform a complete head exam, medical team members often visualize or palpate the occipital region or back of the head during the logroll, so as to reduce head manipulation.	Palpation–Head can be performed during the back exam.
Palpation-Abdomen, Palpation-Right-Lower-Extremities, Palpation-Left-Upper-Extremities, Palpation-Right-Upper-Extremities were found to repeat in the process.	To more completely assess the patient, it is acceptable to repeat activities; for example, repeated palpation of bilateral extremities may occur to assess potential swelling or deformities.	The model should be repaired to allow repetitions of activities within a particular body region.

Chapter 4
An Approach to Automatic Process Deviation Detection in a Time-Critical Clinical Process

This chapter on Automatic Process Deviation Detection is based on our paper [11]. Prior research has shown that minor errors and deviations from recommended protocols in complex medical processes can accumulate to increase the likelihood that a major error will go uncorrected and lead to an adverse outcome. Real-time automatic and accurate detection of process deviations may help medical teams better prevent or mitigate the effect of errors and improve patient outcomes. Our goal was to develop an approach for automatic detection of errors and process deviations in trauma resuscitation. Using video review, we coded activity traces of 95 pediatric trauma resuscitations collected in a Level 1 trauma center over two years (2014-2016). Twenty-four randomly selected activity traces were compared with a knowledge-driven model of trauma resuscitation workflow using a phase-based conformance checking algorithm for detecting true and false deviations (alarms). An analysis of false alarms identified three types of causes: (1) model gaps or discrepancies between the model ("work as imagined") and actual practice ("work as done"), (2) errors in activity traces coding, and (3) algorithm limitations. We repaired the system to remove model gaps, reduce coding errors, and address algorithm limitations. The repaired system was first evaluated with another 20 traces and then applied to the entire dataset of 95 traces. During the training, we detected 573 process deviations in 24 activity traces that include 1,099 activities. Among these deviations, only 27% represented true deviations and the remaining 73% were false alarms. This initial deviation detection accuracy was only 66.6%, with a $F1$-score of 0.42. Detection accuracy of the repaired system increased to 95.2% (0.85 $F1$-score) during system validation and to 98.5% (0.96 $F1$-score) during testing. After deploying the repaired deviation detection system to all 95 activity traces, we detected 1,060 process deviations in 5,659 activities (11.2 deviations per resuscitation). Among the 5,659 activities in these traces, 4,893 fit the repaired knowledge-driven workflow model, 294 were errors of omission, 538 were errors of commission, and 228

were scheduling errors. Our approach to automatic deviation detection provides a method for identifying repeated, omitted and out-of-sequence activities that can be included in the design of decision support systems for complex medical processes. Our findings show the importance of assessing detected deviations for repairing a knowledge-driven model that best represents “work as done.”

4.1 Introduction

Trauma resuscitation—the initial evaluation and treatment of injured patients in the emergency department—is a dynamic medical setting in which multidisciplinary teams often perform life-saving interventions under time pressure
. Because injured patients can rapidly deteriorate, the resuscitation process requires efficiency and accuracy. Although a standardized protocol (Advanced Trauma Life Support [ATLS] [16][46]) has been shown to improve care of the injured patient, additional measures have been used to enhance team performance during resuscitation, including training with patient simulators and observation and feedback with the aid of video recording. Despite a defined protocol and team training, errors in evaluation and treatment persist and can contribute to adverse outcomes [46][47][48].

Errors during trauma resuscitation can be classified as errors of commission (unneeded evaluation or treatment steps), errors of omission (omission of necessary steps), scheduling errors (steps out of sequence), and procedure errors (performance of less effective steps), with errors of omission having the greatest impact on outcome [48]. Although some deviations can be identified as errors directly contributing to adverse outcome, evidence from trauma resuscitation and other high-risk medical domains have shown that even minor deviations from recommended protocol can accumulate to increase the likelihood that a major error will go uncorrected and lead to an adverse outcome [49]. Although results of this previous work have shown an association between deviations and outcomes in high-risk clinical settings, identification of deviations within these processes has relied on approaches not amenable to automatic and real-time analysis, including retrospective chart review and video analysis. The availability of approaches for automatic detection of process deviations will enable large-scale analyses of complex workflows, making it a critical step for the development of clinical decision support systems that depend on real-

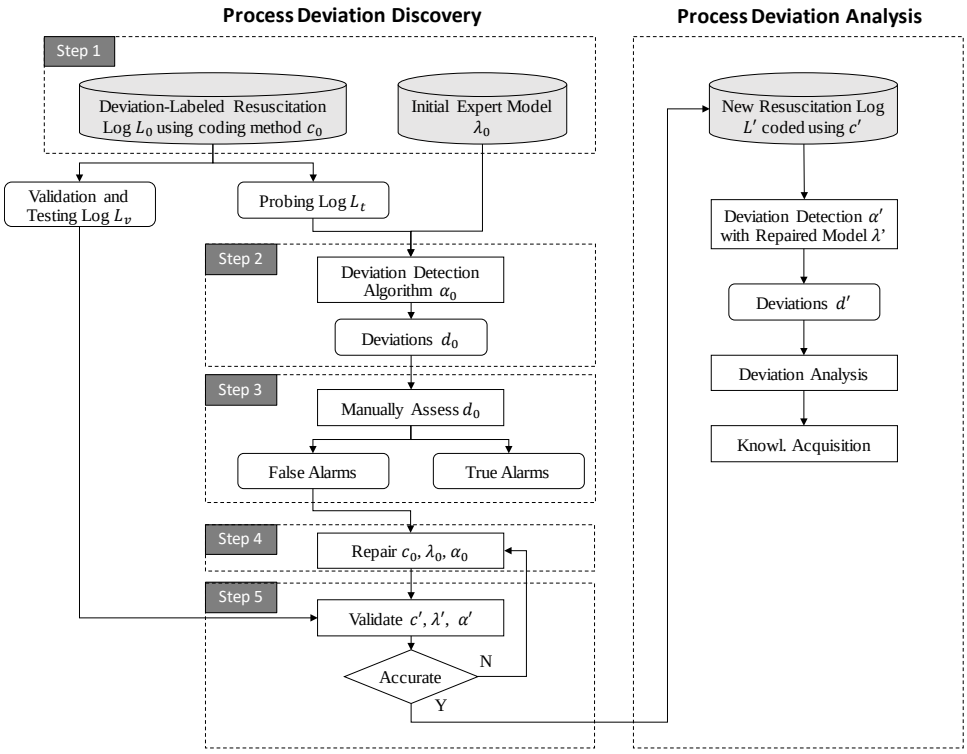


Figure 4.1. Knowledge-and-data driven process deviation discovery and analysis framework.

time data. Previous work has used this approach to discover how healthcare professionals collaborate [50] or to assess the relationships between process deviations and patient outcomes [51], but have not to identify individual deviations within the process.

In this chapter, we present an approach for automatic detection of errors and process deviations in trauma resuscitation. Building on previous process mining techniques [18], we developed an iterative, five-step approach and system for detecting deviations (Figure 4.1) that includes: (1) design of a knowledge-driven workflow model of the process and collection and coding of activity traces reflecting these model components to produce the ground truth data; (2) development of a deviation detection algorithm that uses the model and activity traces as input and generates process deviations and their locations in the process as outputs; (3) manual evaluation of the detected deviations and their classification as either acceptable process deviations (false alarms) or harmful deviations (true alarms); (4) repair of the system (i.e., model, algorithm, and coding of activity traces) using the evaluation results; and (5) validation the repaired system. The third and fourth steps are

needed to reduce the number of false alarms by resolving any discrepancies between the workflow model ("work as imagined") and actual practice ("work as done"). The fifth step is used to quantify discrepancies. Because evaluating and repairing the deviation detection system is time-intensive and laborious, we set a conservative threshold of an error rate of less than 5% for terminating the iterative evaluation and repair process. Similar to many AI-support models and systems (e.g., spam email filters, credit card fraud detection systems, and product recommender systems), the initial learning and tuning of the models require human intervention. Model-or-system developers need to pre-process the data, label the data, select or build machine learning models and tune the parameters. After the systems are built, these AI-support systems can run automatically on the new data. Our system was similarly built with human intervention, but can now automatically detect process deviations.

Our work makes two contributions to the process mining and modeling literature, namely an approach: (1) for automatic process deviation detection in complex medical processes and its implementation in the trauma resuscitation process; (2) to assist knowledge-based process modeling that quantifies and reduces the mismatches between "work as done" (actual practice) and "work as imagined" (model).

4.2 Related Work

Deviation detection methods used in process mining can be classified as data- or knowledge-driven. Data-driven methods rely on process models discovered from the data, while knowledge-driven methods rely on process models developed by domain experts. Data-driven detection methods start by discovering an "average workflow" representation, such as an average activity trace [52], a data-driven model [53], or frequently occurring patterns [30][54][55]. Traces of individual activities are then compared to the average workflow to determine the number of deviations from average. The average workflow, however, is strictly sequential and does not account for concurrent activities and repetitive behaviors. Knowledge-driven methods identify deviations by comparing a set of activities to a workflow model designed by domain experts [41][56][57] or to rules specified by experts [22][58][59]. Rule-based models, however, are designed for loosely structured processes, limiting their applicability for structured clinical processes. Specifying all

possible constraints for rule-based processes requires more effort than creating a knowledge-driven workflow model [58].

Current approaches for detecting process deviations in trauma resuscitation rely on manual review [48][60][61]. An observational study of 100 adult trauma resuscitations using video review found an average of 12 errors per resuscitation, with none being error-free [48]. Errors of omission were twice as common as errors of commission and had a greater potential impact on outcome. Most errors involved the failure to record or observe information needed for decision-making—an average of seven missing items per resuscitation. Another study of 90 pediatric trauma resuscitations also used video review, finding an average of 5.9 deviations from the ATLS protocol per resuscitation [60]. A more recent video review of 39 resuscitations [61] found a total of 337 errors (8.6 ± 4.7 errors per resuscitation), 135 of which were errors of omissions, 106 were errors of commission and 96 were errors of selection. Although effective in identifying and analyzing process errors, manual review using video recordings is labor intensive, subjective, and difficult to replicate.

Automatic process deviation detection has been used in medical and non-medical domains. Bouarfa and Dankelman [52] used an approach of finding the mismatches between an activity trace and a workflow model derived from 26 activity traces to detect process deviations during laparoscopic cholecystectomy. Compared to trauma resuscitation, which includes more than 120 activity types, the coded activities associated with laparoscopic cholecystectomy included only eight activity types. In addition, only 26 activity traces were used to derive an “average workflow,” potentially resulting in a biased representation of the process. Similarly, Lu et al. [54] detected deviations in a business process by identifying frequently common and uncommon behaviors from a model derived from actual behavior traces. In contrast, Christov et al. [55] detected process deviations during chemotherapy and blood transfusion by first creating knowledge-driven workflow models for these processes. Medical experts developed the chemotherapy workflow model, and the blood transfusion workflow model was developed using a standard blood transfusion checklist. Although this study used knowledge-driven workflow models, it only simulated the process error detection by using synthetic activity traces and inserting artificial process errors. Swinnen et al. [57] detected process deviations by first discovering

process rules using association rule mining. A domain expert then reviewed these rules for accuracy. The authors included more than 250 rules, but their approach was evaluated using only seven activity types, limiting the application of this approach.

Data-driven deviation detection approaches use historical data to produce workflow models. Although these models may better represent the observed data, the process of modeling is affected by the amount and quality of data. The training data may also include erroneous activities, which cannot be recognized by the algorithm alone. In contrast, knowledge-driven methods are not affected by the amount or quality of data, but instead represent the order and types of activities that need to be performed. Although domain experts can build simple workflow models, creating detailed models for complex medical processes like trauma resuscitation can be challenging. In addition, previous research on process deviation detection has focused on detecting deviations but has not addressed the impact of those deviations. Our approach combines both knowledge- and data-driven modeling, allowing us to build a more accurate model of a complex medical process by reconciling the differences in models created by medical experts and extracted from data.

4.3 Terms and Definitions

A process is defined as a series of actions or activities to achieve a goal. The process data is stored in an activity log $\boldsymbol{L} = \{c_1, \dots, c_n\}$ where each element represents one process case. A process case $c_i = \{id_i, \boldsymbol{T}_i\}$ is indexed with a unique case ID and consists of the activity trace $\boldsymbol{T}_i$. An activity trace is represented as $\boldsymbol{T}_i = [a_1^{(i)}, \dots, a_k^{(i)}]^T$, where $a_j^{(i)}$ is the *j-th* activity (i.e., a well-defined step in the process) in the activity trace $\boldsymbol{T}_i$ sorted by activity start time, and *k* is the trace length (i.e., number of performed activities). In our case, trauma resuscitation is treated as a process. A series of activities (e.g., airway assessment, chest auscultation) are performed by the trauma team to stabilize and treat injured patients. A process case corresponds to the record for each trauma resuscitation.

A workflow model $\boldsymbol{\lambda}$ represents a set of phases, steps, and activities performed during a process and their dependencies. A process is defined as a series of activities to achieve a particular goal, and a phase is defined as a sub-process that consists of steps needed to achieve a sub-goal of the process (e.g., airway management in the primary survey). A

workflow model can be treated as a combination of several phases. Each phase can be considered as a sub-model within the complete model. Each phase can also have sub-phases ("steps") of more granular goals. The workflow model can therefore be abstracted as a hierarchical model $\boldsymbol{\lambda}^H$. A phase (or sub-model) $\boldsymbol{\lambda}_i^{(\ell,m)} \in \boldsymbol{\lambda}^H$ indicates the *i-th* model at the ℓ*-th* level. m indicates the index of the parent model at the (ℓ–1)*-th* level and is used to represent the connection between the parent model (node) and children models. $m = null$ if the parent model does not exist. A data dictionary $\mathcal{D}$ was created to map each activity type to their respective process phase.

A fitting activity (i.e., an activity performed in conformance with the given workflow model) is indicated as *f*. A process deviation *d* is a performed activity that is not conforming to the model $\boldsymbol{\lambda}$ and can be classified as either an error of omission, error of commission or scheduling error. An error of omission d^o is a skipped-but-necessary activity (e.g., failure to assess pupils). An error of commission d^c is an unnecessary (e.g., intubation without indication) or repeated activity (e.g., several abdominal assessments). An error of scheduling is an activity performed either before or after its predetermined order (e.g., confirming that the patient's airway is secured before assessing mental status). A scheduling error $d^s = \{d^{oo}, d^{o\ell}\}$ consists of: (1) an out-of-sequence activity d^{oo} that violates the process procedural order, and (2) an original-location marker $d^{o\ell}$ that indicates the correct location in the activity trace where d^{oo} should have occurred.

We used three different fitness metrics to quantify the degree of agreement between the target process element (e.g., an activity, process phase or the entire process) and the given model, with values ranging from 0 to 1. These metrics were used in our analyses of process deviations and for evaluating our deviation detection approach. Activity fitness F_a measures the degree to which a particular activity type *a* deviates from the given model (Eq.4.1). Phase fitness F_p measures the percentage of non-conforming activities within a process (sub-)phase (Eq.4.1). Model fitness F_λ describes how well a model $\boldsymbol{\lambda}$ represents the observed activity traces (Eq.4.1). The generic fitness measure is defined as:

$$Fitness = 1 - \frac{|\boldsymbol{S}_d|}{|\boldsymbol{S}_d| + |\boldsymbol{S}_f|} = \frac{|\boldsymbol{S}_f|}{|\boldsymbol{S}_d| + |\boldsymbol{S}_f|} \tag{4.1}$$

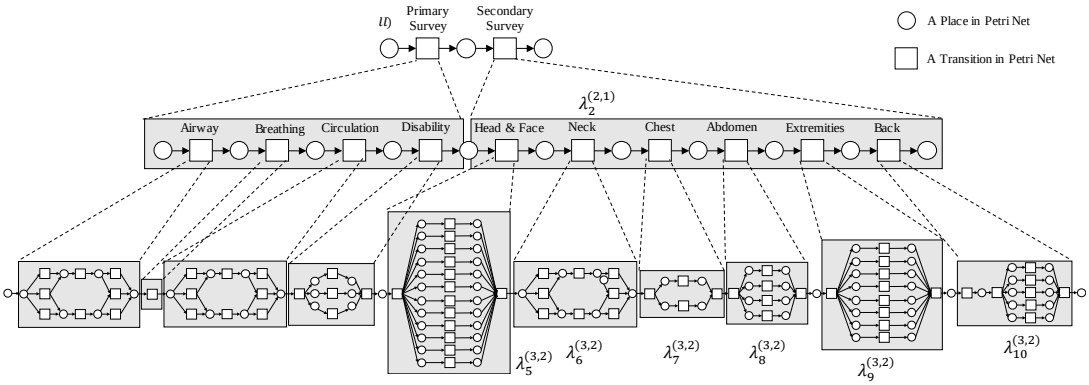


Figure 4.2. Hierarchical model of the trauma resuscitation workflow.

where $\boldsymbol{S}$ is the set of target process elements (e.g., in activity fitness F_a, $\boldsymbol{S}$ is the set of fitting and deviant instances of activity type a), $\boldsymbol{S}_d = \{x | x \in \boldsymbol{S} \text{ and } typeof(x) = d\}$ is the set of deviations and $\boldsymbol{S}_f = \{x | x \in \boldsymbol{S} \text{ and } typeof(x) = f\}$ is the set of fitting activities.

4.4 Deviation Detection Approach and System Description

To automatically identify deviations within the trauma resuscitation process, we first developed a representation of a typical process execution (Figure 4.1). We tested extracting workflow models automatically using workflow mining algorithms, but the discovered models are "spaghetti-like" and difficult to interpret [20]. For this reason, medical experts on our research team developed an initial trauma resuscitation workflow model. We then collected video recordings of 95 trauma resuscitations at a level 1 pediatric trauma center at the Children's National Medical Center (CNMC) and derived activity traces for each. We next applied an algorithm to evaluate the conformance between the workflow model and the resuscitation activity traces. Because the workflow model was incomplete, a fraction of the detected deviations represented false alarms. We then iteratively repaired the initial model using resuscitation cases that were manually annotated with true deviations. Using this approach, we developed an automatic deviation detection system that can be applied to new trauma resuscitation cases. This study was approved by the Institutional Review Board at the Children's National Medical Center.

4.4.1 Knowledge-Driven Model of the Trauma Resuscitation Workflow

We limited the scope of our model to assessment activities because diagnosis and treatment during trauma resuscitation depend on the information that has been gathered through these assessments. We built our model to describe a typical sequence of activities that constitute the complete assessment during trauma resuscitation. The model was iteratively revised until the medical experts reached a consensus about which activities to include and in what order.

We built our workflow model $\boldsymbol{\lambda}^{H}$ as a three-level hierarchical structure based on medical knowledge (Figure 4.2). The top level (Lvl 1) includes the root node, a single model $\boldsymbol{\lambda}_1^{(1,1)}$, which is composed of two key phases of the resuscitation process: primary survey $\boldsymbol{\lambda}_1^{(2,1)}$ and secondary survey $\boldsymbol{\lambda}_2^{(2,2)}$ at the second level (Lvl 2). Sub-phases (steps) at the bottom level (Lvl 3) compose the process phases. The primary survey consists of *airway* $\boldsymbol{\lambda}_1^{(3,1)}$ (assessing airway patency), *breathing* $\boldsymbol{\lambda}_2^{(3,1)}$ (assessing breath sounds and adequate oxygenation), *circulation* $\boldsymbol{\lambda}_3^{(3,1)}$ (assessing extremity pulses and managing blood loss), and *disability* $\boldsymbol{\lambda}_4^{(3,1)}$ (assessing neurological status). The secondary survey consists of a multi-step, head-to-toe physical examination of the patient's body ($\left\{\boldsymbol{\lambda}_5^{(3,2)}, \ldots, \boldsymbol{\lambda}_{10}^{(3,2)}\right\}$,). The bottom-level steps in Lvl 3 are the most granular components and include assessment activities that provide diagnostic information and control activities that represent conditional treatments based on the assessment outcome. For detecting deviations, we omitted control activities from the model for two reasons. First, these activities are conditional, making their occurrences often rare and challenging to study. Second, detecting deviations associated with control activities requires considering findings from the assessment activities and other context attributes (e.g., patient demographics or event attributes), which is beyond the scope of this chapter. Despite this limitation, a model based on assessment activities is sufficient for detecting deviations associated with adverse outcomes because an incomplete assessment increases the likelihood of missed injuries [62] or inappropriate treatments [48]. For example, an omitted or incomplete pupil assessment could delay the recognition and management of a traumatic brain injury.

Omitting control activities has allowed us to build a model of the trauma resuscitation workflow that applies to a range of patient attributes, regardless of treatments.

We included 42 assessment activity types out of 76 that were identified and coded using video review (Figure 4.3). The omitted 34 activity types were either optional for some patients or did not require any particular order for performance. For example, we excluded five vital sign measurement activity types because these occur frequently and without a specific order in the process. Three of these five activities (placement of the cardiac leads, pulse oximeter, and blood pressure cuff) are often missing from the activity traces because of occurrence before patient arrival. When building a workflow model for deviation detection, the goal of capturing potentially harmful deviations needs to be balanced with accommodating clinically permissible deviations. Within each bottom-level step (Lvl 3 in Figure 4.2), bedside physicians can perform several assessment activity types. For example, three assessment activities can occur during the breathing assessment: chest visual inspection, airway visual inspection, and chest auscultation. Among these activities, chest auscultation provides the most complete assessment of ventilation (breathing). For this reason, we classified chest auscultation as a required activity and considered the other two optional. We included only required activities in the model and omitted optional assessment activities to avoid false-detection. Finally, our model considered only the bedside physician role because this role is responsible for performing most assessment activities.

The selected 42 activity types were ordered relative to each another based on the ATLS guidelines. Assessment activities in the bottom-level (Lvl 3) process phases of the primary survey were represented in parallel, meaning that their order is flexible. The order of sub-phases (steps) for the secondary survey is defined less rigorously by the ATLS guidelines. For this reason, we organized these activities within the model based on distinct body regions that reflect a typical head-to-toe physical exam. The resulting workflow model of the trauma resuscitation process allowed us to detect potentially high-risk deviations that could negatively impact patient outcomes.

4.4.2 Trauma Resuscitation Activity Traces

We collected resuscitation activity traces over two time periods, August—December 2014 and April—October 2016. We performed selective sampling by including only the cases with patients who were admitted to the hospital following the resuscitation, when errors and error management were more likely to have an impact on patient outcome. During these two data collection periods, 289 trauma resuscitations were followed by hospital admission. Of these, 35 cases did not have a patient or caregiver consent and 159 cases were unavailable for video review because of technical issues or because members of our research team participated in the resuscitation. Our final dataset included 95 cases, of which seven were triaged as highest acuity “trauma stat attending” level cases, 46 cases were triaged as standard acuity “trauma stat” cases, and 42 cases were transfers from another hospital.

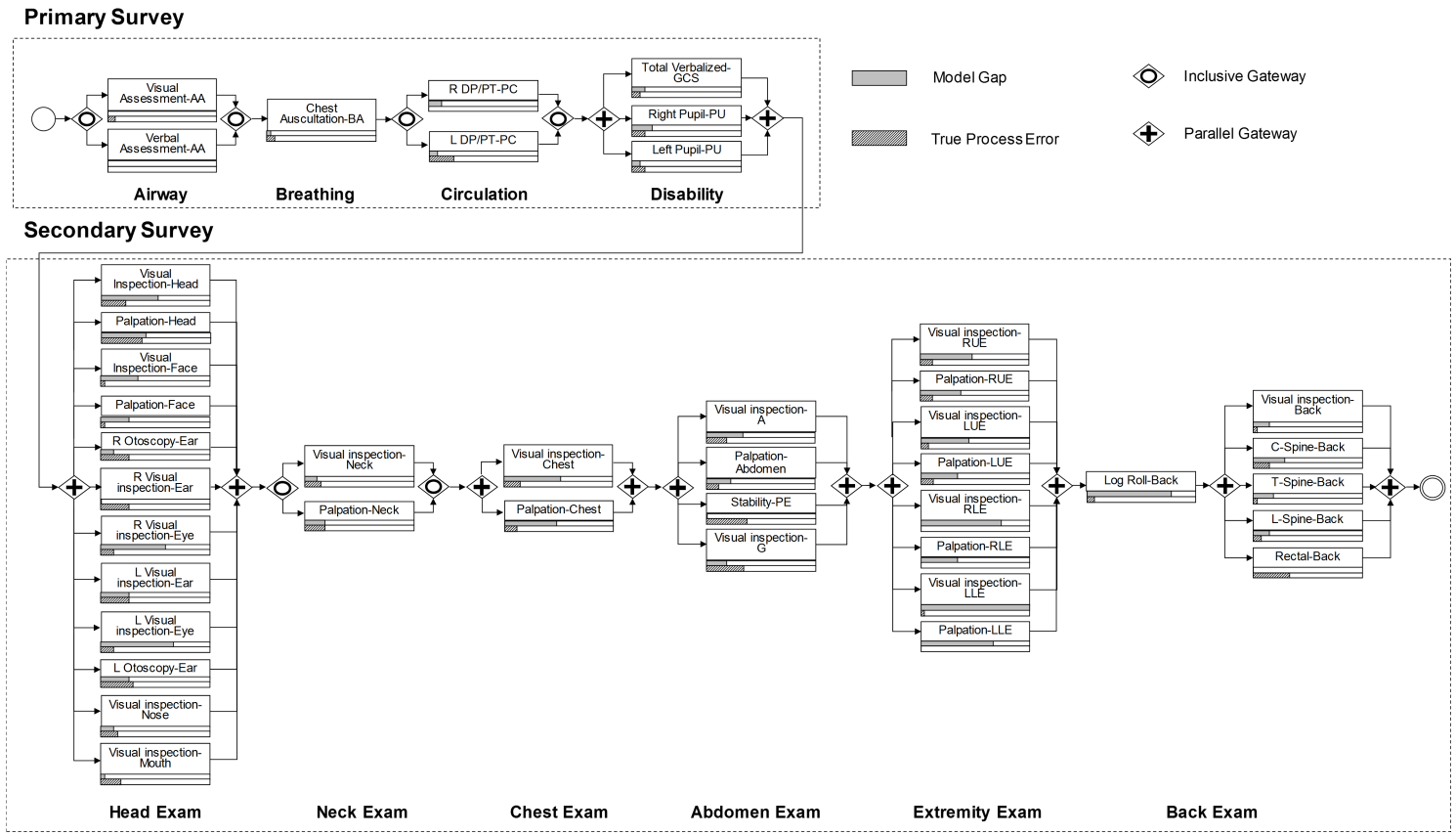


Figure 4.3. Initial knowledge-driven model represented in Business Process Model and Notation (BPMN) for 42 diagnostic activities performed by the bedside physician before repairs were made to the model. The bars under each activity box represent the occurrences of model gaps and true process errors based on 24 probing cases. The length of the bars is normalized by the largest number of occurrences (n=27) which is, the number of model gaps for activity “Visual inspection-LLE”.

Our activity traces were obtained by manual video review and manual coding. Although our research group is developing an automatic activity capture approach as part of a different project, we and others have not yet implemented a strategy that is accurate enough for obtaining activity logs required for this workflow analysis. Each activity trace contained a set of activities performed during the resuscitation, the start and end times for each activity, the role of the individual performing the activity, and a set of patient attributes. Medical experts on our research team first created a data dictionary that defines over 260 resuscitation activities and their associations with each medical phase. For example, the activity "chest auscultation" is labeled as "Chest auscultation-BA," where BA represents the process sub-phase (step) "breathing assessment." The data dictionary also defines the start and end time for each activity. For example, the start time for "chest auscultation" is defined as the time when a stethoscope is placed on the patient's chest to listen for breath sounds and the end time as the time when a stethoscope is removed from chest. Using this data dictionary, coders viewed videos and tracked team members throughout the resuscitation, documenting activity performance and attributes relevant to their performance (e.g., activity completion vs. incompletion or values obtained from the activity). Patient attributes were obtained from the hospital's trauma registry or through medical chart review. The attributes included patient age, gender, triage level, mechanism of injury, Injury Severity Score (ISS), Glasgow Coma Score (GCS), and whether or not the patient was intubated. For the purposes of detecting process deviations, we excluded patient attributes from our analysis and used only activity type, team role performing the activity, and activity start and end times.

The final activity log contained 10,851 activities of 132 types. Among these activities, 5,659 were performed for assessment of the patient. These activities were performed by bedside physicians, bedside nurses (left nurse, right nurse and charge nurse), or other team roles (e.g., surgical coordinator, respiratory therapist). A junior resident or nurse practitioner usually take the role of bedside physician, depending on provider availability. When neither provider is present, this role may be taken by another team member, including the surgical fellow or emergency medicine resident.

Algorithm 3.1. Phase-Based Conformance Checking (PCC) (illustrated in Figure 4)

Input: $\boldsymbol{T}$, $\boldsymbol{\lambda}^{H}$, $\mathcal{D}$
Output: $\boldsymbol{T}^{d}$

Step 1. Annotate activities in $\boldsymbol{T}$ with associated process-phase labels: $\boldsymbol{T}^{p} = Annotate(\boldsymbol{T}, \mathcal{D})$;

Step 2. Align $\boldsymbol{T}^{p}$ with process phases model (e.g., $\boldsymbol{\lambda}_{1}^{(1,1)}$) to find the best split for sub-traces. Let $\Phi = \{\boldsymbol{t}_1, \boldsymbol{t}_2, \ldots, \boldsymbol{t}_r\}$ denote the set of split sub-traces;

Step 3. **for** each $\boldsymbol{t}_i$ in Φ:

Step 4. $\quad \boldsymbol{t}_i^{d} = ConformanceChecking(\boldsymbol{t}_i, \boldsymbol{\lambda}_i^{(x,y)})$, where $\boldsymbol{\lambda}_i^{(x,y)}$ is the sub-phase model associated with sub-trace $\boldsymbol{t}_i$;

Step 5. $\quad$ **for** each activity a in $\boldsymbol{t}_i^{d}$:

Step 6. $\quad\quad \boldsymbol{T}^{d} = \boldsymbol{T}^{d} \cup \{a\}$;

Step 7. $\boldsymbol{T}^{d} = DetectSchedulingDeviations(\boldsymbol{T}^{d})$; /* Alg.3.2 */

Step 8. **return** $\boldsymbol{T}^{d}$;

4.4.3 Conformance Checking Algorithm

The conformance checking algorithm [24] that we used is implemented in ProM [63] and works by comparing the workflow model represented by Petri nets [21] with an activity trace of the same workflow. The outputs of the algorithm are classified into either fitting activities or process deviations of two types, errors of omission and errors of commission. The standard conformance algorithm has two limitations. First, it is computationally demanding [24]. When we applied the algorithm to our dataset using a typical desktop computer configuration (Dell, Windows 10 OS, Intel Xeon 3.7GHz CPU, 48GB RAM), processing exceeded available memory. Second, the standard conformance checking algorithm detects only two types of process deviations—errors of commission and errors of omission. Some activities, however, may be detected as omitted even when they occur in the process but are out of sequence (i.e., errors of scheduling). These activities are then detected by the algorithm as both a commission error (at the location where the activity occurs) and an omission error (at the location where the activity should have occurred).

Our approach to deviation detection includes two novel improvements to the standard algorithm that address these two limitations. First, to reduce computational complexity, we developed a phase-based conformance checking algorithm which adopts a divide-and-conquer strategy, in which a problem is split into manageable sub-problems. Our algorithm (Alg.3.1) first decomposes long activity traces into computationally-manageable sub-

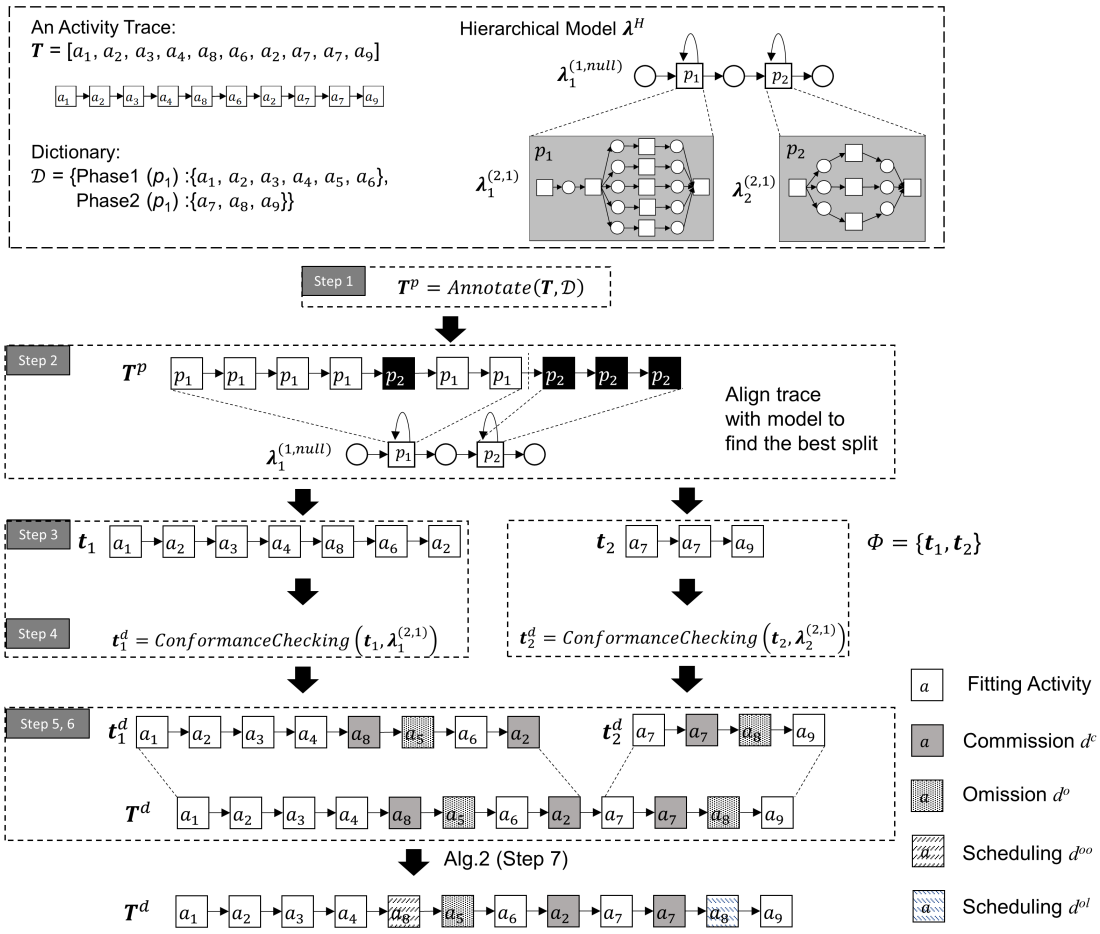


Figure 4.4. Visual representation of the phase-based conformance checking algorithm (Alg. 3.1). The steps correspond to those in Alg. 3.1.

traces. Each sub-trace is associated with a process phase. Deviations are first discovered locally in each sub-phase (step), and then reevaluated and combined globally. The algorithm takes the activity trace $\boldsymbol{T}$, hierarchical knowledge-driven model $\boldsymbol{\lambda}^H$, and activity-phase dictionary $\mathcal{D}$ as inputs and then proceeds with a three-step computation (Alg.3.1, Figure 4.5) : (1) decomposing an activity trace into sub-traces using a top-down approach (Steps 1, 2); (2) using the standard algorithm, detecting deviations in each sub-trace compared to the corresponding sub-workflow model (Steps 3, 4); and, (3) combining the results using a bottom-up approach and detecting errors of scheduling (Steps 5-8). The output is a new trace $\boldsymbol{T}^d$ that contains the original activities labeled as either fitting $\{f_i\}$ or non-conformant $\{d_i^c, d_i^o, d_i^s\}$.

Algorithm 3.2. Detection of Scheduling Deviations from Errors of Commission and Omission

Input: $\boldsymbol{T}^d$ /* Output trace from step 6 in Alg.3.1 labeled with only commissions d^c and omissions d^o */

Output: $\boldsymbol{T}^d$ /* Trace $\boldsymbol{T}$ labeled with commissions d^c, omissions d^o and scheduling deviations d^s */

Step 1. Let deviation type set $\boldsymbol{t}=\{d^c, d^o\}$
Step 2. **for** each activity $a_i \in \boldsymbol{T}^d$:
Step 3. **if** a_i.*deviationType* $\in \boldsymbol{t}$: /* if a_i is commission or omission */
Step 4. $\boldsymbol{t}$.remove(a_i.*deviationType*); /* remove the a_i's deviation type from set $\boldsymbol{t}$ */
Step 5. **for** each $a_j \in \boldsymbol{T}^d$ *s.t.* $i+1 \leq j \leq |\boldsymbol{T}^d|$: /* check the following activities */
Step 6. **if** a_j.*deviationType* $\in \boldsymbol{t}$ && a_j.*activityType*.equals(a_i.*activityType*) :
Step 7. **if** a_i.*deviationType*.equals(d^c) : /* if a_i's deviation is commission */
Step 8. a_i.*deviationType* = d^{oo}; /* assign a_i out-of-sequence marker*/
Step 9. a_j.*deviationType* = d^{ol}; /* assign a_j original-location marker*/
Step10. **else** /* if a_i's deviation is omission */
Step11. a_i.*deviationType* = d^{ol};
Step12. a_j.*deviationType* = d^{oo};
Step13. $\boldsymbol{t}$.*add*(a_i.*deviationType*); /* add a_i's deviation type back to set $\boldsymbol{t}$ */
Step14. **return** $\boldsymbol{T}^d$;

To address the second limitation, we designed a post-processing algorithm (Alg.3.2) to detect scheduling errors among the errors classified as omissions and commissions. This algorithm detects errors of scheduling by searching for an activity omission *a* at the correct location relative to the workflow model and a commission of *a* at the wrong location.

4.4.4 Workflow Model Probing, Repair and Evaluation

To assess the extent to which this initial model reflected actual practice, we performed model probing, manually marking any mismatches between the model and what was observed in practice (i.e., activity traces) as deviations. Our goal was to develop a system that would accurately identify true process errors (true alarms) while reducing the number of false alarms and misses (i.e., true process errors identified as fitting activities). To our knowledge, automated approaches to identifying true process errors from false alarms do not exist. As a basis for this assessment and repair of our initial model, we created a ground truth dataset by manually coding process deviations in 44 resuscitation cases (out of 95 total used in this study).

We divided the 44 cases into three groups: (1) 24 cases (25%) were used as probing (training) traces to first identify true process deviations and then compare how many were

successfully identified to inform system repairs; (2) 10 cases (11%) were used for validating the repaired system and determining any further repairs; and (3) 10 cases (11%) were used as testing traces to measure the performance of our deviation detection system. We selected 24 probing traces and 10 validation traces out of 95 total cases based on chronological order. To avoid sampling bias in selecting 10 testing traces, we first calculated the similarity between the remaining 61 cases using the edit (Levenshtein) distance [38]. We then clustered these 61 traces using a hierarchical clustering algorithm and Ward's method as the criterion [64]. The optimal number of clusters was two, as found by both the Silhouette score and Calinski-Harabasz index [65]. Finally, we performed random sampling from these two clusters, with the number of cases from each cluster being proportional to cluster size (i.e., number of traces in the cluster). We reviewed the recorded videos of 44 cases and manually annotated any process deviations perceived as potentially having a direct or indirect adverse effect on patient outcomes. We then compared these true process deviations to those detected by the algorithm during model probing and found that the initial workflow model did not adequately represent the resuscitation workflow. This result highlighted the notion that "work as imagined" is often different from "work as done" [66]. All of the discrepancies from the previous chapter (Chapter 3) were also captured by the new method (Chapter 4). More discrepancies were able to be discovered. The reason is that in chapter 3, the discrepancies were extracted manually by comparing the data-driven workflow model to the expert model. The data-driven workflow model needs to be simplified so that people can understand. Some details were not included in the simplified model. On the other hand, the new method (Chapter 4) can label and highlight all the activities that deviate from the expert model. By checking the commonly deviant activities and their context, we are able to uncover more insights and discrepancies between work as imagined and work as done.

Most deviations detected by the algorithm were false alarms with an adverse effect on patient outcomes being unlikely. We classified these false alarms and misses into three categories based on their causes: gaps in the model, errors in coding of activity traces, and algorithm limitations. After identifying these causes, we determined all possible repairs for the initial model and categorized each as "repairable in the model" or "irreparable in the model." We addressed the repairable model gaps by manually updating the model (e.g.,

Table 4-1. Deviations from the initial, knowledge-driven workflow model. Only activities with activity fitness scores Fa < 0.5 are listed

Activity	Sub-phase	Fitting	Omission	Commission	Scheduling	Fitness
Log Roll-BK	Back	1	23	0	0	0.04
L Visual inspection-Eye	Head & Face	4	20	1	0	0.16
R Visual inspection- Eye	Head & Face	5	19	0	0	0.21
L Visual inspection-Ear	Head & Face	8	14	1	2	0.32
L Otoscopy-Ear	Head & Face	8	13	0	3	0.33
Palpation-Neck	Neck	9	5	10	2	0.35
Visual Inspection-Neck	Neck	10	3	8	4	0.40
Visual Inspection-LLE	UE & LE	23	1	27	0	0.45
Visual Inspection-G	Abdomen	13	2	4	9	0.46
Palpation-Head	Head & Face	22	2	22	0	0.48
C-spine-Back	Back	15	4	7	5	0.48
Visual Inspection-Chest	Chest	21	0	19	3	0.49

* Abbreviations: “G: genital”, “L: left”, “LLE: left lower extremity”, “R: right”, “UE & LE: upper extremities and lower extremities”.

allowing multiple performances of activity *a*). To address irreparable model gaps, we modified our coding strategies described in our data dictionary or the algorithm for deviation detection (e.g., added post-processing steps, as described in Section 4.4.3).

4.5 Results: Deviation Detection and Analysis of Deviations

We first performed deviation detection in trauma resuscitation workflow using our system before any repairs were made to the initial workflow model, conformance checking algorithm, and coding strategies. We then repeated the deviation detection and analysis process after implementing the model, algorithm and coding improvements to evaluate the system.

4.5.1 Deviation Detection and Analysis Before System Repair

4.5.1.1 Initial Model Probing

We probed the initial workflow model by applying a standard conformance checking algorithm to 24 probing traces with a total 1,099 activities of 42 types, discovering 573

deviations, on average 23.9 deviations per case. Experts observed that 73% (n=418) of these deviations were false alarms, with only 27% (155) being true alarms. Among these false alarms, 78.5% (n=328) represented gaps in the model (e.g., acceptable repetition of visual chest inspection), 13.9% (n=58) were errors in the manual coding of activity traces (e.g., visual inspection of the patient's back was performed but not coded), and 7.7% (n=32) were due to algorithm limitations (e.g., algorithm could not clearly determine the correct sequence of performance). In addition, the system missed 13 true process deviations, including two model gaps, four coding errors, and seven algorithm limitations. Deviation detection accuracy was only 66.6% with 0.42 F_1-score (precision 0.27 and recall 0.92). This high recall and low precision results showed that the initial deviation detection system uncovered most process deviations (i.e., few misses) but also incorrectly labeled many activities as deviations (i.e., many false alarms).

The initial model probing showed that process deviations accounted for about half of the occurrences of each activity (Figure 4.3). The overall model fitness (F_λ) of 24 probing traces was 0.56. This low score showed that 44% of the activities deviated from the model. Errors of commission were more frequent than errors of omission and scheduling, especially during the *Extremities* assessment phase. Errors of omission were often observed during the *Head and Neck* assessment phase. The mean activity fitness score for all 42 activity types was 0.54 ± 0.18 (range 0 to 1). We further analyzed the 12 activities that had very low fitness scores ($F_a < 0.5$,) for potential causes of deviations (Table 4-1, Figure 4.3). For example, the conformance checking showed that the log roll activity (a maneuver used to move the patient for back assessment, coded as "log roll-back") fit with initial model only once, and was omitted in 23 out of 24 cases. Using video review, however, we observed that the log roll was performed in 20 out of 23 cases. The reason for this discrepancy was that this activity was not labeled for analysis in the 23 cases because of being performed by bedside nurses and not by a bedside physician, as assumed by the algorithm. Our initial model did not include the activities of other medical roles because their workflows are less structured and depend on patient conditions. We therefore classified this deviation as a model gap.

We observed both true process errors and model gaps for most activities (Figure 4.3), which led to our assumption that the occurrence of process errors is associated with the

gaps in our model. We hypothesized that modeling a complex workflow such as trauma resuscitation may be more challenging at places where medical teams make more errors. To test this hypothesis, we calculated the correlation between the number of model gaps and the number of true process errors associated with each activity, resulting in a Pearson correlation efficient r = -0.30 (p-value = 0.06) and Spearman correlation efficient r = -0.23 (p-value = 0.14). These results showed no significant correlation between the occurrence of process errors and model gaps, suggesting that the challenges in modeling a complex workflow are not associated with the occurrence of deviations. The true causes of false alarms are often found in the approaches to modeling and in the actual process of modeling.

4.5.1.2 Analysis of Process Deviations Detected as False Alarms

We identified three causes of false alarms: gaps in the model, errors due to manual coding of the activity traces, and errors attributed to algorithm limitations.

False Alarms due to Gaps in the Model

We identified three types of model gaps and repaired them as needed. First, when creating the initial knowledge-driven model (λ_0), medical experts relied on their knowledge and expertise, but still found it challenging to specify an exhaustive list of steps for a complex workflow such as trauma resuscitation. Expert review of deviations detected in the 24 probing cases allowed us to identify the required areas of model flexibility that better represented the process. Our model now includes repeated occurrences of activities in the same body region. Second, although the initial model focused on the work of bedside physicians, other medical roles (e.g., bedside nurses) performed some activities when situations allowed. For example, bedside nurses often assessed the left pupil (“left otoscopy-ear”) to avoid the need for the bedside physician to move to the other side of the bed. The model was therefore repaired to account for assessment activities performed by all providers to avoid labeling those activities as omissions. Third, some activities in the model are marked as optional if performed in advance. For instance, visual inspection of eyes in the secondary survey becomes optional if the pupils were previously checked in the primary survey. This conditional logic is difficult to express using Petri nets. Although two parallel branches can be created to represent these conditions, this approach is impractical because it increases the complexity of the model and deviation detection. We addressed

this problem by considering irreparable model gaps as rule-based constraints associated with the model at the post-processing step.

False Alarms due to Errors in Manual Coding of the Activity Traces

The manual data coding of activity traces was not only labor-intensive, but also prone to coding errors. The initial system returned 58 false alarms (2.4/case) related to coding errors. Most coding errors were more than just clerical errors. We analyzed frequently miscoded activities and found that most of the errors were due to different interpretations between coders. For example, using video review only, it was challenging to determine the exact body parts being observed by providers during visual body inspection. Other disagreements were due to both ambiguous interpretations of anatomical regions (e.g., how to code for a person's flank that can be interpreted as the abdomen or back) and subjective interpretations (e.g., whether the provider looked at or touched a body part). We used two strategies to improve our ground-truth data coding. First, we revised the data dictionary to provide clearer definitions of easily miscoded activities. Second, to reduce individual coder bias and disagreements between different coders, different coders analyzed the same sample cases (10% of all cases) to assess inter-rater reliability. We observed an excellent inter-rater reliability— Pearson correlation coefficient 0.99 for time-to-activity variables and Kappa statistic 0.89 for binary variables. Although these strategies reduced the number of coding errors, this class of errors cannot be completely eliminated because of subjectivity associated with manual coding.

False Alarms Attributed to the Algorithm Limitations

Our deviation detection algorithm cannot avoid all errors, as processes are inherently ambiguous. For example, in a five-activity trace $\boldsymbol{T}=\{a_4, a_5, a_1, a_2, a_3\}$, where activities a_1, a_2 and a_3 were performed late based on the expert opinion, the algorithm may identify activities a_4 and a_5 as deviations (i.e., early performance) to minimize the penalty of mismatches. In another example with a five-activity trace $\boldsymbol{T}=\{a_1, a_2, a_3, a_3, a_4\}$, the medical expert labeled the first a_3 as an erroneous performance and the second a_3 as the fitting activity, while our algorithm labeled the second a_3 as an error of commission. In both examples, our algorithm made different predictions from the ground truth. In the first example, determining whether an activity is performed early or late can be subjective

Table 4-2. Confusion matrices for 24 probing, 10 validating and 10 testing traces.

24 Probing Traces						
	Predicted as deviation			Predicted as non-deviation		
True deviation	155			13	Model 2 Coding 4 Algorithm 7	Recall: 0.92
Non-deviation	418	Model 328 Coding 58 Algorithm 32		705		
	Precision: 0.27					Accuracy: 66.6% F1-Score: 0.42

10 Validating Traces						
	Predicted as deviation			Predicted as non-deviation		
True deviation	85			15	Model 0 Coding 9 Algorithm 6	Recall: 0.85
Non-deviation	16	Model 1 Coding 12 Algorithm 3		526		
	Precision: 0.84					Accuracy: 95.5% F1-Score: 0.85

10 Testing Traces						
	Predicted as deviation			Predicted as non-deviation		
True deviation	139			5	Model 4 Coding 1 Algorithm 0	Recall: 0.97
Non-deviation	8	Model 0 Coding 2 Algorithm 6		522		
	Precision: 0.95					Accuracy: 98.5% F1-Score: 0.96

because it depends on the observer's reference. Although humans tend to label activities as late, computers may choose to label these activities as early to minimize the global penalty. In the second example, the erroneous performance of an activity represented additional information that was not coded in the original ground truth data and was therefore omitted from the deviation detection algorithm. We addressed most of the algorithm limitations by performing additional model modification and adding a post-processing step (Alg.3.2).

4.5.2 Deviation Detection After System Repair: Validation and Testing

After completing the system repairs (i.e., addressing the model gaps and improving the coding strategies and deviation detection algorithm), we used ten cases with a total of 621 activities to validate the system. The repaired system detected 541 conformant activities and 101 deviations (10.1 per case). Among these 101 deviations, 56.4% (57) were errors of commission, 20.8% (21) were errors of omission and 22.8% (23) were scheduling errors. Using manual review, we found that 85 detected deviations were correctly labeled, representing "true alarms," and 16 deviations were identified as false alarms (Table 4-2) , yielding a precision of 0.84. Among these 16 false alarms, 12 were due to coding errors, three were due to the algorithm limitations and only one was attributed to the gaps in the repaired model. In addition, the system missed 15 deviations, including nine errors due to coding issues and six due to algorithm limitations, yielding a recall of 0.85. The repaired system had 95.2% accuracy on the ten validation traces with an F_1-score of 0.85. This result represented a significant improvement over the probing results. Although some coding and algorithm errors are difficult to avoid, model gaps continued to occur because of the need to balance the accuracy and complexity of the model. Introducing more patient-condition dependent branches into the model might produce a more accurate model, but its generality would decrease while increasing complexity.

Finally, our experimental results on ten testing cases with a total of 641 activities showed a F_1-score of 0.96 and an accuracy of 98.5%. Compared to the validation results, the higher accuracy was due to the increased number of true alarms and decreased number of false alarms and misses. We also observed fewer coding errors among the false alarms and misses, which contributed to the increased system accuracy. We did not observe significant changes in the number of model gaps and algorithm issues.

4.5.3 Analysis of Process Deviations Detected with the Repaired System

We applied the repaired system to all 95 cases that included 5,659 activities, finding 4,893 fitting activities, 294 errors of omission, 538 errors of commission, and 228 scheduling errors (an average of 11.2 deviations per resuscitation case). These 95 cases included the 24 probing cases, 10 validation cases and 10 testing cases from the previous stages of

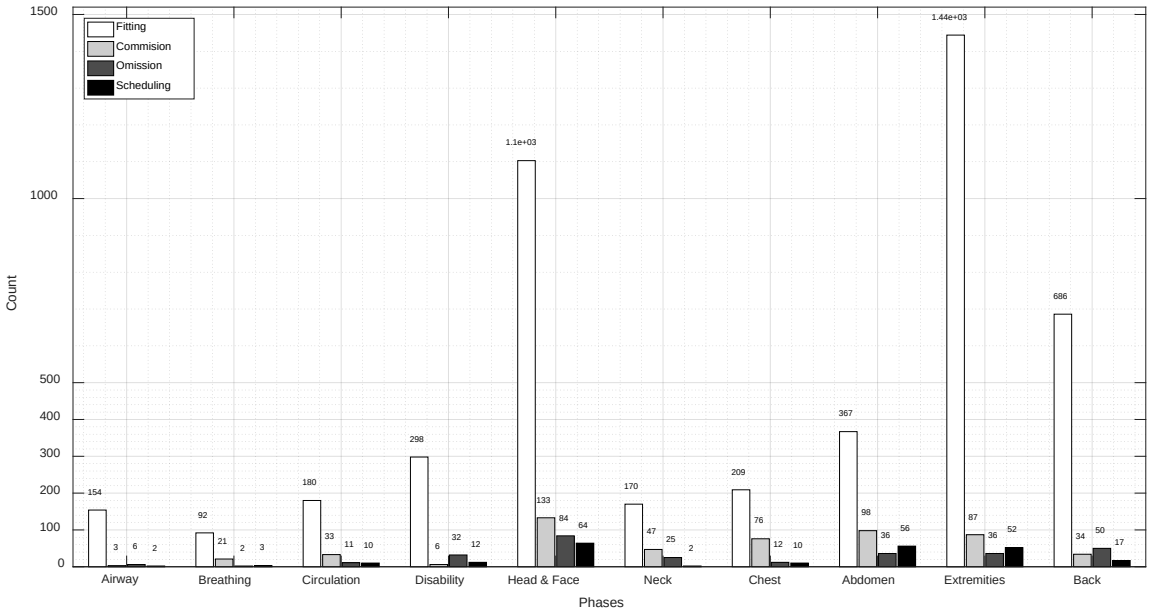


Figure 4.5. Distribution of detected deviations during process sub-phases (steps) for all 95 cases after the system repair.

system development. While re-using probing (training) cases for the evaluation of predictive models is not appropriate, reusing data for the explanatory purposes is an acceptable practice [42]. The probing cases were initially used to detect any gaps in our knowledge-driven workflow model. These gaps were then manually analyzed and the model was repaired where appropriate. The repaired model may be biased because repairs were made based on the probing cases, possibly missing the gaps that would be exposed in the unseen cases. This issue can only be addressed by obtaining the ground truth data (i.e., manual data coding) for all 95 cases, a task that is labor-intensive and impractical.

The overall model fitness (F_λ) of 95 traces was 0.82. Activity fitness scores (F_a) ranged from 0.49 to 1.00 (Table 4-3). The activity "visual inspection-genital" had the lowest fitness score of 0.49 due to errors of omission observed in 22 cases and scheduling errors in 25 cases. The overall mean activity fitness score was 0.80 ± 0.12 (Table 4-3). We found an uneven distribution of detected deviations across process sub-phases (Figure 4.5). Although the *Airway* phase in the primary survey had only 11 deviations (F_p = 0.93), the *Head & Face* phase in the secondary survey had 281 deviations, including 133 commissions, 84 omissions and 64 scheduling errors (F_p = 0.80). The *Chest* and *Abdomen*

phases in the secondary survey had fewer total deviations (98 and 190 deviations, respectively, Figure 4.5), but also ower phase-fitness scores (0.68 and 0.66, respectively). A possible explanation for this result is that the secondary survey allows for more flexibility in the order of activity performance.

Our analysis and expert video review of the 1,060 detected deviations showed that errors of commission were mostly due to reassessments. These errors were most common in the *Chest* and *Abdomen* phases of the secondary survey (Figure 4.5). Seventy-six errors of commission occurred in the *Chest* phase, leading to a decrease in the phase fitness score, $\Delta F_p = -0.23$, from 0.91 to 0.68. Ninety-eight errors of commission occurred in *Abdomen* phase, $\Delta F_p = -0.14$, from 0.8 to 0.66. Several reasons may explain the need for reassessments. First, the bedside physician and other medical team members perform rapid evaluation and often need to confirm the initial findings by reassessing the patient. Second, reassessment may also be needed when the patient presents with new symptoms. Finally, examination results are reported verbally and can often be missed or misheard, requiring team members to obtain new findings by reassessing the patient.

Errors of omission were most common for the rectal and genitalia assessments (Table 4-3), especially when an injury to these body areas was unlikely. The pelvis stability exam and Glasgow Coma Score (GCS) verbalization were omitted in 14 ($\Delta F_a = -0.11$) and 13 ($\Delta F_a = -0.13$) in activity fitness) cases, respectively (Table 4-3). Ten of the cases that lacked the pelvis stability exam involved patients transferred from another hospital. For transfer patients, the trauma team usually follows an expedited protocol that focuses on the most critical injuries for which the patient was transferred. In addition, a junior resident was the bedside physician in eleven of these cases, suggesting that some omissions could also be due to inexperience. Video review did not show any other reasons that could explain why this examination was skipped. Although our ground truth coding showed that GCS exams were omitted in 13 cases, subsequent video review found that the teams performed this exam in two cases, but did not verbally reported the findings, which classified these activities incomplete. Among the remaining 11 cases in which GCS was not calculated, six involved transfer patients, three of which arrived without prior notification, and the other five cases involved three patients who arrived with prior notification and two with no prior notification. Finally, nine of the 13 cases involved patients with head injuries, suggesting

that these omissions were not permissible and could have led to adverse patient outcomes. Three reasons may explain why the GCS exam was omitted. First, all of the patients were alert, awake, and oriented, potentially making it obvious that the GCS was normal. In several videos, the patients were visibly upset and uncooperative with the provider. When this happened, the team leader often instructed the bedside physician to move on with the exam, skipping the GCS calculation.

4.6 Discussion and Conclusions

In this chapter, we introduced an approach and a system for automatic detection of process deviations in trauma resuscitation. This approach provides a method for identifying repeated, omitted and out-of-sequence activities that can be included in the design of decision support systems for complex medical processes. Our results have implications for three areas of research: (a) workflow modeling, (b) conformance checking, and (c) process deviation analysis in complex medical workflows.

To identify deviations within the resuscitation workflow in an efficient and unbiased way, we first developed a representation of the typical execution of the process using a top-down, knowledge-driven workflow modeling approach. Clinical workflow in the resuscitation setting is highly dynamic, context-sensitive, and knowledge-intensive. Although essential assessment activities are needed for every patient, the team must also select additional assessment and treatment activities based on contextual information relevant to a specific patient (e.g., fluid administration in the presence of signs of hemorrhagic shock). This complex and dynamic nature of the resuscitation workflow makes its modeling a challenge. In our study, we treated ATLS as a set of guidelines that assists with the management of injured patients rather than a protocol that precisely defines this management. Variations in the clinical setting may require a deviation from these guidelines to ensure the best care. In viewing the ATLS as a guideline, we allow the team to make decisions based on their experience and expertise. This flexibility allows adjustment to complex and dynamic circumstances.

Our initial model underwent several revisions until the medical experts reached a consensus about which medical activities to include and in which order. Our preliminary analysis with 24 trauma resuscitation cases using conformance checking, however, found

discrepancies between the model and actual practice, showing that our initial model could not fully represent the resuscitation process. These results were significant because they identified false alarms that were triggered by an incomplete model, allowing us to effectively repair the model. The repairs were also iterative and informed by detailed analyses of both true and false deviations. During this study, we made several major model repairs, most of which involved manual updates to the model to include permissible deviations, allow for repeated occurrences of activities, or recognize more than one role as responsible for activity performance. Other model repairs were related to algorithm failures or errors in manual coding of the activity traces, and were addressed accordingly. These results highlight the importance of building accurate and complete workflow models because discrepancies between the model ("work as imagined") and actual practice ("work as done") can also affect the accuracy and effectiveness of the automatic deviation detection. Our results also show that models can serve to discover associations between process deviations and the occurrence of major errors that are associated with a higher likelihood of adverse outcomes.

Our study has three major limitations. First, we focused on assessment activities performed by the bedside physician because these activities are required for every patient. We did not include any treatment or control activities because these are conditional (i.e., performed based on the outcome of assessment activities) or sparse, requiring a more complex model than for assessment activities. Our future work will focus on modeling control activities and using those models for further deviation detection. Second, we only considered three types of deviations related to the activity sequence: errors of omission, errors of commission, and scheduling errors. Other types of deviations exist but cannot be detected with our current system. For example, we did not analyze deviations that violate time constraints (e.g., blood pressure and heart rate need to be measured within the initial two minutes), deviations in decisions (e.g., whether the patient should be intubated or requires intravenous fluid), or deviations related to activity performance (e.g., incomplete evaluation of patient's airway or evaluation performed using an incorrect instrument). A more elaborate workflow model, or even a multi-model system, that represents these additional attributes is needed to detect these deviations. Finally, in this work, we focused

on detecting process deviations and did not analyze the correlation between context attributes (e.g., patient demographics) and process deviations.

Existing conformance checking algorithms fail for workflows with multiple activities and concurrency because of computational complexity. To address this limitation, we developed a novel phase-based conformance checking algorithm and our results on testing cases showed a detection accuracy of 98.5%. Our approach to automatic deviation detection using activity traces and conformance checking also allowed us to characterize the detected deviations as either tolerable variability or harmful errors. Given the overlap of resuscitation activities and steps with those in other critical care settings, our approach can be used to facilitate the retrospective analysis of critical care workflows that can be scaled to large numbers of sessions, as well as the development of novel runtime approaches for reducing mitigating the impact of human errors.

Table 4-3. Deviation detection results for 95 cases after the system repair

Activity	Fitting	Omission	Commission	Scheduling	Activity Fitness	Sub-phase/Step
Visual Inspection-G	52	22	7	25	0.49	Abdomen
L Visual inspection-Ear	29	8	10	0	0.62	Head & Face
L Otoscopy-Ear	63	9	3	26	0.62	Head & Face
Visual Inspection-Neck	86	19	33	0	0.62	Neck
R Visual inspection-Eye	19	4	7	0	0.63	Head & Face
R Visual inspection-Ear	30	7	10	0	0.64	Head & Face
L Visual inspection-Eye	18	4	6	0	0.64	Head & Face
Visual Inspection-Chest	108	3	45	6	0.67	Chest
Rectal-Back	72	30	3	1	0.68	Back
Visual Inspection-A	121	0	49	8	0.68	Abdomen
Palpation-Chest	101	9	31	4	0.70	Chest
R Otoscopy-Ear	75	8	4	20	0.70	Head & Face
Stability-PE	75	14	3	14	0.71	Abdomen
Palpation-A	119	0	39	9	0.71	Abdomen
L DP/PT-PC	85	11	13	9	0.72	Circulation
Visual inspection-Mouth	83	13	7	7	0.75	Head & Face
C-spine-Back	99	11	6	13	0.77	Back
Visual Inspection-RUE	139	6	23	14	0.76	UE & LE
Chest auscultation-BA	92	2	21	3	0.78	Breath
Visual inspection-Face	147	6	32	3	0.78	Head & Face
Palpation-Face	129	6	27	3	0.78	Head & Face
Palpation-RUE	147	6	21	13	0.79	UE & LE
Palpation-Neck	84	6	14	2	0.79	Neck
Visual inspection-Nose	85	12	2	5	0.82	Head & Face
Visual Inspection-LUE	136	9	9	12	0.82	UE & LE
Palpation-LUE	130	10	5	13	0.82	UE & LE
R DP/PT-PC	95	0	20	1	0.82	Circulation
Total Verbalized-GCS	81	13	2	2	0.83	Disability
Log Roll-Back	89	4	8	2	0.86	Back
Left pupil-PU	103	9	2	5	0.87	Disability
Right pupil-PU	114	10	2	5	0.87	Disability
Visual Assessment-AA	88	6	2	1	0.91	Airway
Palpation-Head	201	4	13	0	0.92	Head & Face
Visual Inspection-Back	137	1	10	0	0.93	Back
Visual inspection-Head	224	3	12	0	0.94	Head & Face
Visual Inspection-LLE	228	2	9	0	0.95	UE & LE
T-spine-Back	155	2	4	1	0.96	Back
L-spine-Back	134	2	3	0	0.96	Back
Visual Inspection-RLE	236	1	8	0	0.96	UE & LE
Palpation-LLE	218	1	7	0	0.96	UE & LE
Verbal Assessment-AA	66	0	1	1	0.97	Airway
Palpation-RLE	210	1	5	0	0.97	UE & LE
Patient arrival	95	0	0	0	1.00	N/A
Patient departure	95	0	0	0	1.00	N/A
Sum	4893	294	538	228	0.80	0

* Abbreviations: “A: abdomen”, “AA: airway assessment”, “BA: breathing assessment”, “DP/PT-PC: dorsalis pedis/posterior tibial pulse”, “G: genital”, “GCS: Glasgow Coma Scale”, “L: left”, “LLE: left lower extremity”, “LUE: left upper extremity”, “PE: pelvic”, “PU: pupil”, “R: right”, “RLE: right lower extremity”, “RUE: right upper extremity”, “UE & LE: upper extremities and lower extremities”.

Chapter 5
Process Mining the Trauma Resuscitation Patient Cohorts

This chapter on Process Mining the Trauma Resuscitation Patient Cohorts is based on our paper. In this study, we present a framework for analyzing associations between patient cohorts and the trauma resuscitation procedures their patients received. Our framework works by quantifying associations between discovered patient cohorts and treatment patterns. We evaluated our framework on a trauma resuscitation dataset collected in a level 1 trauma center. Our experimental results show that using weights learned by our algorithm improves measurements of patient similarity. Four patient cohorts were then found via clustering, and statistically significant resuscitation patterns were discovered using process mining techniques. Though only tested on the trauma resuscitation process, our framework can be generalized to analyze other medical processes.

5.1 Introduction

In medical research, patient cohort analysis is widely used to make clinical discoveries [67] [68][69]. A patient cohort is defined as a group of patients who share similar context attributes. Taking trauma resuscitation as an example, the trauma patients of a same cohort are sharing similar attributes like demographics (e.g., age, gender, ethnicity, insurance and medical history), injury information (e.g., injury type, injury severity and injury area), and trauma attributes (e.g., day vs. night shift, trauma activation level and pre-arrival notification). In traditional pipeline of patient cohort analysis, medical analysts [68][70] study patient cohorts by defining the cohorts according to the targeted attributes defined by medical experts. Other context attributes were considered as confounding and ignored in the study. The limitation of doing so is that their studies mostly reveal the expected results within the cohorts they are familiar with. The studies were very well oriented by their domain knowledge so that they were likely to miss the cohorts and findings they were not familiar with.

Process mining [18] is another analysis that has been recently applied in medical domain on medical process analysis. It has been used to discover medical process models, measure the compliance of process executions with expert models [59] and analyze medical process deviations [55]. Existing process mining research [18] however, mostly mines knowledge from an entire dataset of a process without studying the differences among the subsets of the process cases.

In this chapter, we present a framework for medical process data analytics by combining both process mining and patient cohort analysis. Our medical process data includes two parts of information: process activity logs (e.g., trauma resuscitation executions) and context attributes (e.g., patient demographics) associated with each process cases. Our framework works in three steps. First, it applies data exploration methods on patient attributes to find data-driven cohorts. Second, it discovers process patterns (e.g., treatment patterns) from activity logs using process mining techniques. Third, it tests the significance of the correlations between process patterns and patient cohorts.

We applied our framework to a real-world medical process, i.e., trauma resuscitation. Trauma resuscitation is a fast-paced process, where multidisciplinary teams need to rapidly identify and treat potentially life-threatening injuries. Analysis of the correlation between their treatment executions and patient cohorts can potentially improve their understanding of their behaviors and hopefully improve patient outcomes.

Our contributions in this study are:

- A framework for discovering and analyzing the associations between trauma patient cohorts and trauma resuscitation procedures. Our framework is easy to implement and can be used for analyzing processes with event (or activity) logs and external context attributes.
- A practical algorithm and experimental procedure to learn the weighted importance of attributes with little human intervention. Unit weights are usually assigned to attributes when calculating data similarity for clustering, as the actual significance of each attribute is unknown. In this study, we designed an experiment to very efficiently acquire medical experts' input to supplement attribute weight learning.

Table 5-1. Activity trace (a), context attributes (b), data statistics (c) and data formalization (d).

Case ID	Activity	Start Time	End Time
xx1	Pt arrival	0:00:00	0:00:01
xx1	Visual assessment-AA	0:00:45	0:00:52
xx1	Chest Auscultation-BA	0:00:55	0:00:58
xx1	R DP/PT-PC	0:01:04	0:01:05
xx1	Total Verbalized-GCS	0:01:29	0:01:30
xx1	Total Verbalized-GCS	0:01:50	0:01:51
xx1	Right pupil-PU	0:02:12	0:02:18
xx1	Left pupil-PU	0:02:19	0:02:24
xx1	Right pupil-PU	0:02:24	0:02:25
xx1	Visual inspection-H	0:02:33	0:02:34
xx1	Palpation-H	0:02:33	0:02:37

(a) Trauma resuscitation trace

Case ID	xxx1	xxx2
Age category	24-96	24-96
Sex	Male	Female
Night Shift	0	1
Weekend	0	0
Pre-arrival Notification	1	0
Trauma Activation Level	Transfer	Attending
Intubation	0	0
Glasgow Coma Score >13	1	0
Injury Type	Blunt	Penetrating
Injury Severity Score	5	12
Neck Injury Severity Score	3	5

(b) Context attributes

Properties	Stats
Num. Cases (or Patients)	123
Num. Total Activities	7154
Num. Activity Types	44
Num. External Attributes	26
Data Collection Time Period	2014.08 – 2016.12
Size of Medical Team	[7, 12]
Longest Trace (Num. Acts.)	110
Shortest Trace (Num. Acts.)	26
Avg. Num. Acts. in Traces	58.6

(c) Data statistics

ID	Ext. Attributes	Resus. Traces
$id^{(1)}$	$x_1^{(1)}, \dots, x_g^{(1)}$	$a_1^{(1)}, \dots, a_k^{(1)}$
$id^{(2)}$	$x_1^{(2)}, \dots, x_g^{(2)}$	$a_1^{(2)}, \dots, a_k^{(2)}$
⋮	⋮	⋮
$id^{(n)}$	$x_1^{(n)}, \dots, x_g^{(n)}$	$a_1^{(n)}, \dots, a_k^{(n)}$

(d) Data formalization

- An analysis of statistically significant correlations between context attributes (aggregated as patient cohorts) and discovered medical treatment patterns in a real-world dataset of 123 trauma patients.

5.2 Patient Cohort Discovery and Analysis

In this section, we described the core techniques used in attribute weight learning, patient cohort discovery, process mining and statistical analytics. We learnt the attribute weights with the goal to decide the importance of different context attributes. In this way, we could find more accurate patient cohorts through clustering algorithms. We then mined the treatment patterns within each patient cohorts and analyzed them with statistical methods.

5.2.1 Data Description and Formalization

One hundred and twenty-three trauma resuscitation videos were collected from trauma bay of Children's National Medical Center, Washington DC. The videos were reviewed jointly by a surgeon with Advanced Trauma Life Support (ATLS) [46] certification and trauma clinical nurse specialists to identify the activity traces (Table 5-1 (a)). A total of 7154 main activities of 44 types were selected in this study. Twenty-six context attributes were

collected from the trauma database or from medical chart review, including patient age, gender, trauma activation level, mechanism of injury (penetrating, blunt, burn, etc.), date and time of patient arrival, Injury Severity Score (ISS), Glasgow Coma Score (GCS), intubation status, and Abbreviated Injury Scale (AIS) (Table 5-1 (b)). The collection and use of the data for this study were approved by the Institutional Review Board at our hospital.

Here we first define the terms and notations used later. The process **log** $\boldsymbol{L} = [c^{(1)}, \dots, c^{(l)}]^T$ is a vector of elements $c^{(i)}$. Each $c^{(i)} = \{id^{(i)}, \boldsymbol{x}^{(i)}, \boldsymbol{T}^{(i)}\}$ (Table 5-1 (d)) represents a unique **case**, which is indexed with a unique case *id*, contains the activity trace $\boldsymbol{T}^{(i)}$, and has a vector $\boldsymbol{x}^{(i)}$ of context attributes. An activity **trace** is $\boldsymbol{T}^{(i)} = [a_1^{(i)}, \dots, a_m^{(i)}]^T$, where *m* total activities *a* are ordered by activity start time. Traces of different executions may have varying lengths because complex processes may contain optional, omitted, or even erroneously performed activities. **Context attributes** $\boldsymbol{x}^{(i)} = [x_1^{(i)}, \dots, x_n^{(i)}]^T$ is a vector of n recorded patient demographics (e.g., age, gender, ethnicity, insurance and medical history), injury information (e.g., injury type, injury severity and injury area), and trauma attributes (e.g., day vs. night shift, trauma activation level and pre-arrival notification).

5.2.2 Attribute Weight Learning

In our framework, the patient cohort is decided by unsupervised clustering algorithms. The clustering performance is highly associated with the attributes used. The discovered patient cohorts may be meaningless if irrelevant or unimportant attributes are used. Without any prior knowledge, the attribute weights are mostly set as unit weights (i.e., any attribute has the same weight as one). With domain knowledge available, it is possible to obtain the attribute weights by asking medical experts to provide a score (e.g., in the scale of 0 – 10) for each attribute. This approach however can be challenging in practice. We tried this method in our study by asking the medical experts in our team to decide a dictionary of attribute weights. Our medical experts later gave us the feedback that they would prefer to use unit weights rather than decide a set of subjective weights. In addition, even if the medical experts were able to provide a set of weights, it is likely that the study is again

Algorithm 5.1. Patient Attribute Weighting

Input: N random drawn sets $\boldsymbol{S}$; labels from medical expert $\boldsymbol{P}_{Dr}$
Output: Learnt Attribute weights $\boldsymbol{w}$

Step 1. Initialize $\boldsymbol{w} \in \mathbb{R}^{1\times n}$, and $\boldsymbol{acc} \in \mathbb{R}^{1\times 2n}$ as vectors of zeros
Step 2. **do**
Step 3. | **for** each weight $w_i \in \boldsymbol{w}$, **do**
Step 4. | | w_i++
Step 5. | | Calculate the most similar pair in each set of $\boldsymbol{S}$, denoted as $\boldsymbol{P}_s$
Step 6. | | Calculate acc_i ($acc_i \in \boldsymbol{acc}$) based on $\boldsymbol{P}_{Dr}$ and $\boldsymbol{P}_s$
Step 7. | | w_i--
Step 8. | **end for**
Step 9. | **for** each $w_i \in \boldsymbol{w}$, **do**
Step10. | | w_i--
Step11. | | **if** $w_i < 0$, w_i++, **continue**
Step12. | | Calculate the most similar pair in each set of $\boldsymbol{S}$, denoted as $\boldsymbol{P}_s$
Step13. | | Calculate acc_{n+i} based on $\boldsymbol{P}_{Dr}$ and $\boldsymbol{P}_s$
Step14. | | w_i++
Step15. | **end for**
Step16. | $acc_{max} = \max(acc_1, acc_2, \dots, acc_n, acc_{n+1}, \dots, acc_{2n})$, and let α be the number of maximum values.
Step17. | **for** i in range$(1, 2n)$
Step18. | | **if** $acc_i == acc_{max}$ && $i \leq n$
Step19. | | | w_i += $1/\alpha$
Step20. | | **else if** $acc_i == acc_{max}$ && $n+1 \leq i \leq 2n$
Step21. | | | w_{i-n} −= $1/\alpha$
Step22. | | **end if**
Step23. | **end for**
Step24. **until** acc_{max} keeps unchanged for a defined number iterations
Step25. return $\boldsymbol{w}$

* the source code is available at https://github.com/marlonli/PatientCohortsAnalysis

guided and dominated by the domain knowledge, leading to an "expected" result. Hence we designed a simple experiment to collect medical decisions and developed a learning algorithm for learning attribute weights.

Our experiment used 41 sets (denoted as $\boldsymbol{S}$) of three patients (e.g., Patient A, B, C in Table 5-2) drawn randomly from the trauma resuscitation dataset without replacement. A surgeon was asked to decide the most similar one among three pairs of patients, (A, B), (A, C) and (B, C), based on their context attributes only. They used their domain knowledge to judge how important the differences of attributes were, and to decide which pair of patients is more likely to be in the same cohort than others. In our example (Patient A, B, C in Table 5-2), our medical expert labelled patient pair (B, C) as the one that is most likely to be observed in the same cohort. This experiment is simple because it does not need much human effort and each decision in the experiment can be made easily. We then used these labeled results (denoted as $\boldsymbol{P}_{\mathrm{Dr}}$) as the input of our attribute learning algorithm (Alg.5.1).

Our learning algorithm was designed with the core idea that by adjusting the weights of context attributes, we can increase the classification accuracy (i.e., deciding which pair of

Table 5-2. Context attributes (1st column), a set of three patients (2nd-4th columns), and the weights learnt from Alg.5.1.

Attributes	Patient A	Patient B	Patient C	…	Weights Learnt
AGE Group	0	0	2	…	0
Gender (male = 1)	1	1	1	…	0
Transfer	1	1	0	…	0
Stat	0	0	1	…	0
Attending	0	0	0	…	0.14
Blunt	1	1	1	…	1
Penetrating	0	0	0	…	1.31
Animal Bite	0	0	0	…	0.81
Burn	0	0	0	…	0
No Injury	0	0	0	…	0.81
Non-critical admission	1	1	0	…	1
Critical Admission	0	0	0	…	0
Discharged	0	0	1	…	0
ETA Now	0	0	1	…	0
Weekend	1	0	0	…	0
ntubation	0	0	0	…	0.14
Daytime	0	0	1	…	0
GCS>13	1	1	1	…	0.14
ISS Group	0	0	0	…	0
AIS_HEAD_NECK	2	0	0	…	0
AIS_FACE	0	0	0	…	0
AIS_CHEST	0	0	0	…	2.39
AIS_ABD_PELVIC	0	0	0	…	0
AIS_EXTREMITIE	0	0	0	…	0.25
AIS_EXTERNAL	0	1	1	…	0
Maximum AIS	2	1	1	…	0

patient is more similar). The similarity measure is defined using weighted Euclidean distance [65][71], a modification of Euclidean distance with each attribute is weighted:

$$d_{AB} = (\sum_{i=1}^{n} w_i (x_i^{(A)} - x_i^{(B)})^2)^{\frac{1}{2}} \tag{5.1}$$

where w_i is the weight given to the *i*-th component. $x_i^{(A)}$ and $x_i^{(B)}$ are the *i*-th context attributes of patients A and B. If distance $d_{AB} < d_{AC}$ && $d_{AB} < d_{BC}$, it means that patient A and B is most similar pair given a set of weights $\boldsymbol{w} = [w_1, \dots, w_n]^T$. Then if the label given by medical expert is also pair (A, B), it is a hit, otherwise a miss. The overall classification accuracy over N patient sets is defined as the ratios of hits:

$$acc = \frac{num(hits)}{N} \tag{5.2}$$

Our algorithm updates the attribute weights iteratively. At each iteration, we test adding (Step 4) or subtracting (Step 10) a unit weight from a single attribute weight w_i. An important boundary condition is $w_i \geq 0$ (Step 11), otherwise w_i does not have physical

meaning in similarity calculation. Then we calculate the updated accuracy (Step 6 & 13) after addition and subtraction. Last, we update (Step 17-23) the attributes which lead to the highest accuracy (Step 16). The algorithm terminates when the accuracy stays unchanged for a defined number of iterations (Step 24). Alg.5.1 is based on greedy search [72]. At each step, we only update the weights on attributes which provide maximum improvement. Our algorithm gradually improves the weights and accepts the suboptimal solutions, while finding the optimal solution is computationally difficult.

5.2.3 Patient Cohorts Discovery

To discover patient cohorts, we clustered the patients. The clusters were calculated according to the similarity of patient context attributes. The patients being clustered together must share similar attributes. Hence the clusters can be treated as patient cohorts. The learning of attribute weights in the previous step helps us decide which attributes are more important and guide the clustering process to partition the patient data into clusters (cohorts) of more clinical meaning.

Numerous clustering algorithms were developed for all kinds of datasets and problems. Some clustering algorithms were specifically designed for certain data distributions (e.g., EM clustering algorithm on Gaussian distribution and DBSCAN on noise data). In our study, the patient context attributes can be heterogeneous, including categorical, binary, numerical and ordinal attributes. Hence, to achieve the best generation of our framework, we chose two most commonly used clustering algorithms, k-means clustering (centroid-based) [73][74] and hierarchical clustering (connectivity-based) [74].

In addition, selecting the number of clusters is a difficult and well-known problem. Hierarchical clustering itself is widely used to intuitively decide the optimal number of clusters when the results were visualized in the dendrogram. Another widely used method is silhouette analysis [65]. The silhouette value is a measure of how similar a data point is to its own cluster (cohesion) compared to other clusters (separation). The silhouette score ranges from -1 to +1, where a high value usually indicates a better clustering configuration. We used both methods in our study.

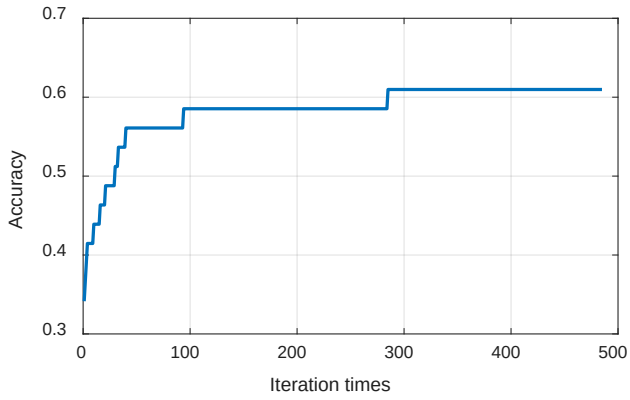


Figure 5.1. Attribute weight learning process. The accuracy increases with the number of the learning iterations.

5.2.4 Workflow Discovery and Sequential Pattern Mining

To discover the treatment patterns and procedures of different patient cohorts, we used two different techniques, workflow discovery (Section 2.2) and sequential pattern mining. In this study, we used Disco (https://fluxicon.com/disco/), a process mining tool based on a fuzzy workflow mining algorithm [44]. The workflow discovery algorithms tend to produce spaghetti-like models [20] which are difficult to interpret for analysts. And searching for the differences in treatment patterns in several workflow models can be even more challenging. The sequential pattern approach can help address this limitation. Treatment patterns can be discovered from activity traces using sequential pattern mining algorithms. Although numerous sequential patterns may be found, the significance of the patterns can be evaluated using the statistical methods. Manual analysis only needs to be done on patterns that are shown as statistical significant. In our implementation, we used SPADE [75], an efficient algorithm for mining frequent sequential patterns.

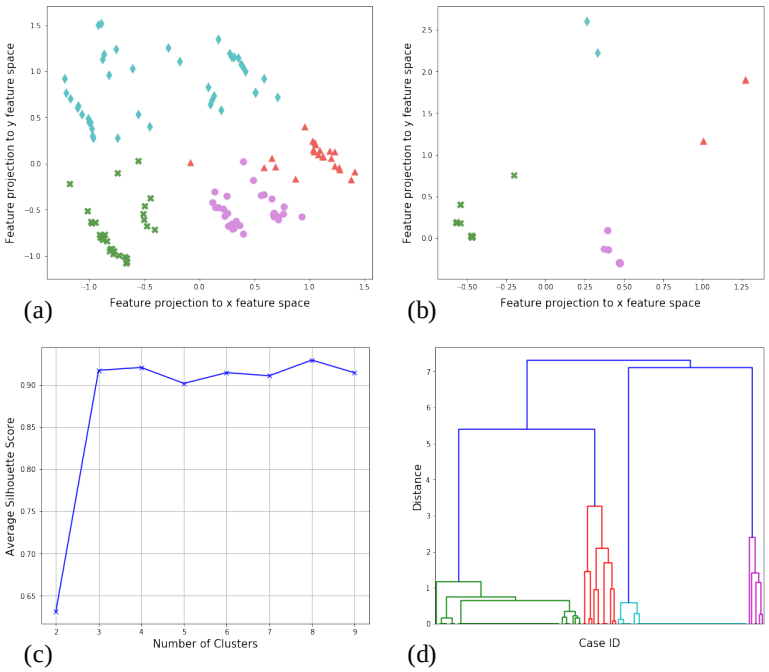


Figure 5.2. (a) K-means clustering results of 123 patients without unit weights. The number of clusters (k) was set as 4. Dots of the same color from a same cluster. (b) K-means clustering results (k = 4) of 123 patients using weighted attributes. (c) The value of average silhouette (y-axis) changes as the number of clusters (x-axis). (d) The dendrogram plot of the hierarchical clustering results.

5.2.5 Statistical Analytics

We used statistical analysis to study the association between the treatment patterns and patient cohorts. Differences in number of treatment patterns between patient cohorts were compared using Student's t-test [76]. Student's t-test compares the means of two sets of data (dataset size can be small) and tells the difference from each other. Two different kinds of comparisons can be performed for patient cohort analysis, comparison between two cohorts (one-vs.-one) and one cohort and the rest of the cohorts (one-vs.-rest). We defined statistical significance level set at $p < 0.05$ [76].

5.3 Experiments

Our experimental results involve three aspects, the performance of attribute learning algorithm, the patient cohorts we discovered and the resuscitation patterns as well as their significance test results.

5.3.1 Attribute Weights

The initial accuracy (Figure 5.1) was 0.34 before applying the weight learning algorithm. The initial accuracy was calculated with unit weights. The accuracy increased quickly in the first 50 iterations before slowing down. The accuracy always increased and never dropped because the learning algorithm (Alg.5.1) is guided by accuracy, only updating the weights when if accuracy increases. The accuracy stopped at 0.61 when the learning terminated and was 0.27 higher than that of using unit weights. The accuracy may stop higher if more data was labeled so that it covers more comparisons. Our results (Table 5-2) showed that the injury features (injury mechanism, injury area and severity score) are more important than patient demographics (age, gender, etc.) and resuscitation attributes (i.e., stat, attending, daytime, etc.). Most attributes of patient demographics and resuscitation have zero weights. Two resuscitation attributes that have non-zero weights are “attending” and “intubation”. Both “stat” and “attending” represents the patient arriving from the scene but attendings are more severe cases. The “intubation” is an important attribute indicating whether the patients were intubated prehospital. The patients who could not maintain their airway were intubated. Within injury features, the injury mechanism (penetrating, blunt, etc.) has the most non-zero weights. “AIS_CHEST” (injury severity of chest) has the highest weights.

5.3.2 Patient Cohorts

The silhouette analysis suggested the number of clusters as four or eight (two peaks in Figure 5.2 (c)). From dendrogram (Figure 5.2 (d)), we can easily identify four clusters. We decided the number of cluster as four. Although the silhouette score is higher if we have four more clusters (eight in total), our result showed the additional four clusters were partitioned from the two smaller clusters of the four clusters. Some of them were too small

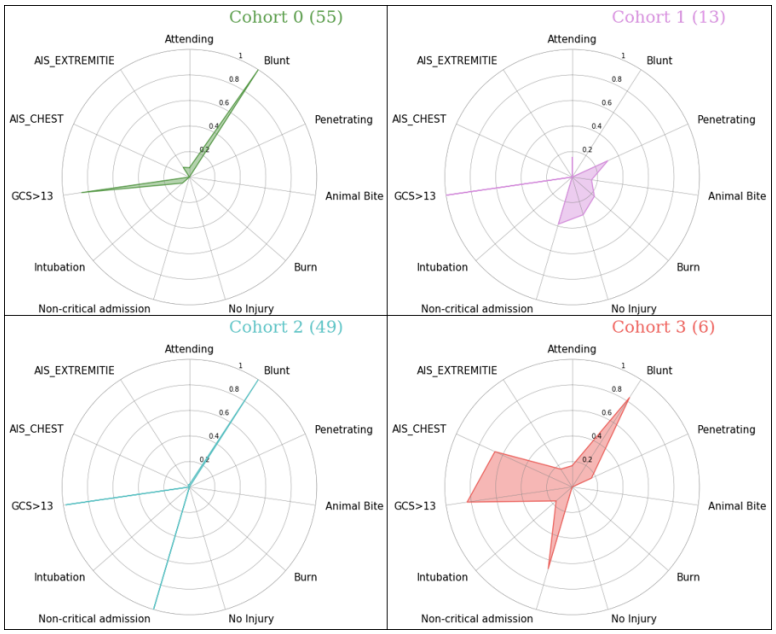


Figure 5.3. Radar charts visualizing the characteristics of the patient cohorts. Each radar chart represents a patient cohort. Each attribute is depicted by the node on the spoke. The number of patients in each cohort is shown in the parenthesis, e.g., cohort 0 has 55 patients.

(few data points), making them a better fit for specific case studies rather than being included in a cohort analysis.

Our result also showed that the k-means clustering result was conformant (100% same) with the hierarchical clustering result when having four clusters (Figure 5.2 (b)(d)). The k-means clustering result was visualized after as the dimension reduction with PCA [77]. We also ran the clustering on the same dataset with unit attribute weights (Figure 5.2 (a)). Great difference can be noticed from the two k-means clustering results (Figure 5.2 (a) vs. (b)). With unit attribute weights, four similar sizes of clusters were formed. No clear boundary can be noted between the clusters. The distribution of data points (a) is much sparser than that of (b) because all attributes were taken in the similarity measurement. As it is usually of a small chance that two patients have exact same or highly similar context attributes, the data points in (a) cannot be distributed as dense as the clusters in (b), where only important attributes were taken into account (Table 5-2). Four patient cohorts (from cohort 0 to cohort 3) include 55, 13, 49 and 6 patients respectively. To better understand the characteristics of each patient cohort, we used a radar chart (Figure 5.3) to help visualize attribute distribution within each patient cohort. We also calculate the significance of each attribute

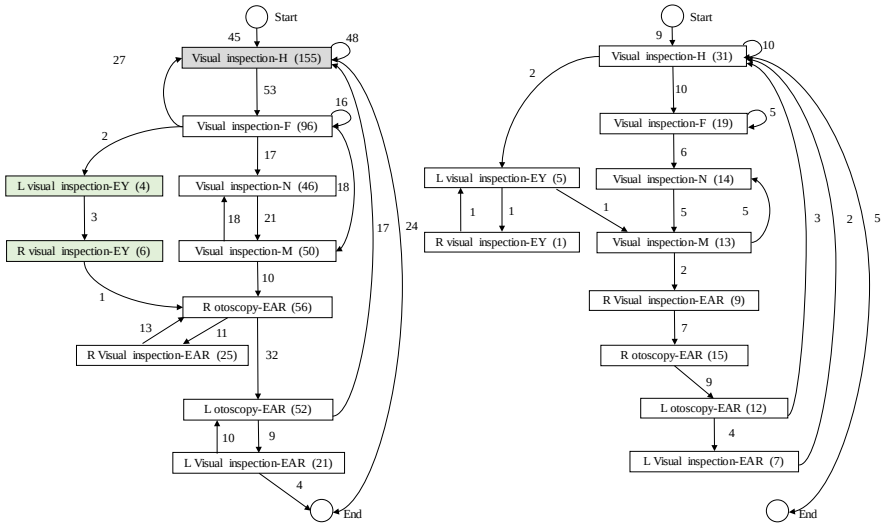
Table 5-3. p-values of each attribute of a cohort versus other cohorts.

Attributes Name	Cohorts 0	Cohorts 1	Cohorts 2	Cohorts 3
Attending	0.758	0.172	0.104	0.305
Blunt	0.000	0.000	0.001	0.679
Penetrating	0.040	0.000	0.064	0.111
Animal Bite	0.203	0.000	0.249	0.749
No Injury	0.068	0.000	0.100	0.648
Non-critical Admission	0.000	0.511	0.000	0.330
Intubation	0.107	0.437	0.064	0.111
GCS>13	0.019	0.260	0.045	0.437
AIS_CHEST	0.042	0.375	0.034	0.000
AIS_EXTREMITIE	0.756	0.529	0.208	0.214

in its cohort versus that attribute of the rest cohorts (Table 5-3). As we have a large number (26) of context attributes, we filtered the attributes by just showing the attributes that have a non-zero weight (Table 5-2). Both cohort 0 and cohort 2 included patients with blunt injuries and GCS over 13. The difference is whether the patients were critically admitted. Cohort 1 and 3 are two smaller cohorts. They have very different distributions from each other, and from the cohort 0 and 2. Cohort 1 includes patients of the remaining injury mechanisms except for blunt. Only cohort 3 includes patients with injury on the chest.

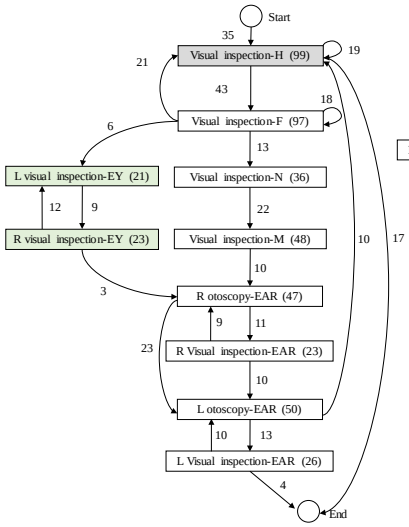
5.3.3 Resuscitation Workflow and Patterns

With patient cohorts identified, we performed workflow mining and sequential pattern mining on each cohort. The complete workflow models are spaghetti-like because of a large number of non-zero transitions. To obtain descriptive and interpretable workflow models, we applied two model simplification methods. First, we focused our study on a specific medical phase each time. A medical phase a part of the complete trauma resuscitation process, e.g., airway assessment phase (checking patient's airway), disability assessment phase (assessing patient's disability level), head assessment phase (assessing injuries on patient's head), etc. Second, we pruned the workflow model by only preserving the most dominant incoming and outgoing transitions for each node. This method omitted insignificant details, i.e., a large number of transitions of rare occurrence.



(a) Cohort 0 (55)

(b) Cohort 1 (13)



(c) Cohort 2 (49)

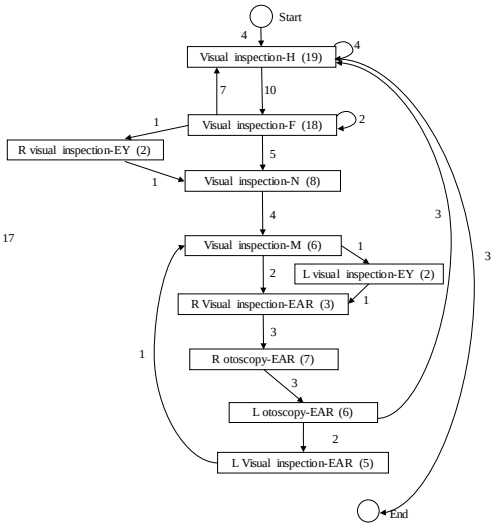


Figure 5.4. Workflow models discovered from patient cohort 0 and cohort 2. The major differences are highlighted in the figure. Each node includes an activity type and the count of its occurrences. The transition represents the sequential order of activities and the numbers on the transitions represent the count of such sequential pattern.

Our workflow results (Figure 5.4) on head phase showed high similarity in the four head assessment workflows. All of them follow a similar sequential order as “head (H) -> face (F) -> (nose (N) -> mouth (M)) || (eye (EY))) -> ear (EAR)”. Two differences can be noticed from workflows of cohort 0 and cohort 2. First, the occurrence of “visual

Table 5-4. Significant resuscitation patterns discovered from cohort 0 and cohort 2

Resuscitation Patterns	Occur. Frequency		Raw Count (Average)		p-value
Visual inspection-H	96.43%	97.96%	155 (2.87)	99 (2.06)	0.026
Visual inspection-RUE	89.29%	93.88%	120 (2.4)	76 (1.65)	0.046
Visual inspection-BK	94.64%	100.00%	94 (1.77)	62 (1.27)	0.001
Visual inspection-H → ⋯→* R otoscopy-Ear	42.59%	10.20%	23	5	0.000
Visual inspection-H → ⋯→ Palpation-RLE	29.63%	4.08%	16	2	0.000
Log roll-BK → ⋯→ Visual inspection-BK, T-spine-BK	24.07%	2.04%	13	1	0.001
Visual inspection-M → Visual inspection-N	33.33%	8.16%	18	4	0.002
Visual inspection-BK, T-spine-BK → ⋯→ L-spine-BK	22.22%	2.04%	12	1	0.002

* "→ "- direct sequence; "→ ⋯→ "- intervening tasks allowed

inspection-head" is more frequent in cohort 0 than that in cohort 2. On the other hand, "left/right visual inspection-eye" are much more frequent in cohort 2. The medical explanation is activities "left/right visual inspection-eye" can be optional because there is another pair of activity "right/left pupil check" in disability assessment phase (prior to head assessment phase) to evaluate patient's disability level. The pupil examination requires a light source to be used to assess pupil response. This exam is more thorough than only performing an unaided visual examination of the eye. In addition, considering most patients in cohort 0 are critically admitted patients with more severe conditions, medical team tended to omit these unimportant eye visual inspections to save time.

By performing sequential mining algorithms on patient cohort 0 and 2, we discovered 39784 sequential patterns in total. 178 sequential patterns were computed as statistical significant (a small part is shown in Table IV). For example, "visual inspection-head" is found to occur on average 2.87 times in cohort 0 versus 2.06 times in cohort 2 (p-value = 0.026). Similarly, "visual inspection-back" is found to occur on average 1.77 times in cohort 0 versus 1.27 in cohort 2. The potential medical explain is the patients in cohort 0 were of critical admission types, indicating they may have more severe injuries than patients of cohort 2. Hence, it is more likely that after the medical team members perform rapid evaluation, they need to confirm the initial findings by reassessing the patient.

Part III

Process Recommender System

Part I: Introduction

Chapter 1 Introduction	Chapter 2 Preliminaries

Part II: Applied Process Mining and Analysis

Chapter 3 Workflow Model Discovery	Chapter 4 Workflow Deviation Analysis	Chapter 5 Patient Cohorts Analysis

---

Part III: Process Recommender System

Chapter 6 Trace-level Recommendation	Chapter 7 Activity-level Recommendation

---

Part IV: Implementation and Conclusion

Chapter 8 VIT-PLA	Chapter 9 Conclusions

Chapter 6
A Data-driven Process Recommender Framework

This chapter on Data-driven Process Recommender System is based on our paper [3]. In this study, we present an approach for improving the performance of complex knowledge-based processes by providing data-driven step-by-step recommendations. Our framework uses the associations between similar historic process performances and contextual information to determine the prototypical way of enacting the process. We introduce a novel similarity measure for grouping traces into clusters that incorporate temporal information about activity performance and handles concurrent activities. Our data-driven recommender system selects the appropriate prototype performance of the process based on user-provided context attributes. Our approach for determining the prototypes discovers the commonly performed activities and their temporal relationships. We tested our system on data from three real-world medical processes and achieved recommendation accuracy up to an F1 score of 0.77 (compared to an F1 score of 0.37 using ZeroR) with 63.2% of recommended enactments being within the first five neighbors of the actual historic enactments in a set of 87 cases. Our framework works as an interactive visual analytic tool for process mining. This work shows the feasibility of data-driven decision support system for complex knowledge-based processes.

6.1 Introduction

Contemporary information systems, such as personal calendars and electronic health records (EHR), often record activity logs. Process mining techniques attempt to extract non-trivial knowledge and insights from activity logs and use them for further analyses [18]. Process mining techniques have been applied to practical problems, assisting in visualizing, interpreting and diagnosing processes [18]. Existing recommender systems have not been developed based on process mining. Our current work presents such a bridge. We are designing a data-driven process analysis and recommender system that can provide contemporaneous recommendations of process steps and help with retrospective analyses

of the process. Our approach relies on mining historic data to uncover the potential association between the way of enacting a process and contextual attributes. If association tests are significant, we train a recommender system to output a prototypical enactment for the given context attributes.

Unlike most recommender systems that propose one or few next steps at a time, our system initially recommends all steps at once. Although it may not be feasible for the performers to study and follow a long list of steps, this recommendation can be used at runtime to automatically verify the process compliance and detect omitted steps and other process errors. Our framework has two stages: process analysis and process recommendation (Figure 6.1 (a)). Process analysis includes: (1) clustering of historic traces based on similarity; (2) determining the cluster prototypes that represent the established process enactment for each cluster; (3) regression analysis to explore the correlation between cluster membership and context attributes; and (4) interactive visualization and statistical analysis of process traces. The recommendation stage includes: (1) predicting the cluster to which the given trace belongs based on the observed context attributes, and (2) displaying the prototype of the predicted cluster as the recommended enactment.

Key technical challenges for this system include measuring the similarity of process traces and determining the cluster prototypes. Similarity measurement strongly affects the results of trace clustering and plays a key role in our system. Several measures of trace similarity exist but suffer from either inaccurate measurement because of timeline stretching needed to normalize the trace duration and compute the overlap between the traces, or information loss from forced sequencing of concurrent activities needed to apply edit distance or pattern-based distance [2]. Another challenge is determining a prototype that represents the recommended sequence of steps for each cluster. Our contributions include:

- A novel measure of pairwise similarity between process traces based on time warping. Unlike existing similarity measures (edit distance, pattern-based distance, and Euclidean distance based on a normalized timeline), our approach incorporates the time information while correcting for temporal differences between the same activities in different process traces, such as different start times, idle times and

duration of the performance. Our approach also handles concurrent activities and parallel activities for which the order of performance is irrelevant.

- A novel approach for determining a prototype for a cluster of process traces. Our prototype captures the established enactment for a given context and considers the temporal relationships between activities. It achieves a higher average similarity to process traces in its cluster than the cluster medoid.
- A data-driven recommender system that selects a representative enactment based on user-provided context attributes. We tested our system on data from three real-world medical processes and achieved high recommendation accuracy.

6.2 Related Work

Complex knowledge-based processes are usually performed based on domain knowledge and standard protocols. For example, for trauma resuscitation the Advanced Trauma Life Support (ATLS) protocol [16] suggests the workflow based on treatment priorities: Airway → Breathing → Circulation → (Neurological) Disability. Clarke et al. [78] and Fitzgerald et al. [47] developed computer-aided decision support that recommends next steps to reduce human errors. These systems rely on rules manually specified by domain experts, lack generalizability, and are subject to human bias. We present an automatic, data-driven, label-free framework for process analysis and recommendation.

Our framework incorporates three main techniques: similarity measures for process traces, trace clustering algorithms, and cluster prototype extraction. These techniques have been well studied in the analysis of time series [79], but are not applicable to process data. Unlike time series with numerical values, process data is typically categorical, representing different activity types and their properties. Different process datasets may have very different features and no rule exists to decide a similarity between traces of process enactment. Common similarity measures include edit or Levenshtein distance [38][30] and pattern-based similarity, e.g., n-gram [80][81]. Both measures accept as input only process traces represented as sequences, which requires that concurrent activities are sequenced (e.g., by activity start time) and that temporal information on activity duration and idle times is ignored. Forestier et al. [82][83] proposed dynamic time warping (DTW) as a similarity measure for process traces. The DTW, however, cannot handle concurrent

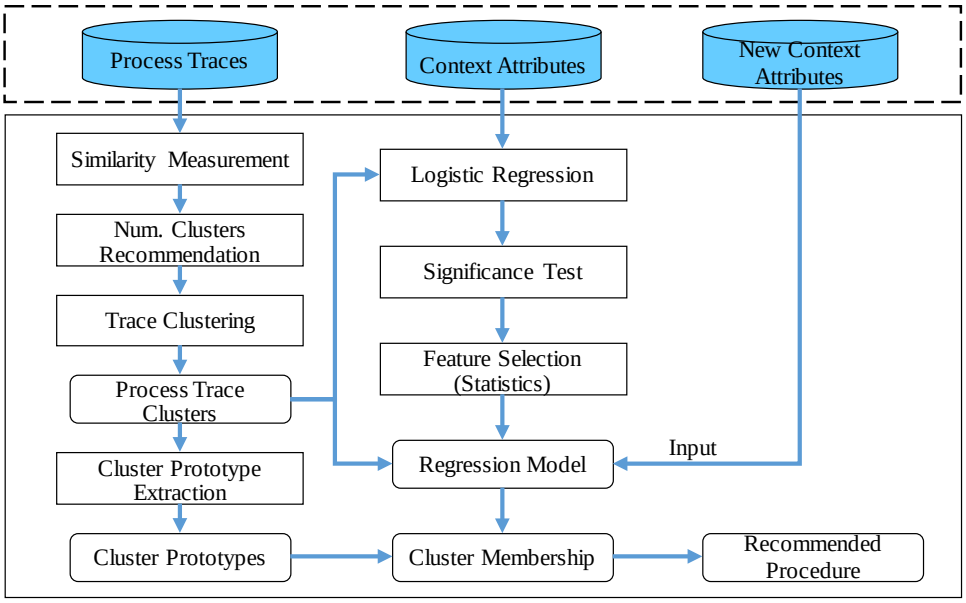
activities, does not consider idle time intervals, and has other issues when used for process traces [1]. In addition, Forestier et al. considered processes that are mostly sequential (non-concurrent), with no activities for which the order of performance is irrelevant. To address these challenges, we introduce a novel similarity measure based on time warping that incorporates temporal information, such as activity start time, performance duration, and idle intervals.

Hierarchical clustering has been commonly used for process trace clustering [82][84][85][86]. This algorithm does not need a predefined number of clusters and produces a visually intuitive dendrogram (tree diagram). Its main limitation is its computational complexity, generally $O(n^2\log(n))$ where n is the number of traces, which makes it too slow for large datasets. We implemented hierarchical clustering in our framework as well as two other state-of-the-art clustering algorithms.

Cluster prototype candidates can be determined using different techniques. A widely used cluster centroid represents the cluster center with a minimum distance to other points in the cluster, e.g., sum-squared distance [79]. For categorical and event-based data, however, the notion of a "center" may not apply [79]. For example, the centroid of categorical data {orange, apple, banana} cannot be determined. An alternative is the cluster medoid as the most representative data object in the cluster—an existing object that has a minimal average dissimilarity to all other objects in its cluster. The medoid, however, may not be adequate when no "suitable" representative exists in the cluster. Another kind of prototype is the consensus sequence, a sequence of commonly observed activities found by aligning many process traces [1][2]. The consensus sequence, however, represents only the order of performance without temporal information. We introduce a novel approach for cluster prototype extraction that incorporates temporal information.

6.3 Process Recommender Framework

Our framework performance (i.e., recommendation accuracy) does not depend as much on the recommender model as on the ability to capture significant commonalities between process performances using a similarity measure and clustering, as well as on determining the proper cluster prototype. Therefore, we focus on the similarity measure, clustering, and prototype extraction for assessing the performance.



(a) An overview of framework

Case ID	Activity	Start Time	End Time
xx1	Patient Arrival	0:00:00	0:00:01
xx1	NRB	0:00:00	0:00:01
xx1	Pre-Oxy Chest Ausc	0:01:08	0:01:23
xx1	Pre-Oxy Breath Verb	0:01:48	0:01:49
xx1	Airway Assessment	0:05:59	0:06:08
xx1	BVM	0:06:43	0:06:44
xx1	Critical Window	0:07:19	0:07:20
xx1	RSI Sedative Meds	0:07:50	0:08:02
xx1	RSI Paralytic Meds	0:08:16	0:08:32
xx1	BVM	0:09:52	0:09:53
xx1	Laryngoscopy	0:10:19	0:10:51

(b) Medical process trace

Case ID	xxx1	xxx2
Age category	24-96	24-96
Sex	Male	Female
Intubator	PEM Attending	PEM/ED Resident
Direct laryngoscopy	1	1
Night Shift	1	0
Reason	Seizure	Respiratory Distress
Type of Call	ED Patient	Now
Height (cm)	86	90
Weight (kg)	13	16.4
BMI	17.6	20.2
Num. Intubation Attempts	3	3

(c) Process case context attributes

Figure 6.1. Data sample and our framework structure.

6.3.1 Terms and Definitions

A performance of a process can be captured with activity codes and timestamps. We represent each **activity** by its type and performance time (Figure 6.1 (b)) denoted as $A = \{A^{type}, A^{ts}, A^{te}\}$, where A^{type} is the activity type, A^{ts} is the start time, and A^{te} is the end time. A **process case** $c = \{id, \boldsymbol{x}, \boldsymbol{T}\}$ is an instance of process performance. It is indexed with a unique case *id* and consists of the *trace* $\boldsymbol{T}$ which is a vector of performed activities (internal information), and the vector $\boldsymbol{x}$ of *context attributes* (external information). An *i*th **process trace** is represented as $\boldsymbol{T}_i = [A_{i1}, \ldots, A_{ik}]$, where k is the trace length (number of performed activities). To make explicit concurrent activities, we use a matrix representation of traces

as $\boldsymbol{T}_i = [\boldsymbol{p}^i_1, .., \boldsymbol{p}^i_{k^i}]$, where the duration of ith trace is discretized into k^i time units and in each time unit m the vector $\boldsymbol{p}^i_m = [a^1, \ldots, a^\ell]$ represents the execution status of all ℓ activity types. If an activity of type a^j is being performed during time m, then $a^j = 1$ and $a^j = 0$ otherwise. The magnitude of each activity vector is $|\boldsymbol{p}^i_m| = \sum_j |a^j|$. **Context attributes** (or external attributes) record the contextual information of a process case, such as the patient demographics (Figure 6.1 (c)) in a vector $\boldsymbol{x} = [x_1, \ldots, x_d]^T$ of d observed attributes x_i. By associating context attributes with step-by-step activities based on historic data, we can recommend the best process enactment for given attributes.

A **process trace cluster** $\boldsymbol{C} = [\boldsymbol{T}_1, \ldots, \boldsymbol{T}_c]$ is a group of c traces that are similar in terms of type, activity performance order and times. The cluster membership is determined by information internal to process traces. A **prototype trace** of a cluster is the most representative or typical process enactment for this cluster. This representative enactment can be an actually observed trace (an exemplar) or derived from other traces in the cluster. Cluster prototypes summarize the cluster information and highlight the commonalities of the process traces, which can help visualize and compare the differences between different clusters.

A **recommended process trace** is determined using both internal and contextual information of historic traces to find a standardized process performance. This trace can be used to guide the process performance or verify the process compliance and detect omitted steps and other process errors.

6.3.2 Trace Similarity based on Time Warping

The process traces we considered are not simple sequences just recording activity type and the order of their performance, but concurrent timelines showing the performance status of each activity type over time. Pairwise comparison of these composite traces is challenging. An effective similarity measure should combine (*i*) intrinsic activity likeness, e.g., some activities are mutually substitutable, (*ii*) activity performance time, (*iii*) relative order of performance, and (*iv*) temporal variation between different performances. The temporal variation has several causes, such as activities initiated at different times relative to the process start, performed at different speeds, omitted or repeated. The same activities may have different temporal characteristics in different traces and traces may have different

duration. Although several similarity measures exist for temporal sequences [30][38][82][83][84][85][86], none satisfies the above requirements.

We introduce a novel similarity measure for complex process traces using timeline warping to determine the optimal pairwise alignment (Figure 6.2). Our measure considers both the sequential order and temporal overlaps of activities during this optimization. We define the similarity between traces $\boldsymbol{T}_i$ and $\boldsymbol{T}_j$ as:

$$s(i,j) = \frac{\left|\boldsymbol{T}_i \cap \boldsymbol{T}_j\right|}{\left|\boldsymbol{T}_i \cup \boldsymbol{T}_j\right|} \tag{6.1}$$

where $|\boldsymbol{T}_i| = \sum_m |\boldsymbol{p}^i{}_m|$ is the total performance time of activities in trace $\boldsymbol{T}_i$ and $\boldsymbol{p}^i{}_m$ is the vector of performance status of all activities in mth time unit. $|\boldsymbol{T}_i \cap \boldsymbol{T}_j|$ is the time when both traces had same activities performed and $|\boldsymbol{T}_i \cup \boldsymbol{T}_j|$ is the time when one or both traces had same activities performed. If we define $|\boldsymbol{T}_i \otimes \boldsymbol{T}_j|$ as the time when only one trace had activities performed, then the total active time in a pair of traces is:

$$|\boldsymbol{T}_i| + \left|\boldsymbol{T}_j\right| = \left|\boldsymbol{T}_i \cup \boldsymbol{T}_j\right| + \left|\boldsymbol{T}_i \cap \boldsymbol{T}_j\right| \tag{6.2}$$

and

$$|\boldsymbol{T}_i| + \left|\boldsymbol{T}_j\right| = \left|\boldsymbol{T}_i \cup \boldsymbol{T}_j\right| + \left|\boldsymbol{T}_i \cap \boldsymbol{T}_j\right| \tag{6.3}$$

By combining these equations, the similarity of $\boldsymbol{T}_i$ and $\boldsymbol{T}_j$ is:

$$\begin{aligned} s(i,j) = \frac{\left|\boldsymbol{T}_i \cap \boldsymbol{T}_j\right|}{|\boldsymbol{T}_i| + \left|\boldsymbol{T}_j\right| - \left|\boldsymbol{T}_i \cap \boldsymbol{T}_j\right|} &= \frac{|\boldsymbol{T}_i| + \left|\boldsymbol{T}_j\right|}{|\boldsymbol{T}_i| + \left|\boldsymbol{T}_j\right| - \left|\boldsymbol{T}_i \cap \boldsymbol{T}_j\right|} - 1 \\ &= \frac{\mathbf{2}|\boldsymbol{T}_i| + \mathbf{2}\left|\boldsymbol{T}_j\right|}{|\boldsymbol{T}_i| + \left|\boldsymbol{T}_j\right| + \left|\boldsymbol{T}_i \otimes \boldsymbol{T}_j\right|} - 1 \end{aligned} \tag{6.4}$$

The only variable term in this equation during warping alignment of two traces is $|\boldsymbol{T}_i \otimes \boldsymbol{T}_j|$. The optimal warping path between $\boldsymbol{T}_i$ and $\boldsymbol{T}_j$ is $\boldsymbol{P}^{ij} = \{\boldsymbol{p}^{ij}_{mn}\} = \{(\boldsymbol{p}^i_m, \boldsymbol{p}^j_n)\}$, which is the solution to this optimization problem:

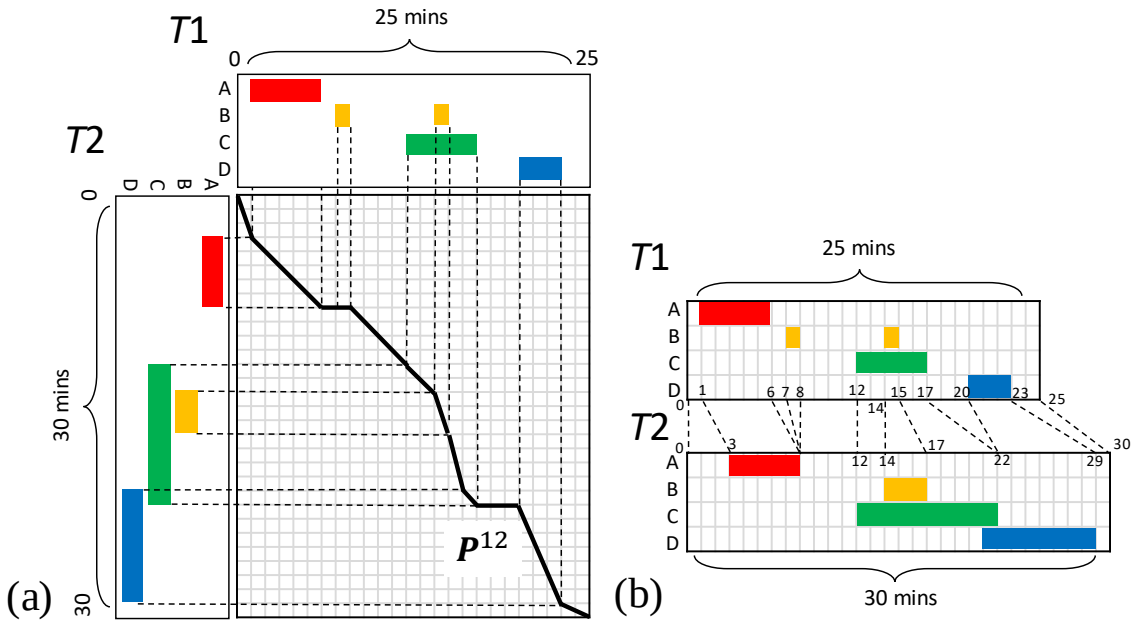


Figure 6.2. Our time warping approach to find the minimum warping distance between two process traces $T1$ and $T2$. (a) Illustration of the warping path calculated between $T1$ and $T2$ (Eq.6.4). (b) Alignment of the warped timelines.

$$\underset{p^{ij}}{\operatorname{argmin}} \left| \boldsymbol{T}_i \otimes \boldsymbol{T}_j \right| = \sum_{m,n} \left| (\boldsymbol{p}^i_m - \boldsymbol{p}^j_n)(\boldsymbol{J}_\ell - \boldsymbol{S}^a)\boldsymbol{w} \right| \quad s.t. \quad \bigcup_m \boldsymbol{p}^i_m \in \boldsymbol{T}_i \quad \text{and} \quad \bigcup_n \boldsymbol{p}^j_n \in \boldsymbol{T}_j \tag{6.5}$$

where $\boldsymbol{w} = [w_1, \ldots, w_\ell]^T$ is a vector of weights indicating that some activities are more important than others. The weight can be any positive real number and the default is 1. When the weights are included, the trace magnitude is redefined as $|\boldsymbol{T}_i| = \sum_m |\boldsymbol{p}^i_m \boldsymbol{w}|$. The ℓ-by-ℓ matrix $\boldsymbol{S}^a(i,j) \in [0,1]$ represents the degree to which any pair of ℓ activity types are substitutable and $\boldsymbol{S}^a(i,j) = 1$ when activity types a^i and a^j are identical. An ℓ-by-ℓ matrix $\boldsymbol{J}_\ell$ of all ones is used to determine the distance between pairwise activity types as $\boldsymbol{J}_\ell - \boldsymbol{S}^a$. The weights and substitutability information are optional and may be given by domain experts when appropriate. Otherwise, they will default to a vector of ones and an identity matrix, respectively. Examples illustrate the influence of activity weight (Figure 6.3 (a)(c)) and

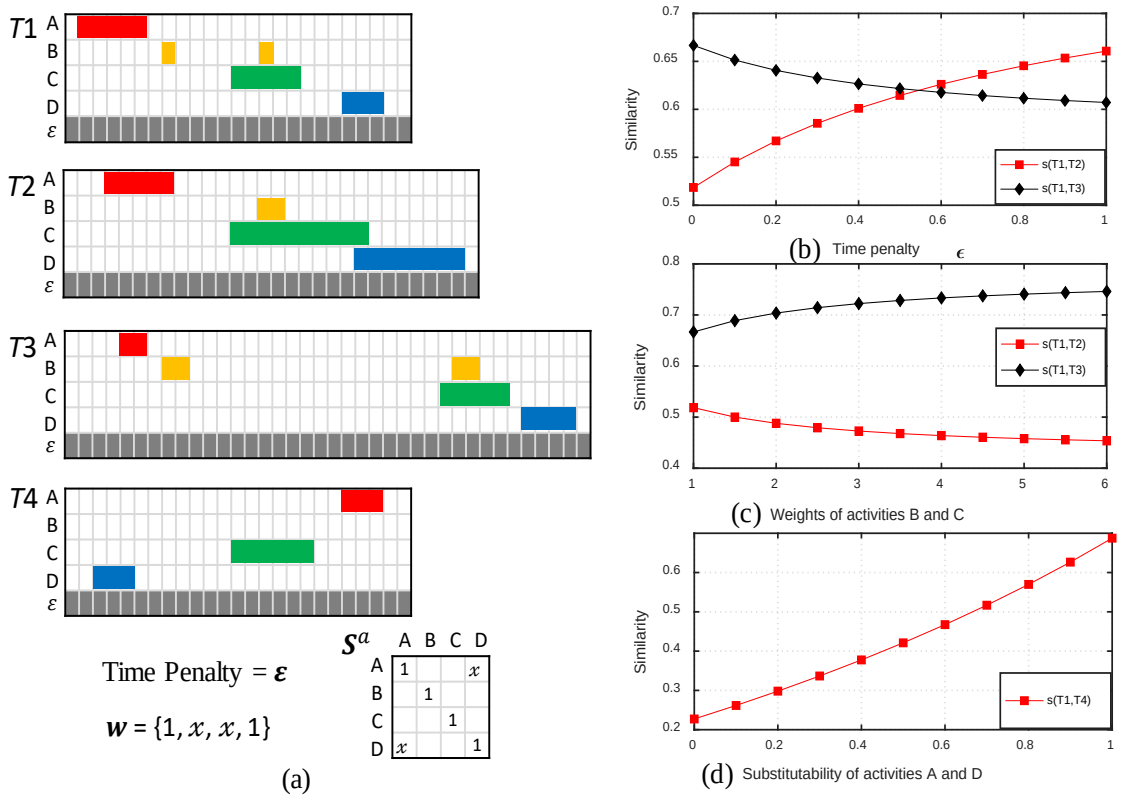


Figure 6.3. (a) Example traces T_1 – T_4 showing how the similarity results are affected by (b) the time penalty ε, (c) activity weights w, and (d) activity substitutability S^a.

substitutability (Figure 6.3 (a), (d)). Eq.6.5) can be solved similarly as Levenshtein distance [38] using dynamic programming with a novel *score function*:

$$t^{ij}(g,h) = \begin{cases} -\sum_{m=0}^{g} |\boldsymbol{p}^i_m \boldsymbol{w}| - \sum_{n=0}^{h} |\boldsymbol{p}^j_n \boldsymbol{w}| - \epsilon, & \text{if } \min(g,h) = 0 \\ \max \begin{cases} t^{ij}(g-1,h-1) - |(\boldsymbol{p}^i_g - \boldsymbol{p}^j_h)(\boldsymbol{J}_\ell - \boldsymbol{S}^a)\boldsymbol{w}| \\ t^{ij}(g-1,h) - |\boldsymbol{p}^i_g \boldsymbol{w}| - \epsilon \\ t^{ij}(g,h-1) - |\boldsymbol{p}^j_h \boldsymbol{w}| - \epsilon \end{cases} \end{cases} \tag{6.6}$$

The score function $t^{ij}(g,h)$ is defined for alignment costs of two time units $\boldsymbol{p}^i_g$ and $\boldsymbol{p}^j_h$. For aligning traces $\boldsymbol{T}_i$ and $\boldsymbol{T}_j$, we define the (k^i+1)-by-(k^j+1) score matrix $\boldsymbol{t}^{ij}$. The time-penalty vector $\boldsymbol{\varepsilon} = [\epsilon, \epsilon, \ldots, \epsilon]^T \in \mathbb{R}^{1\times k}$ is designed to penalize excessive warping of the timeline (grayed out bottom rows of traces in (Figure 6.3(a)). When $\epsilon = 0$, the timeline can be warped without cost, which may declare a short trace similar to a long trace. Constant ϵ

Algorithm 6.1. Time-warping Similarity of Process Traces (TwS-PT)

Input: $\boldsymbol{T}_i$, $\boldsymbol{T}_j$

Output: $s(i,j)$

Step1. Initialize $\boldsymbol{T}_i = [\boldsymbol{p}^i_1,..,\boldsymbol{p}^i_{k^i}]$, $\boldsymbol{T}_j = [\boldsymbol{p}^j_1,..,\boldsymbol{p}^j_{k^j}]$, $\boldsymbol{P}^{ij} = \{\emptyset\}$, $|\boldsymbol{T}_i| = \sum_g |\boldsymbol{p}^i_g|$, $|\boldsymbol{T}_j| = \sum_h |\boldsymbol{p}^j_h|$, $\boldsymbol{t}^{ij} = \{\emptyset\}$.

Step2. Fill score matrix $\boldsymbol{t}^{ij}$ progressively using Eq.6.6;

Step3. Deduce $\boldsymbol{P}^{ij}$ by tracing back $\boldsymbol{t}^{ij}$ from $t^{ij}(k^i,k^j)$ to $t^{ij}(0,0)$ and at each step choosing the neighboring cell that yields the maximum score (Eq.6.6).

Step4. $|\boldsymbol{T}_i \otimes \boldsymbol{T}_j| = (-1) * \boldsymbol{t}^{ij}(k^i,k^j)$;

Step5. **return** $s(i,j)$ computed using Eq.6.4

can be heuristically set to the reciprocal of the standard deviation of case duration. When time penalty $\boldsymbol{\varepsilon}$ is applied, the trace magnitude is redefined as $|\boldsymbol{T}_i| = \sum_g |\boldsymbol{p}^i_g \boldsymbol{w}| + \epsilon$. The above problem is a combinatorial optimization of interval data. We first discretize the time axis and then use a time warping algorithm to find the optimal warping path $\boldsymbol{P}^{ij}$ and similarity s_{ij}. Alg. 6.1 (TwS-PT) shows our approach for calculating the similarity of process traces.

6.3.3 Clustering Process Traces

To determine the recommended enactments from a large number of process traces, we clustered the traces. Exemplar-based clustering (EC) is an important category of clustering algorithms. These algorithms first select exemplars (representative points) from the whole dataset and then assign the remaining objects to their nearest exemplar. EC includes classic clustering algorithms, like *K*-means and *K*-medoids, and recent methods, like Affinity Propagation (AP) [87] and Density Peaks based Clustering (DPC) [88]. Because the similarities of process traces are measured pairwise, we chose to use clustering methods that take similarity matrix as input. We used Hierarchical Clustering, AP, and DPC.

Selecting the number of clusters is a difficult and well-known problem. Our method for setting this number is motivated by an intuition about cluster perception. A set of data points projected onto a similarity space observed from distance would appear as having fewer clusters than when observed up close. We propose that the number of clusters that remains stable over the greatest range of observation granularities represents the most probable structure of the dataset. We used AP clustering to analyze how the number of clusters varies with perception granularity. In methods like *K*-means, *K*-medoids, and

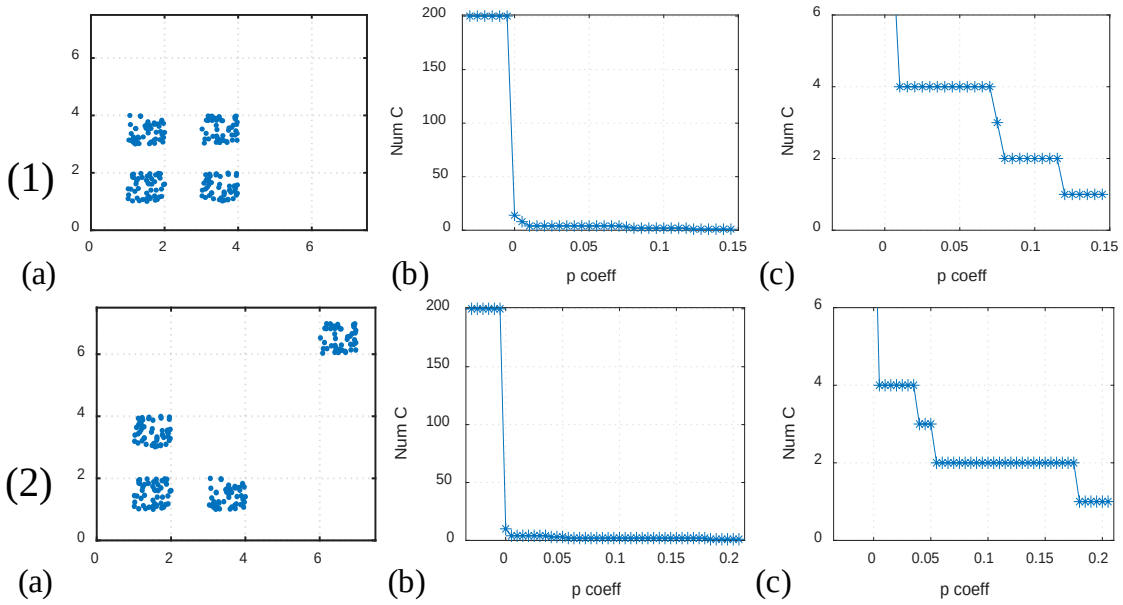


Figure 6.4. Two examples of synthetic data in rows (1) and (2) showing how NumC-AP (Algo. 2) decides the number of clusters. (a) The data distribution in a plane. (b) p^c vs. the number of clusters. (c) Zoomed-in view of (b).

spectral clustering, the number of clusters K is specified by the user. Although a similar parameter (preference p) is specified in AP clustering, the selection of p is more robust than that of K, as p linearly controls the perception granularity. The number of clusters increases with p and depends on the number of input objects [87]. We used p^c (p coefficient) to avoid the dependence on the number of objects:

$$p = \text{mean}(\boldsymbol{S}) - p^c \cdot N \tag{6.7}$$

Algorithm 6.2. Number of Clusters using AP (NumC-AP)

Input: $\boldsymbol{S} = \{s(i,j)\}, p^c_{min}, \gamma$

Output: N^{c*}

Step1. Initialize $u = 1$, $p^c(u) = p^c_{min}$;

Step2. Run AP clustering with $\boldsymbol{S}$ and $p^c(u)$. The output is the number of clusters $N^c(u)$.

Step3. If $N^c(u) > 1$, $u = u + 1$, $p^c(u) = p^c(u-1) + \gamma$, go to Step2.

Step4. **return** the most stable number of clusters $N^{c*} = mode(N^c)$.

where $\boldsymbol{S}$ is the similarity matrix of traces and N is the number of traces. Alg. 6.2 summarizes our approach for selecting the number of clusters using the AP clustering algorithm (NumC-AP). In Alg. 6.2, γ is the increment of p^c and N^c is the number of clusters. We used synthetic data to show how NumC-AP works (Figure 6.4). In the first example, points are distributed into four groups (Figure 6.4 (1.a)). The NumC-AP results show how the number of clusters changes with p^c from N to 1 (Figure 6.4 (1.b)). The proper number of clusters determined by NumC-AP is 4 and the second best choice is 2 clusters (Figure 6.4 (1.c)) as they best reflect the actual distribution of data points (Figure 6.4 (1.a)). Changing the distribution of the synthetic data causes the optimal number of clusters to change accordingly (Figure 6.4 (2)).

6.3.4 Determining the Cluster Prototype

After trace clusters are determined, a step-by-step prototype trace representing the recommended enactment is identified for each cluster. In the past, the medoid or a consensus sequence have been used as process prototypes. Because our traces contain concurrent activities that vary in the order of performance and temporal characteristics, existing methods cannot provide representative prototypes for our application. We developed an approach for determining cluster prototypes in three steps: (1) discovering the time-warped prototype using time warping paired with a divide-and-conquer strategy (a method of dividing the problem into recursively conquerable subproblems used, for example, in Quicksort); (2) unwarping the timeline to find the prototype; and (3) filtering and repairing the prototype for easier interpretation. Given a cluster $\boldsymbol{C}$ of traces, we first build a guide tree t (a dendrogram) using hierarchical clustering with Ward's method linkage criterion [89]. The time-warped cluster prototype $\boldsymbol{q}$ is then solved recursively from

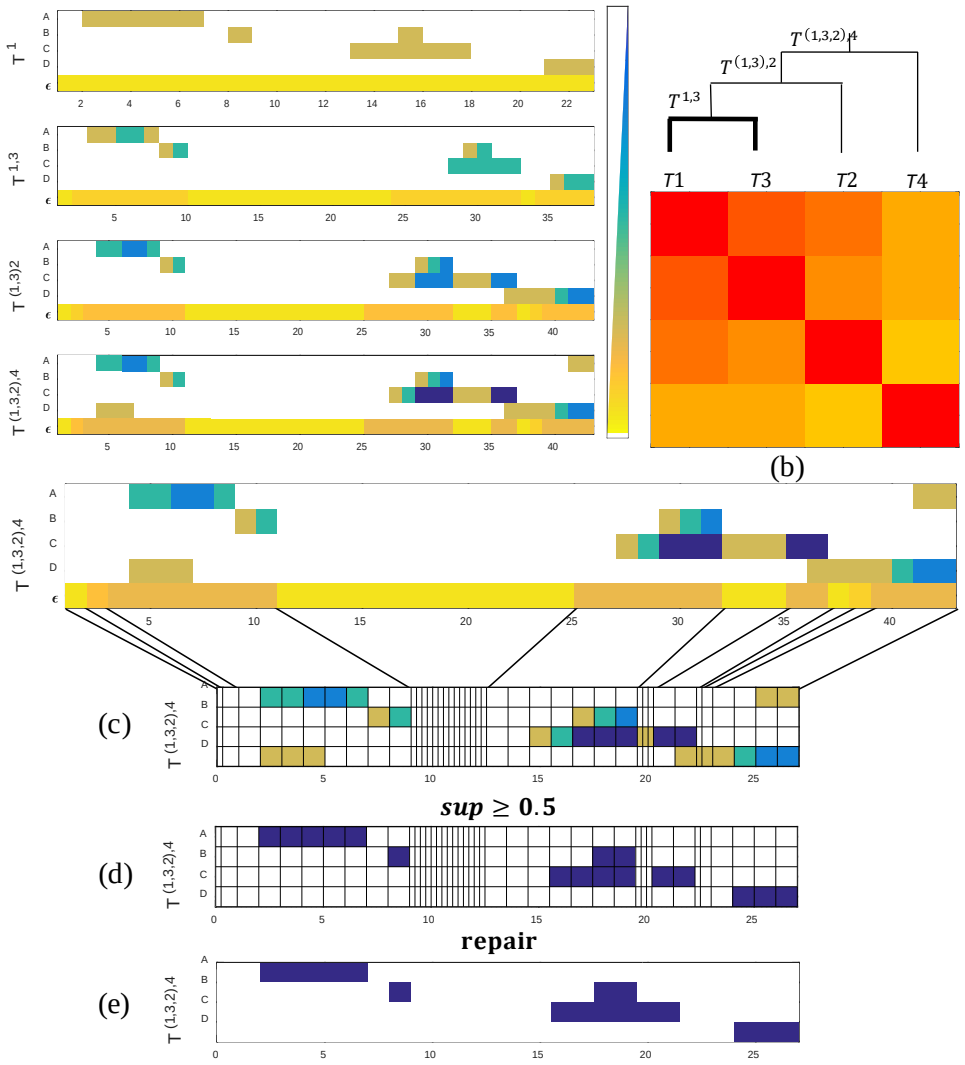


Figure 6.5. Steps for calculating a cluster prototype. (a) Calculating prototype *q* pairwise recursively from a set of process traces. Trace activities are shown in rows. After traces are aligned and activities summed up, the summed value is visualized using the color-bar from 1 to *n*, where *n* is the number of traces. (b) A guide tree for directing the prototype calculation for a cluster of traces. (c) Unwarping the warped timeline to restore the timeline and find the prototype. (d) Filtering the prototype using α. (e) Repairing activity C by merging smaller fragment to the larger one.

the leaves to the root of the guide tree (Figure 6.5 (b)). At each step, ***q*** is calculated pairwise from process traces by summing up their aligned results (Figure 6.5 (a)).

$$\boldsymbol{q} = \boldsymbol{T}^{i,j} = \overline{\boldsymbol{T}_i} + \overline{\boldsymbol{T}_j} = [\boldsymbol{p}_1^i + \boldsymbol{p}_1^j \ \boldsymbol{p}_2^i + \boldsymbol{p}_2^j \ldots \boldsymbol{p}_k^i + \boldsymbol{p}_k^j] \tag{6.8}$$

where $\overline{\boldsymbol{T}_i}$ and $\overline{\boldsymbol{T}_j}$ denote traces aligned using Alg. 6.1 and k is the length of the warped timeline. The time penalty vector $\boldsymbol{\varepsilon}$ is set to $[0.2, \ldots, 0.2]^T$ (bottom rows in Figure 6.5 (a)). The penalties start as equal for the original traces so during alignment $\boldsymbol{\varepsilon}$ can capture whether a warped time unit was frequently aligned or only existed in few cases. The summed $\boldsymbol{\varepsilon}$ in $\boldsymbol{q}$ in the root of guide tree t can guide the time unwarping by its values in each time unit (Figure 6.5 (c)). For example, the long yellow bar in the bottom row of Figure 6.5 (a), between time 10 and 25 in $\boldsymbol{T}^{(1,3,2),4}$ comes from trace $\boldsymbol{T}_3$ which has a long idle period in the middle. For easier interpretation, we simplify $\boldsymbol{q}$ by thresholding out the rare activities (Figure 6.5 (d)). To this aim, we define the *support* of a time cell $a_{ij}^{\boldsymbol{q}}$ as:

$$sup = a_{ij}^{\boldsymbol{q}}/c \tag{6.9}$$

where c is the number of traces in the cluster; i is the ith row (also ith activity type) of $\boldsymbol{q}$; j is the jth column (also jth time unit) of $\boldsymbol{q}$. The time unit is set to 1 when its support is greater than a threshold α and 0 otherwise, where α is by default set to 0.5. A potential drawback of this thresholding strategy is that it cannot capture frequent but sparsely distributed activities. To address this problem, we estimated the activity's frequency and included frequent activities (unique freq ≥ 0.5 in the clusters' cases) back into the prototype at the most likely position. This adjustment was done because the sparsely distributed rare activities may be aligned to several different positions during the prototyping. The thresholding removed them from consideration during warping, and left them to reincorporate more appropriately later. Another problem is, as time units are independent and discrete, activity-time cells of an activity may be fragmented after alignment and filtering (e.g., activity *C* in Figure 6.5 (d)). This fragmentation occurs because the time axis is discretized, a continuous activity is sliced into discrete slices and each slice is aligned independently with the corresponding time slice in other traces. When the slices of a continuous activity are independently aligned with other traces, the alignment may introduce gaps between the slices (e.g., activity *C* in in Figure 6.5 (d)) because in another trace the same activity was performed with an interruption or because a concurrent activity forced this fragmentation to achieve higher similarity score. We apply a repair to mitigate

Algorithm 6.3. Time-warping based Cluster Prototype (TwCP)

Input: $\boldsymbol{C}$, α, β
Output: $\boldsymbol{q}$

Step1. Calculate similarity matrix $\boldsymbol{S}$ of $\boldsymbol{C}$ using Alg. 6.1 (TwS-PT);

Step2. Build the guide tree t with HC algorithm and $\boldsymbol{S}$;

Step3. Traverse t bottom up, from leaves to the root;

Step4. $\boldsymbol{T}_i$= node.get(left), $\boldsymbol{T}_j$= node.get(right), align $\boldsymbol{T}_i$ and $\boldsymbol{T}_i$;

$$\boldsymbol{q} = \boldsymbol{T}^{i,j} = \overline{\boldsymbol{T}_i} + \overline{\boldsymbol{T}_j};$$

Step5. Go to Step 3 until current node equals root;

Step6. Unwarp $\boldsymbol{q}$ to recover the timeline;

Step7. Filter $\boldsymbol{q}$ with a predefined α and repair $\boldsymbol{q}$ with β;

Step8. **return** $\boldsymbol{q}$

this problem by moving the smaller fragment to merge with the large one and close the gap if the gap is smaller than a time threshold β, which can be set as the mean value of all activity durations. We move the smaller fragment to the larger one since this repair has a smaller cost. Our procedure for extracting cluster prototype is summarized as Alg. 6.3.

6.3.5 The Recommender Model

We chose to use regression model for our recommender system rather than a complex model (e.g., SVM or neural networks), as the statistical analysis (e.g., significance test) in regression model can help us easier interpret the correlations between data cluster membership and context attributes.

The goal of our logistic regression model is to leverage a set of n process cases to design a classifier that can distinguish between $m \geq 2$ clusters given context attributes $\boldsymbol{x}$. The cluster label of a process trace is encoded as $\boldsymbol{y} = [y^{(1)}, y^{(2)}, \ldots, y^{(m)}]^T$ where $y^{(i)} = 1$ if $\boldsymbol{x}$ is the context information of a trace that belongs to cluster i and $y^{(i)} = 0$ otherwise. The n process cases can then be represented as $\mathcal{S} = \{(\boldsymbol{x}_1, \boldsymbol{y}_1), \ldots, (\boldsymbol{x}_n, \boldsymbol{y}_n)\}$. By default, we define the last class (the mth cluster) as the reference category, against which logits of the first $m - 1$ categories are compared. Our logistic regression was trained with L2 regularizer:

$$\widehat{\boldsymbol{\beta}} = \arg\max_{\boldsymbol{\beta}} \left[\sum_{j=1}^{n} \log P(\boldsymbol{y}_j | \boldsymbol{x}_j, \boldsymbol{\beta}) + \lambda \|\boldsymbol{\beta}\|^2 \right] \tag{6.10}$$

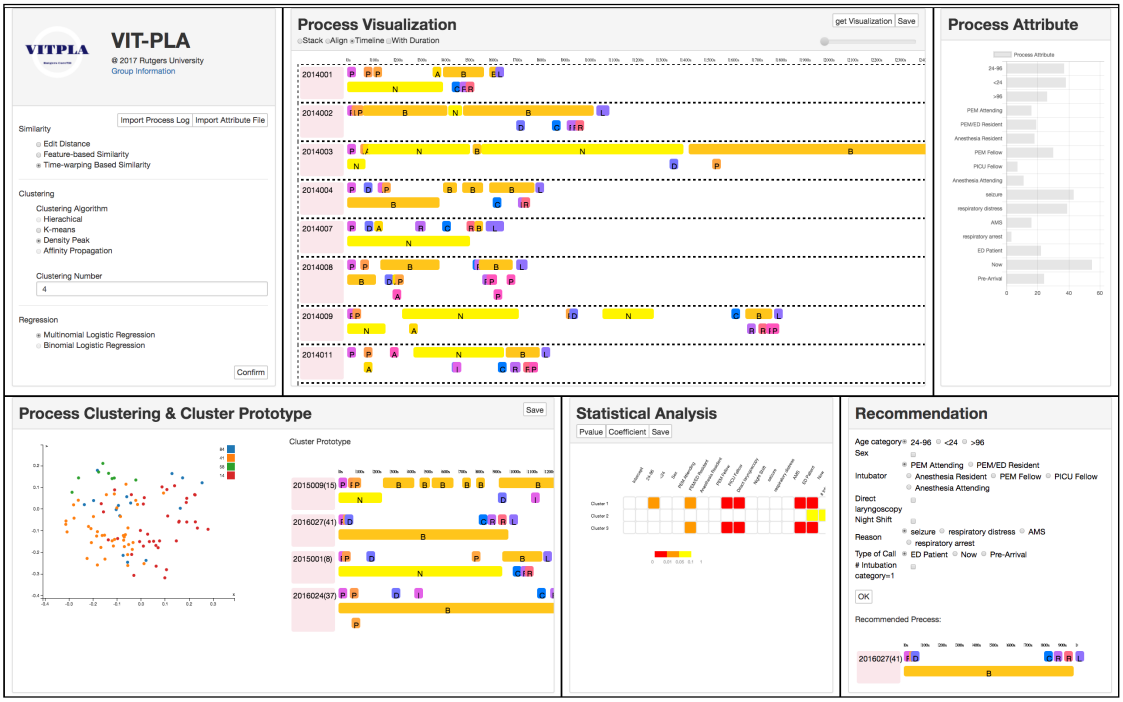


Figure 6.6. Graphical user interface of the implementation of our framework for process mining and recommendation.

where $\boldsymbol{\beta}$ are regression coefficients for context attributes and λ is the ridge estimator of L2 regularizer. To find which attributes are associated with cluster membership, we used the Wald test [90] for logistic regression and a significance level at <0.05.

To generate recommendations, our system works by taking a new context attribute set $\boldsymbol{x}'$ (given by the user) and outputs a recommended enactment. The trained regression model selects the cluster class label $\boldsymbol{y}$ that maximizes the likelihood function:

$$\boldsymbol{y} = \arg\max_{\boldsymbol{y}} P\left(\boldsymbol{y}|\boldsymbol{x}', \widehat{\boldsymbol{\beta}}\right) \tag{6.11}$$

Our system then returns the prototype of the most probable cluster as the recommended enactment. Because not all contextual attributes are good predictors, we used only statistically significant attributes to improve the recommendation accuracy. If no attribute was found as significant, all attributes are considered. Our framework was implemented as a web app (VIT-PLA, Figure 6.6) using D3.js, Bootstrap, JSP, Java, and includes interactive visual functions.

6.4 Experiments

We demonstrated the use of our framework with three real-world logs and evaluated the performance of different techniques.

6.4.1 Real World Medical Process Datasets

Datasets from three medical processes, collected in the emergency department of Children's National Medical Center, a level 1 pediatric trauma center in Washington, DC, were used for evaluating our framework (Table 6-1):

Tracheal Intubation Data: Ten context attributes are of three types: (a) patient demographics: age (<24 months, 24-96, >96), gender, height, weight, body mass index (BMI); (b) provider attributes: intubator's medical role (emergency medicine attending, anesthesia resident, etc.); and (c) event attributes: night/day, emergency/pre-arrival, direct-laryngoscopy/video-laryngoscopy and reason for intubation (seizure, respiratory distress, altered mental status—AMS).

Trauma Resuscitation Data: The trauma resuscitation is performed by a trauma team comprised of several physicians, nurses and ancillary medical staff, all working concurrently. Each case was coded with 17 context attributes of two types: (a) patient demographics: age, race, gender, injury type, injury severity score, pre-arrival intubation, mental status, body region injured (e.g., head, face, chest, etc.); and (b) treatment attributes: paged response (stat, transfer), day/night, weekend/weekday.

Emergency Department (ED) Data: This dataset contained a very diverse set of patient procedures. The attribute types are the same as for the trauma resuscitation data. Unlike tracheal intubation and trauma resuscitation, which are standardized processes, the ED process is not. ED data is quite different from case to case and the activities are temporally sparse.

6.4.2 Similarity Measure Evaluation

To evaluate our similarity measure, we performed experiments using 65 randomly selected sets of three traces from the Intubation dataset $\{\boldsymbol{T}_i, \boldsymbol{T}_j, \boldsymbol{T}_k\}$ (Figure 6.7(a)). Three medical experts were asked to decide the most similar among three trace pairs, $(\boldsymbol{T}_i, \boldsymbol{T}_j)$, $(\boldsymbol{T}_i, \boldsymbol{T}_k)$ and

Table 6-1. Properties of our three medical process datasets.

Stats \ Dataset	Intubation	Trauma	ED
Num. Patient Records	101	87	644
Num. Total Acts	1244	9477	2290
Num. Act Types	15	128	65
Longest Trace (Num. Acts)	20	196	12
Shortest Trace (Num. Acts)	8	60	1
Num. External Attributes	10	11	11

($\boldsymbol{T}_j$, $\boldsymbol{T}_k$) based on their domain knowledge. Our visualizations (Figure 6.7(a)) of traces helped them to quickly detect the differences between traces in a set. They used their domain knowledge to judge how important these differences are, and decide which trace pair is more similar than others. We used these labeled results to evaluate our similarity measure. Our baselines included edit distance (ED), sequential-pattern based distance (SP based on algorithm CM-SPADE [92]), normalized Euclidean-distance (NE), and dynamic time warping distance (DTW). We also evaluated these similarity measures using a majority voting strategy that determines whether the most similar pair selected by each measure matched the majority decision.

The results (Figure 6.7(b)) showed that our time-warping-based similarity measure achieved the highest accuracy on both medical expert labels (0.69) and voting-based results (0.80). Edit distance, the simplest measure considered, also performed well because the intubation data was mostly sequential so the activity type and order of performance were the keys to comparing the traces. Normalized Euclidean distance and DTW performed worse because they failed in cases where a long intubation trace (e.g., 40 mins) was compared with a brief trace (e.g., 10 mins). The normalized Euclidean distance failed because it could only capture few similarities after normalizing long and short timelines. The DTW failed because it did not penalize long idle times and activity duration differences between traces. In addition to individual measures, we also computed the accuracy of the majority. The majority of our similarly measures correctly identified 39 sets (≥3 votes) and 5 sets as unsure (with two tied majorities).

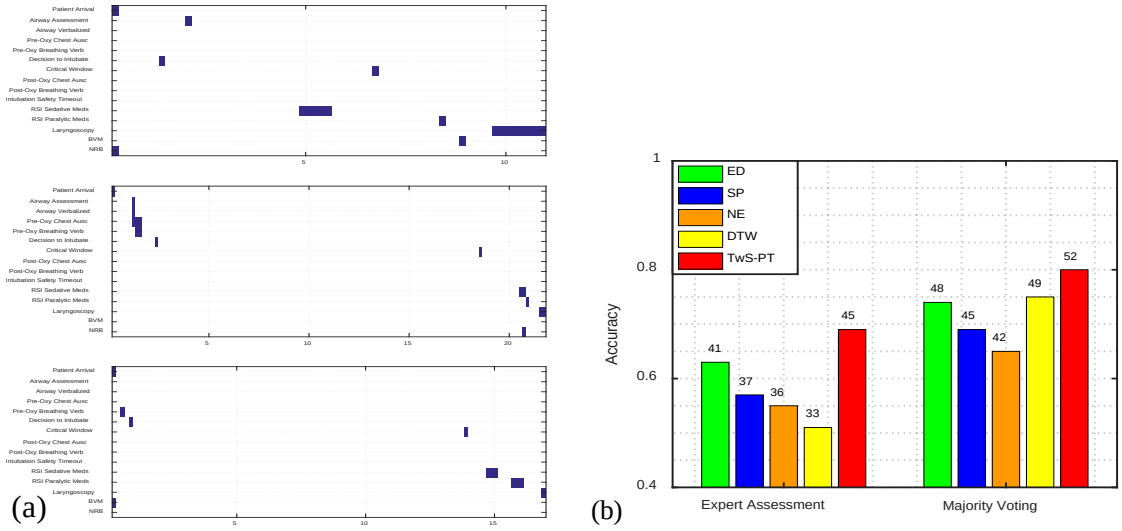


Figure 6.7. (a) A sample set of Intubation procedure given to medical experts to evaluate. The horizontal-axis denotes timestamp in minutes and vertical-axis denotes activity types. The blue blocks represent the performance time and duration of activities. (b) Performance of different similarity measures compared to expert opinion.

In 12 of the 65 sets, all measures and the medical experts agreed on the most similar trace pairs. In another 5 cases, all measures made wrong choices. We reanalyzed these 5 cases and found that the ground truth was incorrectly labelled in two, and in the other three cases the experts used medical knowledge that was not explicitly considered by the similarity measures: (1) time-to-task for “decision to intubate,” and (2) the type of oxygen mask (BVM vs. NRB). Even without additional domain knowledge, we found that in 62 of 65 cases (95.4%), at least one data-driven similarity measure made the same decision as the experts did. Our TwS-PT (Alg. 6.1) independently achieved 69% decision accuracy. These two findings show the feasibility of using purely data-driven similarity measures for comparing complex process traces.

6.4.3 Prototype Analysis

We evaluated our prototype extraction method (TwCP, Alg. 6.3) quantitatively and by qualitative feedback from domain experts. We used mediod as the benchmark prototype since it is often used as cluster exemplar. For this comparison, we extracted the prototypes

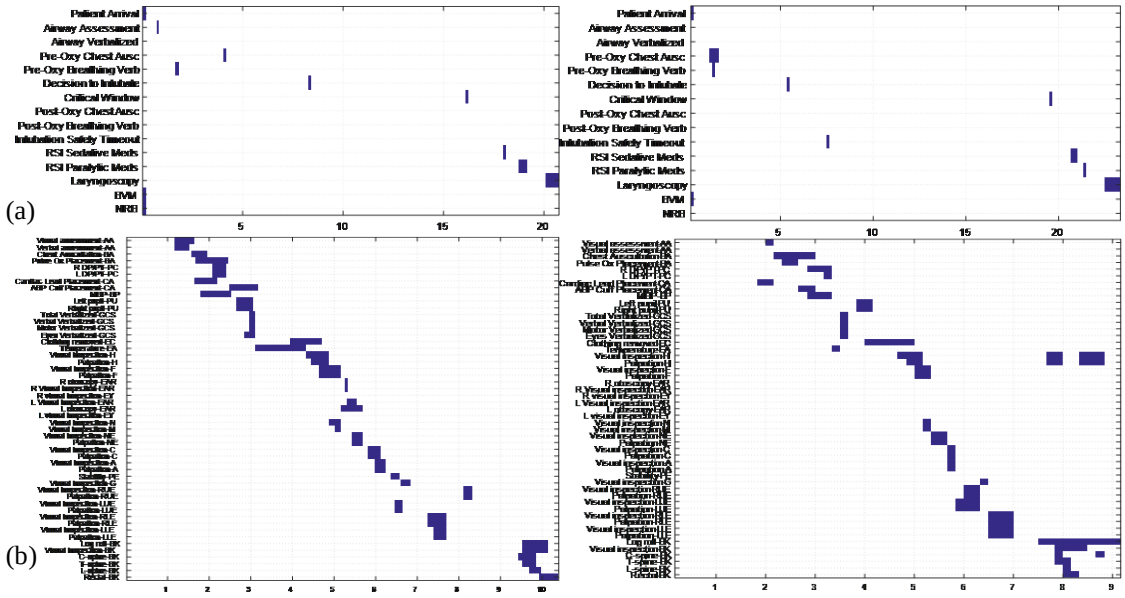


Figure 6.8. (a) TwCP prototype (left) and medoid (right) for the whole Intubation dataset. (b) TwCP and medoid for Trauma dataset showing the 52 commonly performed activities. For easier comparison, the vertical axis labels (activity names) were ordered based on a rough temporal order of activities. The horizontal axis denotes the real (not warped) timeline in minutes.

and medoids from the whole datasets without clustering, to avoid potential bias from clustering algorithms (Figure 6.8). We omitted the ED dataset from this comparison because its prototype and medoid had only two activities. Our results show TwCP prototype had higher average similarity to other traces than the medoid (Figure 6.9(a)). This difference was greater for the trauma dataset than for the intubation dataset because the medoid depends on dataset size (number of traces) and trace complexity. A large dataset is more likely to contain a trace close to the centroid. In a small dataset, the medoid may be far from the centroid. Process complexity also affects the medoid because more activities and greater variability make it less likely that an existing trace will well represent the characteristics of the process. Our intubation data is much simpler than trauma data that had more than 100 activity types and average trace length of 109 activities.

The medoids may not fully capture deviations from the standard protocol due to the variable injuries of different patients. Our TwCP prototype better captured standard practices and included more tasks applicable to a diverse range of injuries, but it may

capture idiosyncratic details that would not be expected by a domain expert. For the Trauma data (Figure 6.8 (b)), the medoid omitted inspection of the eyes, nose and pelvis while the TwCP suggested an acceptable but uncommon sequence for the extremity exam. For the Intubation data (Figure 6.8 (a)), TwCP included the performance of airway assessment and the use of the non-rebreather (NRB), which the medoid omitted. The medoid more accurately represented oxygen delivery during intubation because one cannot use a bag valve mask (BVM) and NRB simultaneously. TwCP, however, showed that both mechanisms of oxygen delivery were acceptable before intubation and included airway assessment, making the prototype more complete. The human factors literature [93] and our study [11] suggests to study work-as-done rather than work-as-imagined when designing computerized support systems. TwCP prototype is useful since it captures actual work. By comparing a given trace to the prototype, one can detect and analyze the process deviations.

6.4.4 Recommendation System Evaluation

Our recommendation system was evaluated using two approaches: (E1) whether the actual process trace (denoted as $\boldsymbol{T}_a$) belonged to the most probable cluster decided based on context attributes by the trained regression model; and (E2) whether the recommended trace (denoted as $\boldsymbol{T}_r$) was close to the actual trace.

Because trace clusters may be of very different sizes (multi-class imbalance learning problem), we adopted the F-measure (F_1-score) and geometric mean (G-mean) [91] to properly evaluate the performance using the first approach (E1). We did not choose the commonly used accuracy measure as it is ineffective at evaluating imbalanced learning scenarios, where the accuracy of the majority class may dominate. F-measure and G-mean can balance the classification performances of all majority and minority classes.

The second approach (E2) evaluated our system by checking if the recommended trace $\boldsymbol{T}_r$ was among the k nearest neighbors of $\boldsymbol{T}_a$, where k ranged from 1 to n and for k=1 the recommended prototype was the closest neighbor. This measure is not symmetrical, i.e., $\boldsymbol{T}_r$ being within k neighbors of $\boldsymbol{T}_a$ does not imply that $\boldsymbol{T}_a$ is within k neighbors of $\boldsymbol{T}_r$. Therefore, a recommended trace that is among a few neighbors of most traces is very representative for the given cluster.

We implemented several similarity measures: edit distance (ED), sequential pattern (SP), and TwS-TP, and several clustering algorithms: hierarchical clustering (HC), density-peak clustering (DPC) and affinity propagation clustering (APC). We clustered the process traces using different combinations of similarity and clustering algorithms. We used tenfold cross-validation to reduce the variance of the recommendation accuracy. We selected ZeroR as the baseline, which always takes the largest cluster as the prediction result. Our experimental results (Table 6-2) show that the combination of time-warping distance and APC algorithm achieved the highest F_1 score for both the Intubation and Trauma data. Edit distance with APC algorithm achieved the highest F_1 score for the ED procedure data. From the perspective of the clustering algorithm, APC performed better than HC and DPC in most cases regardless of the similarity measure. From the perspective of the similarity measure, our TwS-PT performed best for both the Intubation and Trauma data. Edit distance performed best for ED data (Table 6-2), because ED procedures are sparse with only few activities and temporal information is not essential. Temporal information is informative and important for some but not all processes. The selection of similarity measure is best decided by the nature of dataset with the help of visualization tools. Medical procedures depend on other factors that were not recorded in our data, such

Table 6-2. Recommendation evaluation on three medical process datasets. The format $\boldsymbol{\alpha}$ ($\boldsymbol{\tau}$) represents the regression model result $\boldsymbol{\alpha}$ and the baseline (ZeroR) result ($\boldsymbol{\tau}$). Rec NC stands for recommended number of clusters.

	Intubation Data		Trauma Resuscitation Data		ED Procedure Data	
Rec NC	ED (2), SP (3), Time-warping (2)		ED(2), SP (2), Time-warping (2)		ED (2), SP(3), Time-warping (2)	
Metrics	F-Score	G-means	F-Score	G-means	F-Score	G-means
ED + HC	0.505 (0.504)	0.445 (0.479)	0.634 (0.654)	0.448 (0.428)	0.615 (0.615)	0.445 (0.445)
ED + DPC	0.719 (0.755)	0.383 (0.374)	0.692 (0.686)	0.436 (0.413)	0.860 (0.860)	0.293 (0.293)
ED + APC	0.415 (0.339)	0.416 (0.500)	0.346 (0.353)	0.392 (0.500)	0.595 (0.447)	0.571 (0.491)
SP + HC	0.286 (0.275)	0.412 (0.497)	0.637 (0.533)	0.603 (0.471)	0.395 (0.292)	0.531 (0.499)
SP + DPC	0.446 (0.264)	0.566 (0.496)	0.637 (0.533)	0.603 (0.471)	0.516 (0.516)	0.476 (0.476)
SP + APC	0.487 (0.277)	0.593 (0.471)	0.645 (0.519)	0.591 (0.475)	0.485 (0.477)	0.485 (0.494)
TwS-PT + HC	0.596 (0.419)	0.590 (0.495)	0.526 (0.392)	0.520 (0.497)	0.502 (0.395)	0.554 (0.497)
TwS-PT + DPC	0.605 (0.567)	0.494 (0.461)	0.713 (0.670)	0.556 (0.421)	0.531 (0.387)	0.538 (0.498)
TwS-PT + APC	0.700 (0.384)	0.695 (0.498)	0.767 (0.366)	0.683 (0.499)	0.581 (0.471)	0.549 (0.486)

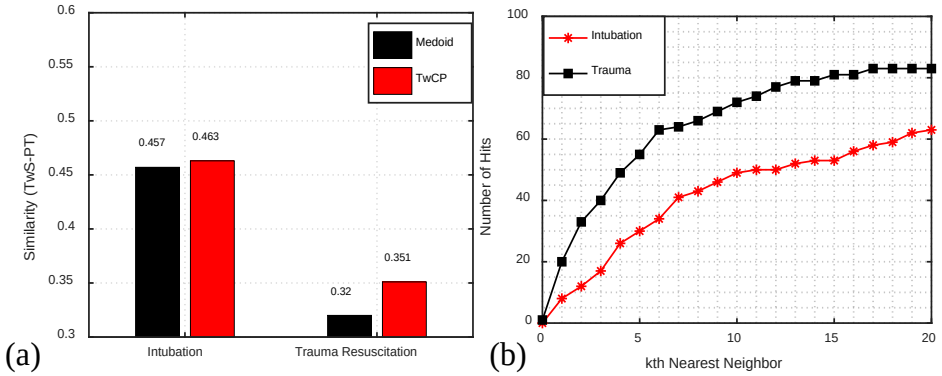


Figure 6.9. (a) Avg. similarity between prototypes and other process traces. (b) Number of hits of recommended process enactment within k nearest neighbors of the actual enactment.

as the environment, patient condition, and medical team status. This fact explains why we could not achieve very high recommendation accuracy for these complex datasets. An alternative is making recommendations only for a subset of cases when regression model has a high confidence. For example, when we made prediction only for patients whose intubation reason was altered mental status (AMS) and type of call was "now," we achieved 87.5% recommendation accuracy using the TwS-PT + APC combination.

In 55 of 87 cases (63.2%) in the Trauma dataset, our recommended prototype was among the 5 nearest neighbors of the actual trace (Figure 6.9 (b)). In the remaining 32 cases, the recommended prototype was not among the 5 nearest neighbors of the actual trace because our regression model incorrectly predicted the cluster membership from trace's context. For example, TwS-PT+APC had 0.767 F_1 score for finding cluster membership using context attributes for trauma data (Table 6-2). A wrong cluster, in turn, results in recommending a wrong prototype.

6.4.5 A Case Study with Intubation Process

We used the Intubation dataset as a case study to further illustrate the performance of our framework. The recommended number of clusters given by NumC-AP (Alg. 6.2) was 2 (Figure 6.10 (a)(b)). The process traces were clustered using algorithms HC (Figure 6.10 (c)), DPC, and APC. In the trained regression model, several context attributes, e.g.,

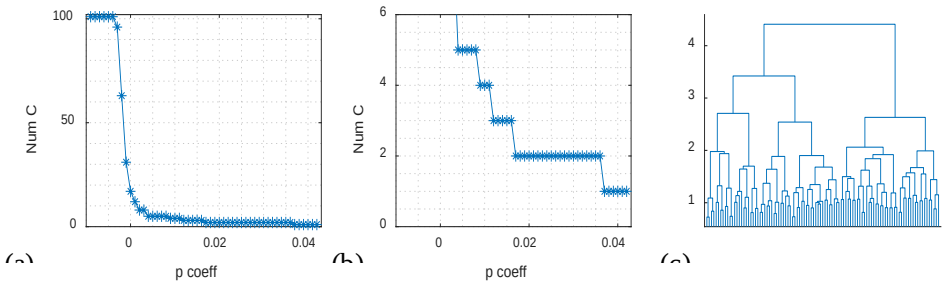


Figure 6.10. (a)(b) NumC-AP (Alg. 6.2) on Intubation data and (c) hierarchical clustering (based on Ward's method).

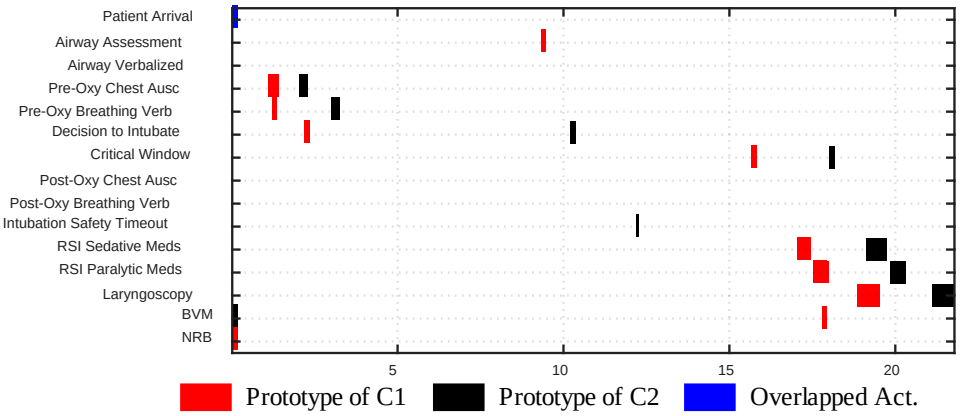


Figure 6.11. Prototypes of cluster-1 (q^{c1}) and cluster-2 (q^{c2}).

intubator role, night shift, intubation reasons, were statistically significant for trace clusters (Table 6-3). Using the APC result as an example, the reason for intubation and intubator role was significantly correlated with the two clusters. The two prototypes ($\boldsymbol{q}^{c1}$ and $\boldsymbol{q}^{c2}$) (Figure 6.11) extracted from two clusters showed many differences: (1) $\boldsymbol{q}^{c1}$ (Figure 6.11 (a)) had the activities "airway assessment" and "NRB," while in $\boldsymbol{q}^{c2}$ (Figure 6.11 (b)) these activities were missing; (2) $\boldsymbol{q}^{c1}$ (~19.5 mins) was shorter than $\boldsymbol{q}^{c2}$ (~22 mins); (3) activities "pre-oxy breathing verb." and "decision to intubate" occurred later in $\boldsymbol{q}^{c2}$. In addition to these differences, $\boldsymbol{q}^{c1}$ and $\boldsymbol{q}^{c2}$ had many commonalities, e.g., performance time and sequential order of activities "chest auscultation," "critical window," "RSIs" and "laryngoscopy." Our medical experts explained that in cluster-1 clinicians used a passive non-rebreather (NRB) instead of an active bag-valve-mask (BVM) for initial oxygen delivery. The ATLS protocol requires that providers secure the patient's airway before

Table 6-3. *p*-values from regression model.

Attributes \ Clustering		HC	DPC	APC
(Intercept)		0.43	0.73	**0.03**
Age	<24 months	0.43	**0.07**	0.11
	24-96 months	0.75	0.39	0.94
Gender		0.76	**0.1**	0.34
Intubator Role	Anesthesia Resident	0.41	0.31	0.2
	PEM Attending	0.58	0.85	**0.03**
	PEM Fellow	0.17	0.2	0.25
	PEM/ED Resident	0.64	**0.09**	0.79
	PICU Fellow	0.43	0.74	0.36
Direct Laryngoscopy		0.77	0.11	0.4
Night Shift		0.18	**0.03**	0.51
Reason	Respiratory Distress	0.15	0.87	**0.02**
	Seizure	0.74	0.94	0.56
Type of Call	ED Patient	0.85	0.53	0.14
	Now	0.79	0.34	0.57

moving onto other survey items. Our results showed that patients in cluster-1 more frequently underwent intubation for respiratory distress. If patients in cluster-1 originally presented with a secured airway, it would make sense that the onset of respiratory distress would necessitate intubation to secure the airway. Patients in cluster-2 were already experiencing some degree of respiratory distress or they would not have needed a BVM. It is plausible, then, that other clinical indicators prompted intubation in cluster-2.

Chapter 7 A Context-Aware Deep Learning Framework for Next Medical Treatment Activity Recommendation

This chapter is about the Deep Process Recommender System. AI recommender systems have been successfully applied in many domains (e.g., e-commerce, e-learning). It is believed by medical experts that such systems may help reduce medical team errors and improve patient outcomes in treatment processes (e.g., trauma resuscitation, surgical processes). Limited research, however, has been done to explore automatic data-driven treatment recommendations. In this chapter, we bridge this gap by presenting a deep-learning-based process recommender system to provide runtime treatment recommendations. The system is built on state-of-the-art recurrent neural networks, which take into account both environmental (e.g., patient demographics) and behavioral (i.e., preceding treatment activities) contextual information. In our implementation, we presented novel designs like Act2vec and sliding-window attention to improve the model performance and help interpret the results. We also proposed a data augmentation algorithm to address the limited amount of data and help pre-train the model. Our framework was evaluated on two real-world medical process datasets. The experimental results show our system outperforms baseline methods in recommendation accuracy, demonstrating the feasibility of data-driven context-aware recommender systems for complex real-world medical processes.

7.1 Introduction

Medical teams make unavoidable errors in fast-paced and high-risk medical treatment processes. Take trauma resuscitations for example. Critically injured trauma patients have up to a four-fold higher risk of death from errors than general hospital patients. Nearly half of these preventable deaths are related to errors during the initial resuscitation phase of treatment [94]. During such medical processes, multidisciplinary teams are responsible for rapidly identifying and treating potentially life-threatening injuries, then developing and executing a short-term management plan for those injuries. Some computer-aided decision

support systems [47] and expert-derived algorithms [95] have been proposed to reduce medical team errors and improve patient outcomes for treatment processes. Despite being carefully designed by medical experts, these initial attempts have had limited success for three reasons. First, the expert-derived knowledge-based models might not reflect reality. Second, the expert model's rules are meant for general patients and do not take into account the specific needs of a particular patient. Third, these approaches are heavily case-based and lack generalizability to other medical processes. Our research explores how to automatically provide data-driven recommendations to accompany the clinicians' decision process. The recommender system built on artificial intelligence (AI) and data mining techniques would provide the medical team leader (or surgical coordinator) with next-step treatment recommendations through the wall displays (monitors).

Despite the wide application of data-driven recommender systems in e-commerce and e-service [96], there are only a limited number of applications in health or medical related fields [97][98]. There is even less related work for temporally correlated data. Sun et al. [99] proposed a similarity-based framework to extract typical treatment regimens from large-scale electronic medical records; they used these to match the discovered treatment regimens with patient cohorts for personalized medication recommendation. However, their work recommends a whole treatment regimen to patients, while this chapter studies how to recommend next activities dynamically. Yang et al. [3] clustered patients into cohorts to find prototypical treatment patterns for each cohort. The prototypical treatment patterns were recommended to the new patients by first deciding which cohort the patient belonged to. This work only considered static contextual information (patient demographics), and did not consider the dynamic contextual information (e.g., ongoing treatment process). The recommendations were given before the treatment started and would not adjust during the treatment. Edward et al. [98] studied how to predict the next clinical event by considering both environmental and behavioral medical information, but their method can only predict the occurrences of three main endpoints and takes the advantage of sufficiently available electronic medical records.

Our recommender system application for medical processes has two characteristics. Firstly, temporal information plays an important role. Treatment activities have temporal correlations, i.e., the secondary survey of the trauma resuscitation usually follows a head-

to-toe examination. Temporal correlations do not only exist between directly adjacent activities, but also exist between activities separated by several intervening activities. Secondly, the medical team must take into account that different patients with different conditions (e.g., injury area and severity) need different medical treatment procedures. For this reason, it is important to incorporate context attributes into the recommender system for prescriptive analytics [100]. There are two types of contextual information: environmental and behavioral. Behavioral context refers to the treatment workflow: the activities performed and the order of their performance. Environmental context can be further divided into two categories: static and dynamic. Static context is features of the patient or resuscitation that are present when the patient arrives and do not change. Examples are time of day, age of patient, and mechanism of injury. Dynamic context is features that change as treatment goes on; these are usually activity attributes (e.g. descriptor and whether the activity was completed).

Our recommender system was specifically designed to handle these differences. To model temporal dependencies, we used recurrent neural networks (RNN). To address the second challenge of patient diversity, we modified the RNN to receive and incorporate patient demographics as auxiliary inputs to the network. In addition, we included two novel designs, Act2vec and sliding-window attention, to improve RNN performance and interpretability. Another technical challenge in our study is the limited amount of medical process data. The proper learning of the complex temporal correlations requires a sizable amount of training data. Coding medical process data, however, is labor-intensive. Over two years, we coded 122 resuscitation cases, but our data is still too small to train a deep neural network. We address this limitation by pre-training the neural network with synthetic data. The synthetic process data was generated by a novel data-augmentation algorithm that is based on trace alignment algorithm and multivariate Bernoulli distribution. Our main contributions are:

- *A novel process recommender framework* using a multi-input recurrent neural network that integrates both environmental and behavioral context information. We applied our approach to the trauma resuscitation process, and the results show that we achieved better recommendation accuracy than the baselines.

- *Act2vec* activity embedder designed to translate treatment activities into numerical vectors. Act2vec can capture the proximity between different activities. The use of Act2vec in the model helps improve model performance. And the visual analytics based on Act2vec can reveal interesting insights from medical processes.
- *A sliding-window attention* designed to improve model performance and interpretability. The sliding-window attention mechanism can focus RNN on outputs of the last several hidden states, rather than just a single hidden state (i.e., the current hidden state). This mechanism can focus the network on states produced many time steps earlier, and does not require that the last state store all the information. By checking the attention score vector, it is also possible for us to know what information the neural network focuses on when making decisions.
- *A data-augmentation algorithm* to help pre-train and regularize neural networks. The algorithm can procedurally fabricate a large amount of unique synthetic patient data that closely resembles authentic data. Generated synthetic data was used to pre-train the neural network, addressing the problem of limited training data.

7.2 Treatment Recommendation with Deep Learning

7.2.1 Data Description and Notations

The historic patient records $\boldsymbol{r} = [r^{(1)}, \dots, r^{(n)}]^T$ is a vector of n elements $r^{(i)}$, where i here is the index of the patient record. Each patient record $r^{(i)} = \{id^{(i)}, \boldsymbol{x}^{(i)}, \boldsymbol{T}^{(i)}\}$ is indexed with a unique patient id, contains the medical process trace $\boldsymbol{T}^{(i)}$, and has a vector $\boldsymbol{x}^{(i)}$ of associated patient attributes. A trace $\boldsymbol{T}^{(i)} = [\left(a_1^{(i)}, \boldsymbol{b}_1^{(i)}\right), \dots, \left(a_{\mathrm{t}}^{(i)}, \boldsymbol{b}_{\mathrm{t}}^{(i)}\right) \dots, \left(a_{\mathrm{q}}^{(i)}, \boldsymbol{b}_{\mathrm{q}}^{(i)}\right)]^T$ includes q treatment activities that are ordered based on activity occurrence time. $\boldsymbol{b}_{\mathrm{t}}^{(i)} = [b_1^{(i)}, \dots, b_m^{(i)}]^T$ is a vector of attributes associated with the activity $a_{\mathrm{t}}^{(i)}$ (e.g., who is the activity "Descriptor" and whether the activity is "Verbalized" in Table 7-1 (a)). Patient attributes $\boldsymbol{x}^{(i)} = [x_1^{(i)}, \dots, x_g^{(i)}]^T$ is a vector of g recorded attributes (e.g., patient age, injury type and injured body area). Attribute vector $\boldsymbol{x}$ is at patient level while attribute vector $\boldsymbol{b}$ is at activity level.

Table 7-1. A data sample of medical process data

ID	Activity	Start Time	End Time	Descriptor	Verbalized
xx1	Pt arrival	0:00:00	0:00:01		
xx1	Visual assessment-AA	0:00:45	0:00:52	Jr Resident	1
xx1	Chest Auscultation-BA	0:00:55	0:00:58	Jr Resident	0
xx1	Oxygen Preparation	0:01:04	0:01:05	EM Attending	1
xx1	Oxygen-BC	0:01:29	0:01:30	EM Attending	1
xx1	Total Verbalized-GCS	0:01:50	0:01:51	Jr Resident	0
xx1	MBP-BP	0:02:12	0:02:18	Nurse Left	1
xx1	Left pupil-PU	0:02:19	0:02:24	Jr Resident	1
xx1	Right pupil-PU	0:02:24	0:02:25	Jr Resident	1
xx1	Visual inspection-H	0:02:33	0:02:34	Jr Resident	0
xx1	Palpation-H	0:02:33	0:02:37	Jr Resident	0

(a) Trauma resuscitation trace

Attributes	xx1
Age category	24-96
Sex	Male
Night Shift	0
Weekend	0
Pre-arrival Notification	1
Trauma Activation Level	Transfer
Intubation	0
Glasgow Coma Score >13	1
Injury Type	Blunt
Injury Severity Score	5
Neck Injury Severity Score	3

(b) Patient demographics

7.2.2 Context-aware Deep Treatment Recommendation Framework

7.2.2.1 Overview of the Proposed Framework

The goal of the proposed algorithm is to recommend next-step treatment activities to the medical team based on the observed behavioral contextual information (up-to-now treatment traces) and environmental contextual information (i.e., activity attributes and patient attributes). The recommender system (Figure 7.1) built on an RNN. The RNN takes as input the concatenation of the activity embedding vectors $\boldsymbol{v}^a = [\boldsymbol{v}_1^a, \dots, \boldsymbol{v}_t^a]$ (main input) and the activity attribute vectors $\boldsymbol{v}^b = [\boldsymbol{v}_1^b, \dots, \boldsymbol{v}_t^b]$ (auxiliary input, dynamic environmental context). The latent vector outputs from the RNN go through our attention layer and then merged with the patient attribute vector $\boldsymbol{v}^x$ (auxiliary input, static environmental context). For the final output, we used a densely connected layer after the merging layer followed by a top-*k* softmax activation function. The most probable *k* activities will be shown to the medical team as the recommended treatment for the next step (*t+1*). In practice, the dynamic contextual information will be updated by our sensor-based activity recognition system or by the nurse recorder who has access to the computerized decision support system.

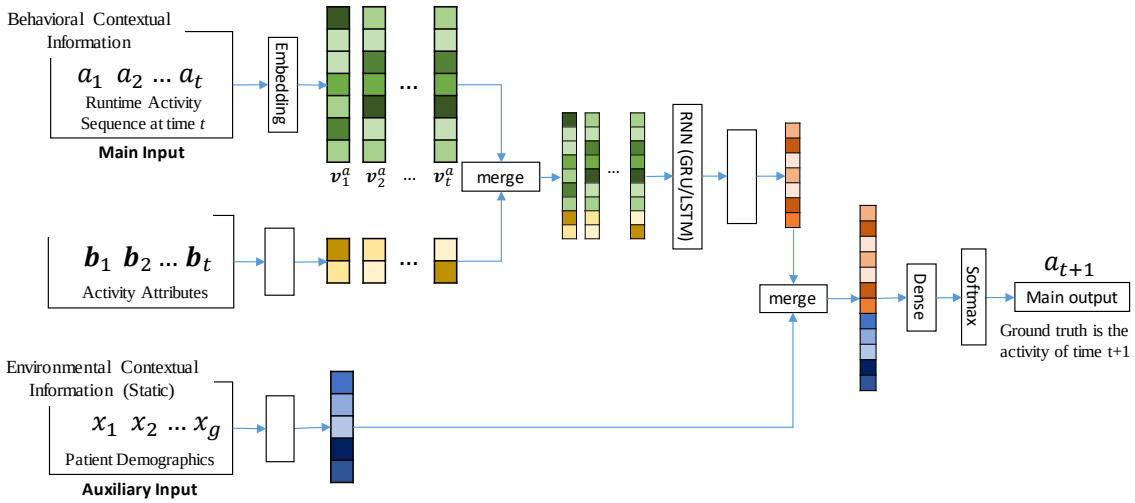


Figure 7.1. Proposed context-aware deep treatment recommendation framework

7.2.2.2 Multiple Contextual Information as Input

Given the treatment activity trace from time 1 to t: $\boldsymbol{T} = [a_1, \dots, a_t]^T$, the i-th activity $a_i \in \boldsymbol{T}$ is embedded into a vector representation $\boldsymbol{v}_i^a$. This behavioral context is the main input to the model. Each activity may also be associated with a set of attributes $\boldsymbol{v}_i^b$, e.g., descriptor and whether the task was completed. As the environmental context is also dynamically changing over time, we merged $\boldsymbol{v}_i^a$ and $\boldsymbol{v}_i^b$ before feeding it to the recurrent neural network. The other auxiliary input, patient attributes $\boldsymbol{v}^x$, is static over time. We thus integrate this information after the RNN step. In addition, according to our domain knowledge, we however know that not all environmental context will contribute to the model performance. Environmental attributes, like patient gender and weight, may have little or no predictive power. Hence, we add a dense layer after the auxiliary input layer to help reduce the feature dimensionality and let the model itself learn the weights of different features.

7.2.2.3 Activity Embedding (Act2vec)

A simple way to format treatment activities is through a one-hot vector. This naïve approach disregards relationships between activity types. In medical processes, treatment activities may have causal relationships and coexist in the same case; some activities are temporally closer than other activities according to treatment protocols. Inspired by

Word2vec [101] in natural language processing, we proposed Act2vec, which embeds the activity types into numerical vectors via a neural net. Act2vec uses skip-grams [101] to maximize the conditional probability of behavioral context (neighboring treatment activities) given an activity, learning a vector representation of each activity type. Act2vec can help reveal the proximity of treatment activities to each other. Compared to the huge word vocabulary in nature language processing domain, the vocabulary of treatment activity types is usually much smaller. The datasets we used in this study have 102 and 15 activity types respectively. The embeddings of the activities are easier to train even with a limited amount of treatment process data.

7.2.2.4 LSTM and GRU

RNNs are powerful at modeling temporal sequences. The standard RNN, however, still suffers vanishing or exploding gradients when learning long-term dependencies. We thus used two RNN variations, Long Short Term Memory (LSTM, [102]) networks and Gated Recurrent Unit (GRU, [103]). The gates in both RNNs are able to decide what information to store and forget. The RNN in our system generates a sequence of hidden state representations $\boldsymbol{h}_1$, …, $\boldsymbol{h}_t$,…, $\boldsymbol{h}_q$, and each state $\boldsymbol{h}_t$ can retain information from previous members of the sequence.

7.2.2.5 Sliding-Window Attention Mechanism

In our problem, the goal is to recommend the next-step activity a_{t+1} according to the treatments that have been done in previous steps, i.e., from a_1 to a_t. Without an attention mechanism, the prediction is made only based on the *t*-th hidden vector output $\boldsymbol{h}_t$, a fixed-length vector. However, medical processes can be very flexible, e.g., some parallel treatment activities may occur in any order. The next-step activity may not be strongly associated with the latest state, but instead be associated with a state produced many time steps earlier. LSTM and GRU, memory based networks, were designed to address this problem. The long and short memories are used to store the information that have been observed. It is just like the human's memory. But because LSTM and GRU flow in one direction and the memorized information diminishes as the networks proceed, this memory based mechanism cannot provide complete context information when making the

predictions at each step. Our sliding-window attention mechanism can supplement the memory based mechanism by providing complete context information within the window. The network will decide what context information to focus on at each step. For example, when people read or review a paper, it is common that the readers look back at the context (previous paragraph or sentences) to understand the current sentence. It is possible but usually difficult to understand every sentence just based on the memory without looking back. We thus incorporated the attention mechanism in LSTM and GRU to help predict a_{t+1}.

Unlike attentions used in machine translation [104][105] or text classification problems [106], where the attentions are applied on the entire input sequence, our attention must span different inputs as the prediction proceeds in the timeline. Hence, we proposed a sliding-window attention. The sliding window attention can guide the RNN to focus on nearby hidden vectors with more predictive power for the next activity. In implementation, we take the hidden vectors $\{\boldsymbol{h}_{t-l+1}, \dots, \boldsymbol{h}_t\}$ from LSTMs or GRUs as input to the attention layer. The aim is then to derive a context vector $\boldsymbol{c}_t$ that captures the information within the window to help predict the current target activity a_{t+1}. We specify l as our sliding window size, and d as the size of the hidden state vector. We implemented three variations of attention. The attention score $\alpha_{t,i}$ (alpha) for each time step i in the sliding window at time step t can be calculated as follows:

$$\alpha_{t,i} = \begin{cases} \boldsymbol{h}_t^T \boldsymbol{W}_\alpha \boldsymbol{h}_i & \text{general} \\ \boldsymbol{v}_\alpha^T \tanh(\boldsymbol{W}'_\alpha [\boldsymbol{h}_t; \boldsymbol{h}_i]) & \text{concat} \\ \boldsymbol{v}_\alpha^T \tanh(\boldsymbol{h}_i) & \text{simple} \end{cases} \tag{7.1}$$

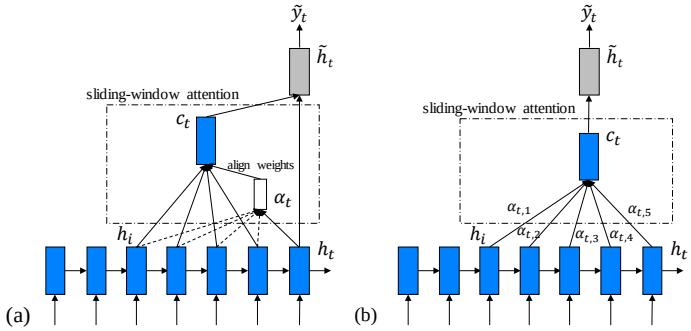


Figure 7.2. Two types of sliding-window attention architecture. (a) Alignment-based attention mechanism. (b) Simple-weight attention mechanism. The window size illustrated here is 5.

where $\boldsymbol{W}_\alpha \in \mathbb{R}^{d\times d}, \boldsymbol{W}'_\alpha \in \mathbb{R}^{d\times 2d}, \boldsymbol{h}_i \in \mathbb{R}^d,\ \boldsymbol{v}_\alpha \in \mathbb{R}^d$. The first two are alignment-based attentions (Figure 7.2 (a)) , "general" and "concat" [104], that are widely used in machine translation. The score vector $\boldsymbol{\alpha}_i$ is derived by comparing the target hidden vector state h_t with the source hidden vectors $\boldsymbol{h}_i \in \{\boldsymbol{h}_{t-l+1}, \dots, \boldsymbol{h}_{t-1}\}$. This process thus takes into account the relationship between $\boldsymbol{h}_t$ and $\boldsymbol{h}_i$ to make predictions [107]. The third score function (namely simple-weight in this chapter, Figure 7.2 (b)) is more straightforward and is used in text classification [106]. The current hidden vector $\boldsymbol{h}_t$ is considered a member of the source hidden vectors $\boldsymbol{h}_i \in \{\boldsymbol{h}_{t-l+1}, \dots, \boldsymbol{h}_t\}$. Vector $\boldsymbol{v}_\alpha^T$ and matrix $\boldsymbol{W}_\alpha$ (or $\boldsymbol{W}'_\alpha$) are trained and shared over all time steps. Zero paddings were added to the front when the current location t is smaller than the window size k. The computed attention scores were normalized by a softmax function. The normalized attention score vector $\boldsymbol{\alpha}_t$ of alignment-based attentions and simple-weight attentions are Eq.7.2 and Eq.7.3 respectively:

$$\boldsymbol{\alpha}_t = softmax([\alpha_{t,t-l+1}, \alpha_{t,t-l+2}, \dots, \alpha_{t,t-1}]) \tag{7.2}$$

$$\boldsymbol{\alpha}_t = softmax([\alpha_{t,t-l+1}, \alpha_{t,t-l+2}, \dots, \alpha_{t,t-1}]) \tag{7.3}$$

The context vector $\boldsymbol{c}_t$ is then computed as a weighted sum of hidden state vectors within the window size:

$$\boldsymbol{c}_t = \sum_i \boldsymbol{\alpha}_t \boldsymbol{h}_i \tag{7.4}$$

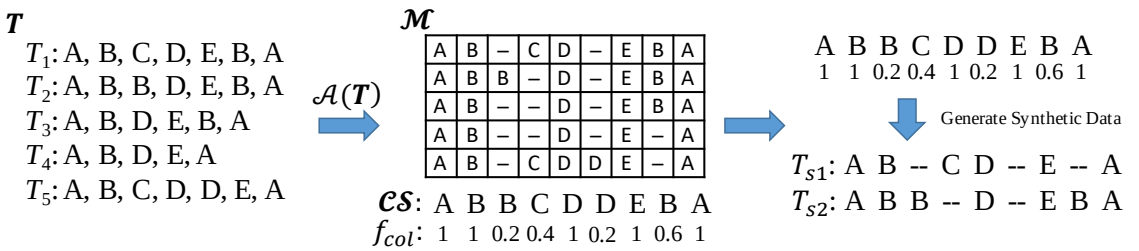


Figure 7.3. Alignment-based synthetic patient data generator. Environmental context attributes associated with each activity were not drawn.

In alignment-based attentions, given the target hidden vector $\boldsymbol{h}_t$ and the context vector $\boldsymbol{c}_t$, a concatenation layer is used to combine the information from both vectors to produce the final attentional hidden vector (Eq.7.5). And in simple-weight attention, we obtained the final attention vector with a hyperbolic tangent function directly (Eq.7.6).

$$\widetilde{\boldsymbol{h}}_t = \tanh(\boldsymbol{W}_c[\boldsymbol{c}_t; \boldsymbol{h}_t]) \tag{7.5}$$

$$\widetilde{\boldsymbol{h}}_t = \tanh(\boldsymbol{c}_t) \tag{7.6}$$

where $\boldsymbol{W}_c \in \mathbb{R}^{d \times 2d}$ is the weight matrix.

7.2.2.6 Classifying

The attentional vector $\widetilde{\boldsymbol{h}}_t$ is fed through the softmax layer to produce the predicted distribution of the next treatment activity:

$$\hat{\boldsymbol{y}}_t = softmax(\boldsymbol{W}_s\widetilde{\boldsymbol{h}}_t + \boldsymbol{b}_s) \tag{7.7}$$

where $\boldsymbol{W}_s \in \mathbb{R}^{u \times d}$ is the weight matrix and $\boldsymbol{b}_s \in \mathbb{R}^u$ is the bias. u is the size of vocabulary of treatment activity types. We used categorical cross-entropy as the loss function (Eq.7.8).

$$\mathcal{L}(\{\hat{\boldsymbol{y}}_t^{(i)}\}, \{\boldsymbol{y}_t^{(i)}\}) = -\frac{1}{n}\sum_{i}^{n}\frac{1}{q^{(i)}}\sum_{t}^{q^{(i)}}((\boldsymbol{y}_t^{(i)})^T \log \hat{\boldsymbol{y}}_t^{(i)} + (1 - \boldsymbol{y}_t^{(i)})^T \log(1 - \hat{\boldsymbol{y}}_t^{(i)})) \tag{7.8}$$

where $\{\boldsymbol{y}_t^{(i)}\}$ is the ground truth activity type at time t in the i-th patient record.

7.2.3 Data Augmentation and Model Pre-training

Pre-training is widely used to address having a limited amount of data. Model parameters are initialized by learning reusable features from datasets of similar domains. However, to our best knowledge, the data we collect for our medical process application is not collected by any other institution in the same manner. Hence, we present a synthetic data generator to fabricate pre-training samples from the limited real training data. Our algorithm (Alg.7.1) has two steps: (1) calculate the alignment matrix (step 1 in Alg.7.1) and (steps 2-10 in Alg.7.1) fit the data distribution into a multivariate Bernoulli (binary) distribution (step 1 in Alg.7.1) and generate synthetic data (steps 11-20 in Alg.7.1).

Different treatment procedures have different patterns and numbers of activities. We thus used trace alignment [10] to find the best alignment across different traces. Given process traces T, the trace alignment algorithm $\mathcal{A}(\boldsymbol{T})$ forms an alignment matrix $\boldsymbol{\mathcal{M}}$ with the traces in T as rows and activities of the same type as columns. If for a given trace a matching activity cannot be found, a gap symbol "-" is inserted in the corresponding cell (Figure 7.3). $\mathcal{A}(\boldsymbol{T})$ also returns the consensus sequence $\boldsymbol{\mathcal{CS}}$, a sequence that records the activities of frequent columns of the alignment matrix. In the meantime, we can calculate the occurrence frequency of each column f_{col}.

With computed $\boldsymbol{\mathcal{CS}}$ and f_{col}, a naïve data generation approach is to consider the occurrence of each activity in $\boldsymbol{\mathcal{CS}}$ follows a univariate Bernoulli distribution with "success" probability f_{col}. The limitation of this naïve approach is that it assumes that the occurrences of the activities across the columns are independent. However, in practice, the occurrences of the activities are correlated. For example, in trauma resuscitation, there is a much higher change to observe the activity "Miami-j collar adjustment" when activity "Miami-j collar

Algorithm 7.1. Synthetic Patient Record Generator with Historical Patient Data

Input: $\boldsymbol{r} = \{\boldsymbol{id}, \boldsymbol{x}, \boldsymbol{T}\}$ /* historic patient records with ids, patient attributes and treatment traces */
Output: r^s /* A synthetic patient record */
Step 1. Calculate alignment matrix $\{\boldsymbol{\mathcal{M}}, \boldsymbol{\mathcal{CS}}\} = \mathcal{A}(\boldsymbol{T})$
Step 2. Calculate correlation matrix $\boldsymbol{corr} = [\phi_{ij}]_{m \times m}$ (Eq.7.10)
Step 3. **for** *col* in range(1,m):
Step 4. Calculate f_{col} (Eq.7.9); $\boldsymbol{p} = \boldsymbol{p} \cup f_{col}$
Step 5. **for** each activity *a* in column *col*:
Step 6. Let $\boldsymbol{b}$ as the activity attributes of *a*
Step 7. Let $\boldsymbol{x}$ as patient attributes associated with the trace that contains *a*
Step 8. $\boldsymbol{B}_{col} = \boldsymbol{B}_{col} \cup \boldsymbol{b}$; $\boldsymbol{X}_{col} = \boldsymbol{X}_{col} \cup \boldsymbol{x}$
Step 9. $\boldsymbol{b}_{col}, \boldsymbol{x}_{col} = \text{avg}(\boldsymbol{B}_{col}), \text{avg}(\boldsymbol{X}_{col})$ /* compute average probability distribution */
Step 10. $\boldsymbol{\mathcal{CS}}[col] = \{a, f_{col}, \boldsymbol{b}_{col}, \boldsymbol{x}_{col}\}$
Step 11. Generate activity trace $\boldsymbol{A} = \text{MVB}(\boldsymbol{p}, \boldsymbol{corr})$ /* A trace of 0s and 1s */
Step 12. **for** *i* in range(1,m):
Step 13. Let $a, f_{col}, \boldsymbol{b}_{col}, \boldsymbol{x}_{col} = \boldsymbol{\mathcal{CS}}[i]$
Step 14. **if** $\boldsymbol{A}[i] == 1$
Step 15. Randomly generate activity attributes $\boldsymbol{b}$ based on activity distribution saved in $\boldsymbol{b}_i$
Step 16. $\boldsymbol{T}^s = \boldsymbol{T}^s \cup \{a, \boldsymbol{b}\}$
Step 17. $\boldsymbol{x}^s = \boldsymbol{x}^s + \boldsymbol{x}_i$ /* sum up the patient attributes over columns */
Step 18. **else continue**
Step 19. Randomly generate $\boldsymbol{x}^s$ based on patient attributes distribution $\boldsymbol{x}^s$/n averaged over columns
Step 20. **return** $r^s = \{0, \boldsymbol{x}^s, \boldsymbol{T}^s\}$

* the source code is available https://github.com/allen9408/Deep_treatment_recommender

application" occurs ("Miami-j collar" is a neck brace used to prevent patient neck movement). Thus, to take into account the correlations, a multivariate Bernoulli (MVB) distribution is more accurate. MVB has two parameters that needs to be estimated from data, the probability vector for each binary variable and the matrix of binary correlations. The probability vector in our problem is the column frequencies:

$$\boldsymbol{p} = [f_1, \dots, f_m] = [\frac{e_1}{\text{n}}, \frac{e_2}{\text{n}}, \dots, \frac{e_m}{\text{n}}] \tag{7.9}$$

where m is the number of columns of the alignment matrix and e_i is the number of non-gap activities in column i. The correlation matrix can be calculated with the phi coefficient [108], a measure of association for two binary variables. The phi coefficient is special case of the Pearson correlation coefficient when the input data are binary variables. The phi coefficient of two columns $\boldsymbol{\epsilon}_i$ and $\boldsymbol{\epsilon}_j$ in the alignment matrix $\boldsymbol{\mathcal{M}}$ is

$$\phi_{ij} = \frac{n_{11}n_{00} - n_{10}n_{01}}{\sqrt{n_i(n - n_i)n_j(n - n_j)}} \tag{7.10}$$

Table 7-2. Statistics of two real-world medical datasets and two synthetic datasets generated using Alg.7.1.

Stats \ Dataset	Trauma	Intubation	Synthetic Trauma	Synthetic Intubation
Num. Patient Records	122	101	5,000	5,000
Num. Total Acts	11,464	1239	470,090	61,356
Num. Act Types	102	15	102	15
Avg. Num. Acts in Trace	93.97	12.27	94.02	12.27
Num. Patient Demographics	22	17	22	17
Num. Activity Attributes	23	0	23	0

where n_{11}, n_{10}, n_{01}, n_{00}, are non-negative counts that add to n. They represent the number of both-present, present-absent, absent-present, and both-absent of the corresponding entries in $\boldsymbol{\epsilon}_i$ and $\boldsymbol{\epsilon}_j$. The n_{i} and n_{j} represent the total number of present entries in $\boldsymbol{\epsilon}_i$ and $\boldsymbol{\epsilon}_j$, respectively. Note that $n_i = n_{10} + n_{11}$ and $n_j = n_{00} + n_{01}$. The correlation matrix $\boldsymbol{corr} = [\phi_{ij}]_{m\times m}$. It is important to note that the Bernoulli distribution requires 0 < p < 1. Hence columns with p = 1 need to be handled separately. The generation algorithm MVB($\boldsymbol{p}, \mathbf{corr}$) is based on the methodology proposed by Demirtas and Doganay [109]. To generate environmental context attributes at the same time, we only need to associate them with the activities in the alignment matrix. Our synthetic data will retain most of real data's characteristics. Throughout, we introduce noise to vary the synthetic data from the authentic data, helping the model generalize better to unseen data.

7.3 Experiments

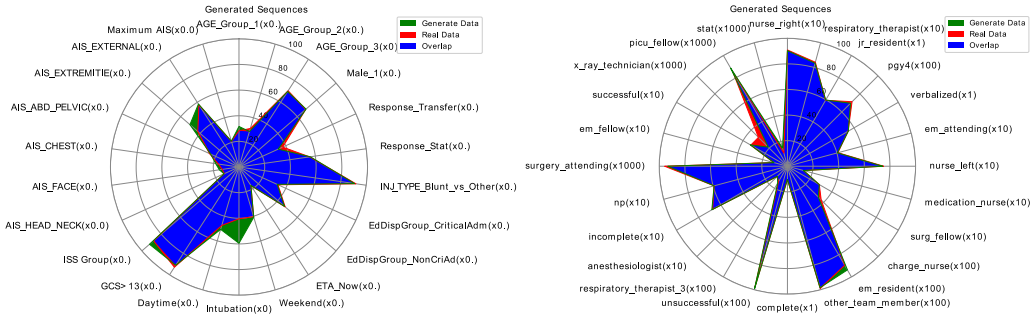
7.3.1 Real-world Data and Synthetic Pre-training Data

The use of medical data for this study was approved by the Institutional Review Board at our hospital. 122 trauma resuscitation records and 101 endotracheal intubation (breathing tube insertion) records were coded from surveillance videos. In addition, we generated 5,000 synthetic trauma records and 5,000 synthetic intubation records from the two real

Table 7-3. Similarity comparison between synthetic data and authentic data (training set). For trace length distribution, the mean value and standard deviation is reported. For the distribution of each activity type and context attribute, the relative difference is reported.

Measures \ Dataset	Trauma (auth.)	Trauma (synth.)	Intubation (auth.)	Intubation (synth.)
Trace length	93.93 ± 24.13	94.02 ± 27.71	12.24 ± 2.56	12.27 ± 2.54
Activity occurrence	reference	6.27% rf	reference	1.08% rf
Patient attributes	reference	7.51% rf	reference	2.89% rf
Activity attributes	reference	8.40% rf	N/A	N/A

* The "rf" stands for "relative difference", which is quantified using mean absolute percentage error: mean(abs($x_s - x_a$)/x_a)*100% , where x_s, x_a are values of activity occurrence, activity attributes or patient attributes in synthetic data and authentic data respectively.



(a) (b)

Figure 7.4. (a) Distribution of patient attributes, authentic data (red) vs. synthetic data (green). Blue shows the overlapped area. Each dimension has scale 0 to 100, indicating the corresponding probability distribution. (b) Distribution of activity attributes, authentic data (red) vs. synthetic data (green). Dimensions of small values are x10, x100 or x1000 for better view.

training sets (Table 7-2). We evaluated the synthetic data quality based on statistics and medical expect feedback. Our results (Table 7-3) showed high similarity between our synthetic data and the authentic data. We compared trace length, activity distribution, patient attributes ((a) in Figure 7.4), and activity attributes ((b) in Figure 7.4). The synthetic intubation records have a high proximity with the authentic records. The distribution of generated trace length only has a 0.03 difference in mean value and a 0.02 difference in standard deviation. Relative differences of the distributions of activity occurrence and context attributes were less than 3%. The trauma process is more complex than the intubation process because it has a larger activity vocabulary (102 vs. 15) and longer traces

(94 vs. 12). Hence bigger distribution differences were observed in the more complex trauma dataset. Our results also show that all synthetic records were unique. In addition, we created a mixed log with 16 authentic and 19 synthetic patient records. A medical expert with experience coding our datasets was asked to identify which cases are authentic and which cases are synthetic. Our results show her classification accuracy was only 54.3 %, similar to random guessing, implying that the synthetic data was realistic and may be observed in practice.

7.3.2 Experimental Setup

7.3.2.1 Evaluation Metrics

Our goal is to correctly recommend the next-step treatment activities to the medical team. The ground truth at a particular time step is therefore the set of activities that occur next. Our trauma dataset and intubation dataset have 102 and 15 possible activity types to recommend, respectively. Hence, in addition to using the standard accuracy (i.e., top-1 accuracy), we also evaluated the top-*k* accuracy. The top-*k* accuracy measures the fraction of the recommendations for which the correct label is among the top-k most probable predicted.

7.3.2.2 Baseline Methods.

We compared our method with six baseline methods.

POP [110]: a baseline method that always recommends the most popular (POP) items in the data. In our problem, it recommends the most frequent activity types.

Act-KNN [111]: a classic recommendation method that returns *k* most similar (*k* nearest neighbors) items. In our problem, it recommends *k* activities most similar to the current activity. The similarity between activities is measured by their locations in the activity traces.

MC [112]: a classic modeling method for sequential data. Markov chains (MC) make predictions based on the latest activity.

HMM [37]: another model for sequential data. Hidden Markov models (HMM) are able to model observations driven by latent variables. The latent variable in our problem can be considered as the treatment goals of the medical team. To avoid subjective initialization of

latent variables, we used a state-splitting method [37] which can automatically infer an optimal HMM.

LSTM: a simple implementation of LSTM with activity sequences as input and weighted loss function, recently introduced to recommendation problems [96].

GRU: a simple implementation of GRU with only activity sequences as input and weighted loss function [98].

7.3.2.3 Our Approaches

We proposed several model variants with different designs (**A**ct2vec, **m**ultiple inputs, sliding-window **a**ttention of 3 types (Eq.7.1) and **p**re-training):

$LSTM_{(A)}$: LSTM with Act2vec (and embedding)

$LSTM_{(Am)}$: $LSTM_{(A)}$ with multiple contextual information as input.

$LSTM_{(Ama1)}$: $LSTM_{(Am)}$ with general sliding-window attention.

$LSTM_{(Ama2)}$: $LSTM_{(Am)}$ with concat sliding-window attention.

$LSTM_{(Ama3)}$: $LSTM_{(Am)}$ with simple-weight sliding-window attention.

$LSTM_{(Ama1p)}$: $LSTM_{(Ama1)}$ pre-trained by synthetic data and fine-tuned by authentic data.

The number of variants doubles to 12 by replacing LSTM with GRU. The implementation details can be found in our code (same link as Alg.7.1). For our and baseline models, we divided the dataset with an 80-10-10 training, validation, and testing split.

7.3.3 Comprehensive Comparison

Our experimental results (Table 7-4) show that the neural networks outperform conventional recommendation and sequential models. As important baselines, POP and Act-KNN only achieved top-1 accuracies of 3.06% and 4.84% respectively on the trauma resuscitation. This exemplifies the challenges of making treatment recommendations from a large activity vocabulary (102 activity types in trauma and 15 in intubation). It also shows the importance of modeling sequential associations between activities in our problem. Classical sequential models like MC and HMM achieved much higher accuracy. Depending on first-order Markov assumptions, their prediction is only based on the immediate previous state. Their high accuracy reveals the strong dependency between the

Table 7-4. Model performance comparison on two real-world medical datasets. The attention window is set to 5 for intubation data and 10 for trauma data. The trauma data has a bigger window size and large *k* values in Top-*k* measure. This design is mainly because that the trauma resuscitation process is much longer and more complex than intubation process.

Model	Trauma			Intubation		
	Top-1	Top-5	Top-10	Top-1	Top-3	Top-5
POP	0.0306	0.1358	0.2552	0.0979	0.3023	0.4537
Act-KNN	0.0484	0.3559	0.4952	0.1134	0.4948	0.6804
MC	0.3469	0.6013	0.6975	0.4276	0.6477	0.7892
HMM	0.3532	0.6178	0.6761	0.3955	0.6949	0.8257
LSTM	0.3646	0.6093	0.7263	0.4227	0.7422	0.8144
$LSTM_{(A)}$	0.3800	0.6305	0.7369	0.4536	0.7113	0.8453
$LSTM_{(Am)}$	0.3849	0.6344	**0.7543**	0.4639	0.7422	0.8866
$LSTM_{(Ama1)}$	0.3878	0.6218	0.7388	0.5051	0.7216	0.8762
$LSTM_{(Ama2)}$	0.3800	0.6325	0.7446	0.5154	0.7422	0.8659
$LSTM_{(Ama3)}$	0.3858	0.6179	0.7427	0.4845	0.7319	0.8350
$LSTM_{(Ama4)}$	0.2746	0.5657	0.7069	0.4536	0.7422	0.8659
$LSTM_{(Ama1p)}$	0.3871	0.6277	0.7302	0.5361	0.7835	**0.9175**
GRU	0.3694	0.6315	0.7515	0.4433	0.7319	0.8556
$GRU_{(A)}$	0.3878	0.6392	0.7408	0.4845	0.6907	0.8453
$GRU_{(Am)}$	0.3955	**0.6412**	0.7524	0.5051	0.6907	0.8453
$GRU_{(Ama1)}$	0.3858	0.6237	0.7301	0.4845	0.7525	0.8247
$GRU_{(Ama2)}$	0.3810	0.6170	0.7417	0.5154	0.7216	0.8659
$GRU_{(Ama3)}$	0.3868	0.6208	0.7466	0.5463	0.7422	0.8866
$GRU_{(Ama4)}$	0.2659	0.5290	0.6818	0.4433	0.7525	0.8350
$GRU_{(Ama1p)}$	**0.3955**	0.6296	0.7302	**0.5567**	**0.7938**	0.8866

adjacent activities. By considering both adjacent and long-term dependencies, the simple GRU and LSTM achieved the better performance. $LSTM_{(A)}$ and $GRU_{(A)}$, featured with Act2vec and embedding layer achieved higher accuracy than LSTM and GRU with a one-hot vector representation. This is expected, as the skip-gram training of Act2vec takes neighboring activities (a form of low-order logic) into account. HMM and MC perform well with just first-order logic, so Act2vec would intuitively help the neural network in prediction. $LSTM_{(Am)}$ and $GRU_{(Am)}$ achieved better performance than $LSTM_{(A)}$ and $GRU_{(A)}$ by taking the advantage of extra context information. Attention mechanism of different architectures improves the model performance in most cases. While in some cases, e.g., $GRU_{(Ama2)}$ and $LSTM_{(Ama2)}$ in trauma data, they do not improve the performance. The

Table 7-5. Different attention architectures with different window size. The dataset used is trauma records. Top-1 accuracy is reported.

Model	Attention Window Size				
	Win = 10	Win = 20	Win = 30	Win = 40	All preceding
$GRU_{(Ama1)}$	0.3858	0.3829	0.3752	0.3762	0.3491
$GRU_{(Ama2)}$	0.3810	0.3820	0.3742	0.3675	0.3627
$GRU_{(Ama3)}$	0.3868	0.3791	0.3771	0.3684	0.3665

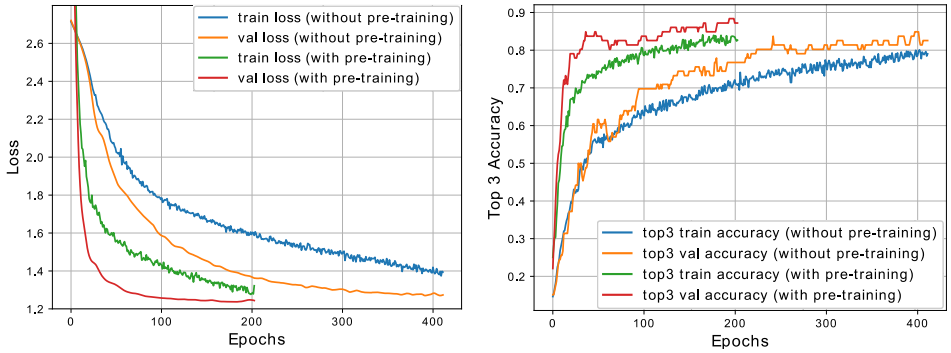


Figure 7.5. Training and validation loss plot (a) and accuracy plot (b) for models with ($GRU_{(Ama1p)}$) and without pre-training ($GRU_{(Ama1)}$). Intubation data was used for these plots. The data used for pre-training is the 5000 synthetic intubation data. The training loss is higher than validation loss because of dropout and regularizer applied in the model.

window size of the attention also affects the performance. Our experimental results show (Table 7-5) the model performance usually decreases as the window size increases. Pre-training on the synthetic data also improves model performance. The pre-trained models ($LSTM_{(Ama1p)}$ and $GRU_{(Ama1p)}$) outperform the models without pre-training, achieved top-1 accuracy of 39.55% and 55.67% with trauma and intubation data respectively. This implies that the synthetic data helps to generalize the model. Another experiment shows that compared with a randomly initialized model, the validation loss of the pre-trained model converges faster to a lower loss and higher accuracy (Figure 7.5). In addition, it is also noticeable that the GRUs outperforms LSTMs in most cases (Table 7-4). The reason is that the GRU has simpler internal structure than LSTM and are easier to train with fewer data.

The results not only show the model performance improvements from our methods but also show the challenges of accurately recommending the next treatment activities. The

recommendations need to be done within 102 classes in trauma data and 15 classes in intubation data. The best performance we obtained is on the intubation process, where we achieved 55.67% top-1 and 79.38% top-3 accuracy. This performance may not be satisfactorily high enough to use in real-world cases. But we were applying this method to simulated medical processes to help train the new medical students.

7.3.4 Visual Analytics for Knowledge Discovery

In the medical field, model interpretability is important. In this section, we show the visual analysis of Act2vec and the sliding-window attention mechanism. The analysis reveals interesting medical insights.

7.3.4.1 Act2vec.

We embedded 102 trauma activities into 100D vectors. Then, we used t-SNE (dimension reduction) to project them onto a 2D plane (Figure 7.6). To test if the data-driven insights matched our human knowledge, we requested our medical experts to group the activities based only on their domain knowledge. According to their treatment goals, our medical experts clustered the 102 activities into 12 groups (colors in Figure 7.6).

Our result reveals several interesting insights. First, activities of the same medical goal are usually closer than activities of different medical goals. This finding indicates that, in most cases, our medical team accomplishes the trauma resuscitation by addressing medical goals one by one rather than simultaneously. Second, without taking into account the points that lie alone in low-density regions, the activity points can be grouped into four major clusters (dashed circles in Figure 7.6). The clusters reveal high-level medical goals. The left cluster consists of goals airway (A), breath (B), circulation (C), disability (D), and exposure (E). These five goals constitute the primary survey, a medical phase with the goal of quickly identifying life-threatening injuries. The top and right clusters constitute the secondary survey, a head-to-toe physical examination of the patient's body. The bottom cluster includes activities assessing the patient's back and the conditional treatments performed depending on assessment outcomes.

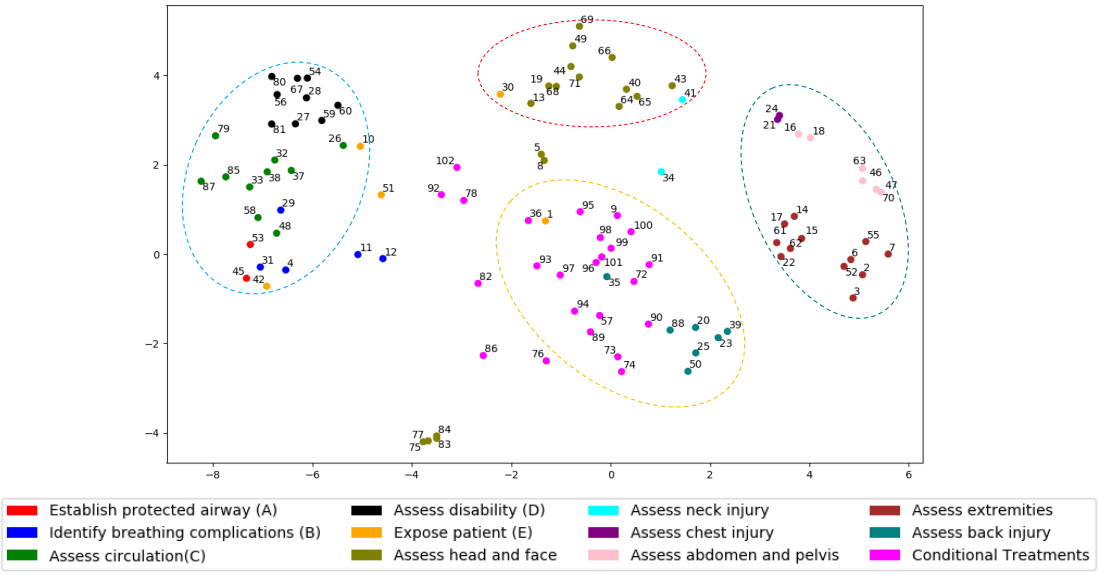
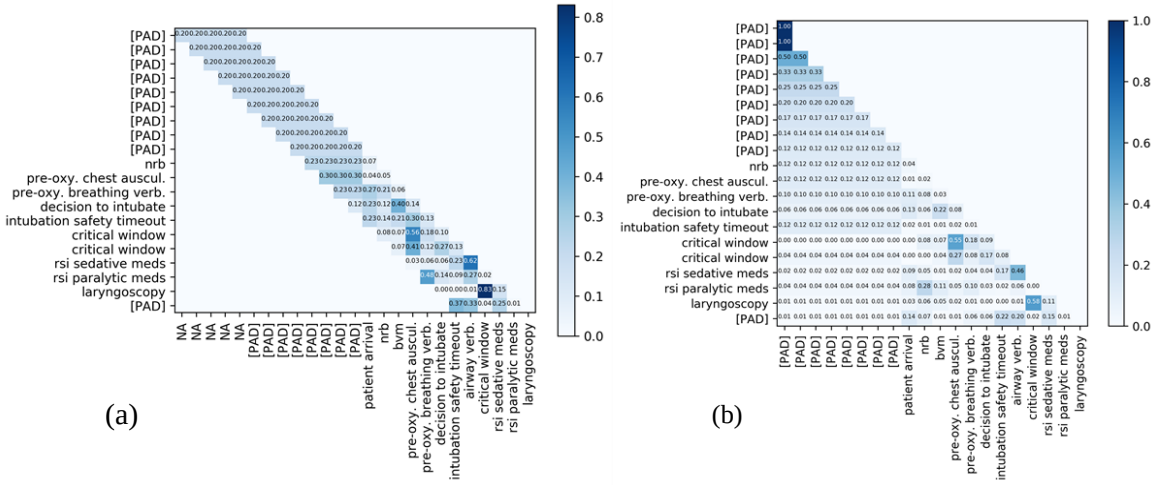


Figure 7.6. Activity vector visualization. Each dot (labeled by a unique number) represents an activity (type). The distance between dots reveal the similarity of the activities. The color of the dots reveals the probable associated medical goals of the treatment activities.

7.3.4.2 Sliding-window Attention.

We visualized the sliding-window attention score vectors of each time step in a matrix (Figure 7.7 (a)). The attention scores were computed from the general attention architecture. The value of the attention score roughly reflects how important the hidden vector helps the current $\boldsymbol{h}_t$ predict the next activity. For example, when the current input is activity "rsi paralytic medicine" (the 2nd to last activity in the horizontal axis in Figure 7.7 (a)), the current hidden vector $\boldsymbol{h}_t$ heavily relies on presence of the "critical window" hidden vector in the sliding window (the 4th to last activity in the horizontal axis) to predict "laryngoscopy" (the 2nd to last activity in the vertical axis). In addition, we also visualized that all previous hidden layer was given some attention, including those of padding (Figure 7.7 (b)). Considering more previous hidden layers can then be said to dilute attention, compared to using a small sliding-window. It is also interesting to see that attention is usually given to the several latest states. This finding is aligned with our knowledge that the medical team tends to consider just the most recent activities when planning the next steps.



(a) (b)

Figure 7.7. Visualization of attention scores for an intubation case. The attention scores were generated using the general attention architecture (Eq.7.1). The horizontal axis represents the activities (from left to right) that occurred sequentially in this intubation case. The vertical axis represents the predicted activities. [PAD] denotes the “paddings” added for variable length input trace, and [NA] denotes the “paddings” added for the sliding window in the attention. The color of each cell represents the attention score (α_t in Figure 7.2) of each hidden vector. (a) Sliding-window attention mechanism with a window size of 5. (b) Attention mechanism that considers all the preceding hidden vectors. It can also be considered as a special case of the sliding-window mechanism when the window size is larger than the length of activity trace.

Part IV

Implementation and Conclusion

Part I: Introduction

Chapter 1 Introduction	Chapter 2 Preliminaries

Part II: Applied Process Mining and Analysis

Chapter 3 Workflow Model Discovery	Chapter 4 Workflow Deviation Analysis	Chapter 5 Patient Cohorts Analysis

---

Part III: Process Recommender System

Chapter 6 Trace-level Recommendation	Chapter 7 Activity-level Recommendation

---

Part IV: Implementation and Conclusion

Chapter 8 VIT-PLA	Chapter 9 Conclusions

Chapter 8
VIT-PLA: Visual Interactive Tool for Process Log Analysis

We developed two different versions of VIT-PLA, the Java version (Figure 8.1) and the Web version (Figure 8.2). This chapter focuses on the Java version, which is based on our paper [2]. Techniques for analyzing and visualizing process or workflow data have been developed and applied in a wide range of domains. Visual analysis of large process logs and integration of statistical analysis, however, have been limited. We introduce the Visual Interactive Tool for Process Log Analysis (VIT-PLA) that provides a simplified process log visualization and performs statistical correlation analysis on process attributes. We demonstrate its use by applying it to an artificial dataset and running a preliminary analysis of trauma team task data collected from a medical emergency department.

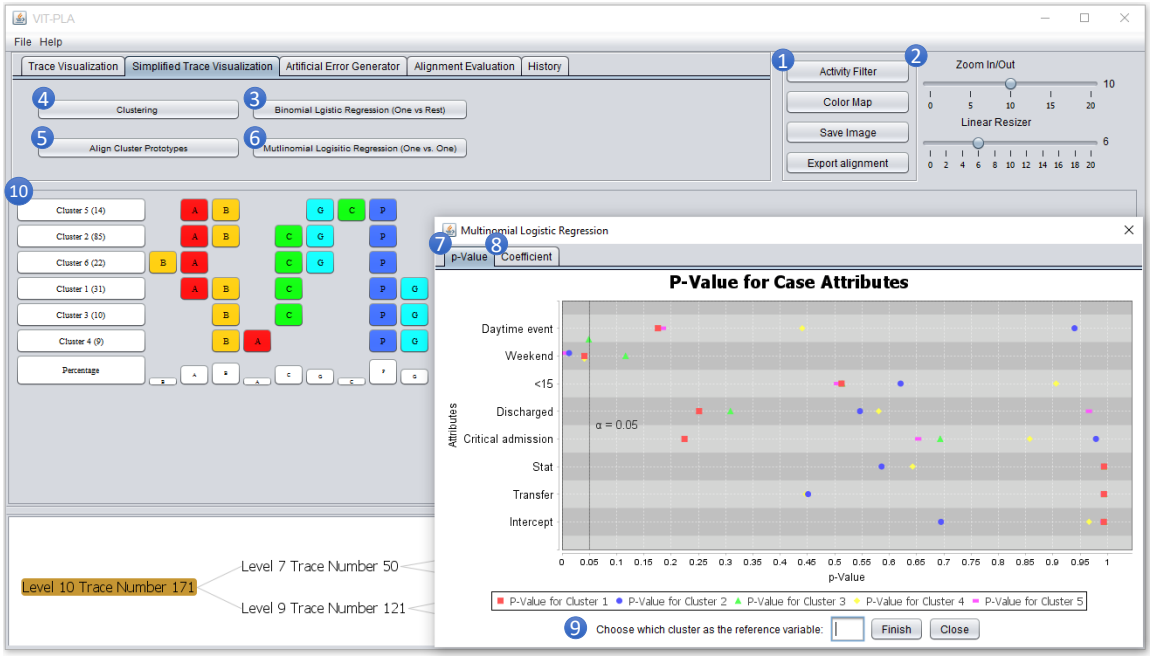


Figure 8.1. VIT-PLA, the Java version. This work is based on our paper [2].

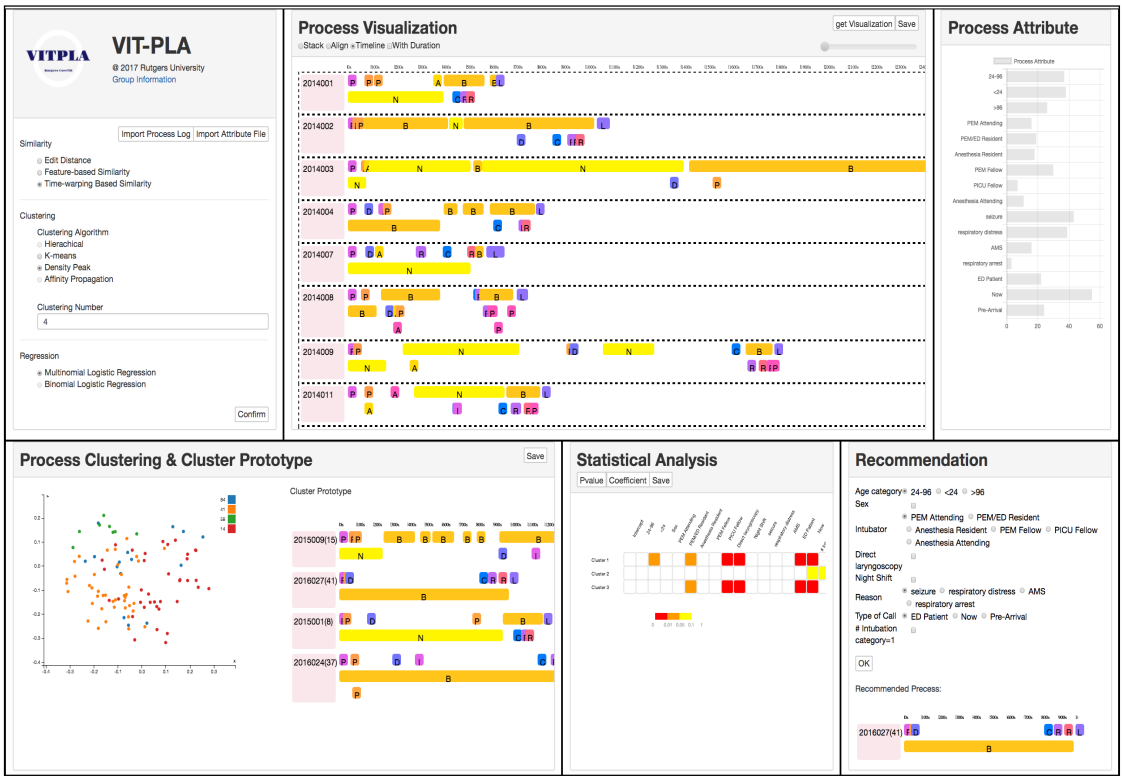


Figure 8.2. VIT-PLA, the Web version. This work is based on our paper [3].

8.1 Motivation

Many contemporary information systems record activity logs, including personal calendars and electronic health records (EHR). Process mining techniques attempt to extract non-trivial knowledge and insights from these activity logs and use them for further analyses [18]. Most research in process mining has focused on workflow discovery and process execution visualization [18][113]. When visualized, real-world workflow often produces "spaghetti-like" graphics that are difficult to analyze and do not provide useful observations or insights. In addition to graphical visualization, other efforts have also been made to produce different visualizations for process executions or workflow data [30][114][115][116][117][118][119]. Although these systems have been shown to work well with focused processes and relatively small event logs, little work has been done with large process logs with many execution traces (typically hundreds or thousands of different process cases). Simply displaying all traces at once does not make a useful visualization. We observed that only several dozen traces can fit intelligibly on one screen at a time. Even if the symbols were distinguishable, the amount of displayed data make it inconvenient for human interpretation. When working with large workflow datasets, it is often useful to obtain a concise visualization that summarizes the data into an easily interpretable format. We present an approach for visualizing a summary of large process logs by aggregating the data with a trace clustering method. Process traces are clustered based on the similarity or proximity between their elements (i.e. process tasks). Each cluster is represented using a "representative" or "average" trace extracted from the corresponding cluster. Using this approach, we are able to usefully visualize large process logs. To help users better understand the clusters, we also included tools for running statistical tests on the clusters and their associated process attributes. These statistical test results can reveal significant and interesting correlations between process executions and process attributes. We implemented these approaches in a Java-based application, named VIT-PLA.

Our main contribution is a novel approach to producing summarized visualizations of large process logs and directly integrating statistical analyses into the visualization. These

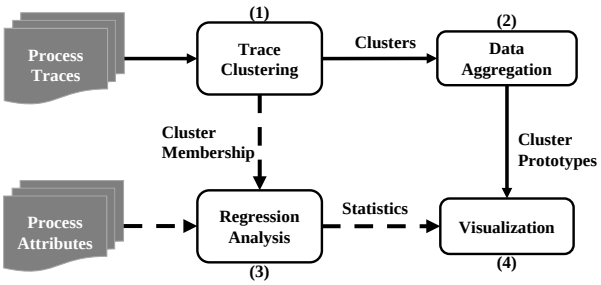


Figure 8.3. Flowchart outlining the core methods implemented in VIT-PLA and their corresponding inputs and outputs.

features help users discover attributes associated with specific sequence progressions and deviations within the dataset.

8.2 Methods

The core methods implemented in VIT-PLA can be summarized as follows (Figure 8.3) : (1) clustering of process traces (workflow data) based on proximity of data objects, (2) aggregation of process traces and selection of cluster prototype, (3) regression analysis to explore underlying knowledge, (4) interactive visualization of process traces and statistical analysis results. This section will describe (1), (2), and (3); (4) will be discussed in Section 3.

8.2.1 Data Preprocessing: Sequencing of Traces

Process sequencing is necessary before more advanced processing. Activities coded in a process log usually have start and end timestamps (some logs may not include end time) for each activity. Idle time may exist between activities, and some activities may be executed concurrently (Figure 8.4(a)). In process mining, process traces are usually sequenced by ascending order of the start time of activities (Figure 8.4(b)).

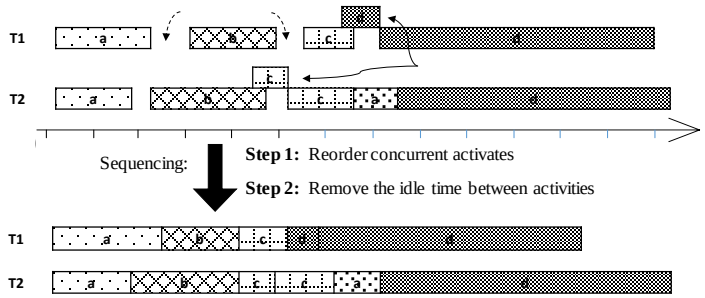


Figure 8.4. Two steps of sequencing the traces with concurrent activities (such as *d* in T_1 and *c* in T_2) and idle times (white spaces between activities). (a) Example process traces before sequencing. (b) The same process traces after sequencing.

8.2.2 Summary Visualization of Process Logs

8.2.2.1 Process Trace Clustering

Our approach uses clustering techniques to simplify the process trace visualizations. Clustering provides an abstraction from the original data objects to generalized data representatives, i.e. cluster prototypes. In most data mining problems, data clusters are calculated based on the data objects' feature set. However, to aggregate process traces that follow an underlying workflow model, we cluster the traces based on the similarity of their constituent tasks in terms of task type and sequential order of execution [86]. That is to say, our sole feature used for clustering is the structure of each trace's task sequence, not the process attributes.

In VIT-PLA, the clustering algorithm we use is agglomerative hierarchical clustering [120] with Ward's method [89] as clustering criterion. We calculate the similarity of process traces based on Edit Distance (a.k.a. Levenshtein Distance [38]). If activity duration information is also available, the similarity can be calculated with "Duration-Aware Edit Distance" [1], a metric derived from Edit Distance that penalizes dissimilarity between durations of the same activity type.

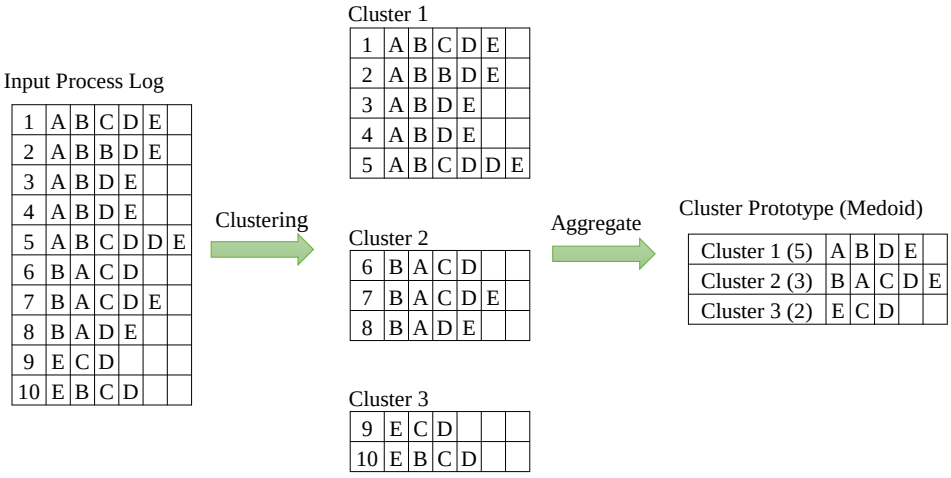


Figure 8.5. An example showing data clustering and aggregation. The cluster prototype used here is cluster medoid.

8.2.2.2 Cluster Prototype and Trace Alignment

After clustering, each cluster can be characterized by a cluster prototype (Figure 8.5). Because it is not practical to visualize all the data objects on a single computer screen, a substantial reduction in the data size is needed. The deployment of cluster prototypes helps compress the dataset.

Several candidates can be considered as cluster prototype, such as the widely-used cluster centroid [121], the center of a cluster. There is, however, a great chance that there may not be an actual data point at the cluster's center. In this case, the centroid location is calculated from the data in the cluster with the aim of minimizing the sum-squared distance to other points.

Note that for categorical data and event-based data, the notion of a center (centroid) does not apply [121]. For example, the centroid of categorical data (e.g. {orange, apple, banana}) cannot be determined. In this case, we may use the cluster medoid, the most representative data object in the cluster, i.e. a data point with minimal average dissimilarity to all other objects in the cluster. The medoid, however, may not be adequate if the cluster does not contain an "appropriate" representative.

To ensure that the chosen sequence is representative of the cluster, we used the consensus sequence as the cluster prototype even though it may not be an observed trace from the data. The consensus sequence, a concept derived from aligning biological

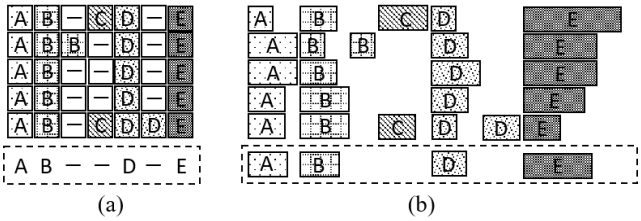


Figure 8.6. An example of two types of trace alignment: (a) Context-Aware and (b) Duration-Aware. The sequences at the bottom of (a) and (b) are consensus sequences derived from the data. A gap symbol "-" or white space is inserted if a match cannot be found. The five process traces shown here are from Cluster 1 in Figure 8.5.

sequences (e.g. DNA) in bioinformatics, is a sequence of the most frequent residues found in the alignment matrix's columns. In process mining, consensus sequences may be considered the "average" or "common" sequence of tasks [30] (Figure 8.6). To find the consensus sequence for each cluster, trace alignment [1][30] needs to be performed using traces from each cluster respectively. Trace alignment reformats the original data by placing the same or similar activities of all traces to the same column of the alignment matrix. If a matching activity cannot be found, a gap symbol "-" is inserted. Bose and Van der Aalst [30] have shown how to use trace alignment techniques to visualize and analyze process traces (Figure 8.6(a)). In our previous work, we extended their work by introducing a duration-aware trace alignment algorithm [1] that also takes activity duration into consideration. In our implementation, the alignment algorithm can work for data either with or without activity durations (Figure 8.6).

8.2.3 Association between Trace Clusters and Trace Attributes

In addition to visualization, VIT-PLA also provides statistical analysis functions. The goal of our statistical analyses is to help the user discover the underlying associations between data cluster membership and trace attributes. This goal is accomplished using either multinomial or binary logistic regression. The user chooses between these two statistical methods depending on the domain question being asked. Multinomial logistic regression works for binary comparison between two clusters (one-vs.-one cluster comparison), while binomial logistic regression works for binary comparison between one cluster and the rest

of the clusters (one-vs.-rest). Using both logistic regression models can help discover attributes associated with particular clusters.

8.2.3.1 Multinomial logistic regression

In multinomial logistic regression [122], let *K* denote the number of independent variables, and let *J* denote the number of discrete categories of the dependent variable, where $J \geq 2$. In our case, the independent variables correspond to the trace attributes and the dependent variables correspond to the trace cluster membership. The number of trace attributes is *K* and the number of clusters is *J*. By default, we define the last category (the *J*th cluster) to be the reference category, against which logits of the first *J*−1 categories are compared. Let *C* denote cluster membership. Represented formally:

$$\ln\left(\frac{P(C=i)}{P(C=J)}\right) = \ln\left(\frac{P(C=i)}{1-\sum_{j=1}^{J-1}P(C=j)}\right) = \beta_{i0} + \beta_{i1}x_{i1} + \beta_{i2}x_{i2} + \cdots + \beta_{iK}x_{iK}, \qquad i = 1, \ldots, K-1 \tag{8.1}$$

where x_i are trace attributes, and β_i are regression coefficients for each of the trace attributes. In VIT-PLA, users can also choose which cluster to use as the reference category.

8.2.3.2 Binomial logistic regression

Binary logistic regression [122] is a special case of multinomial logistic regression, in which there are only two categories ($J = 2$). In our problem, one category is the target cluster of interest and the other category is all other clusters. Let *K* denote the total number of independent variables and *C* denote cluster membership. Represented formally:

$$\ln\left(\frac{P(C=i)}{P(C\neq i)}\right) = \ln\left(\frac{P(C=i)}{1-P(C=i)}\right) = \beta_{i0} + \beta_{i1}x_{i1} + \beta_{i2}x_{i2} + \cdots + \beta_{iK}x_{iK}, \qquad i = 1, \ldots, K \tag{8.2}$$

where the parameters have the same meaning as in Eq.8.1.

8.2.3.3 Hypothesis Test

To identify which trace attributes are significantly associated with cluster membership, we use the Wald test [90] for logistic regression, which is defined as:

$$W = \frac{(\hat{\beta}_i - \beta_i)}{\widehat{se}(\hat{\beta}_i)} \tag{8.3}$$

where $\hat{\beta}_i$ is the regression coefficient for trace attributes x_i; $\beta_i = 0$ is the null hypothesis, i.e. the trace attribute x_i has a corresponding coefficient of zero; se is standard error. In our implementation, we use a normal distribution and z-values for calculating p-values. The null hypothesis can be rejected when p-value is less than or equal to alpha, the significance level which is most often set at 0.05.

8.3 Visual Interface Design

During software development, we received feedback from domain experts and continuously improved our design. In this section, we describe the first prototype of VIT-PLA. The visual interface design (Figure 8.1) was developed with three main goals:

G1. Interactive visualization of raw process traces, the basic visualization functionality.

G2. Simplified visualization of process traces (for large data applications).

G3. Visualization of trace cluster vs. trace attribute association statistics.

Although VIT-PLA has many other functions, the rest of this chapter focuses on how its design achieves these three goals.

8.3.1 G1: Three Common Ways to Visualize Raw Process Traces

VIT-PLA provides three common ways of visualizing raw process traces. We refer to the data as “raw process traces” to distinguish goal G1 from G2, where the data is visualized in an aggregated format. The three visualization methods are:

Simple stack of activities in the process traces (Figure 8.7 (a) without activity duration, and Figure 8.7 (b) with activity duration). This approach is one of the simplest ways to visualize process traces. Activities are stacked based on their occurrence time. Activity information can be accessed with a mouse click on the corresponding symbol. This

visualization is easily interpretable and computationally efficient, but it cannot provide deep insights into the data.

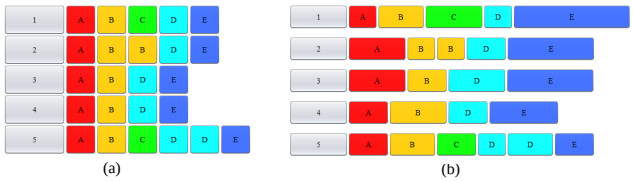


Figure 8.7. Simple stack (a) Process executions are stacked (b) Process executions are stacked and symbol blocks are scaled based on activity duration. Each row represents a single trace and each block represents a single activity. The data comes from Cluster 1 in Figure 8.5.

Overlay of the process execution on the timeline (Figure 8.8). Activities are scaled based on duration and aligned to the timeline according to their start and end times. The advantage of this visualization approach is that it clearly shows the concurrent activities in each process.

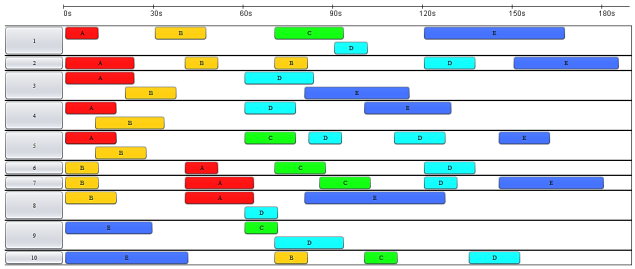


Figure 8.8. Visualize process traces on a timeline. The top scale is the timeline with second as the unit. Each row, separated by a bold line, represents a single process. Each block represents a single activity. Symbol blocks that are vertically stacked in one process are activities occurring simultaneously. The data comes from the input log in Figure 8.5**.**

Process trace alignment (Figure 8.9 (a) context-aware alignment and Figure 8.9 (b) duration-aware alignment). The context-aware trace alignment algorithm is based on Bose and Van der Aalst's work [30] and the duration-aware trace alignment algorithm proposed in our previous research [1]. The duration of each activity in the consensus sequence

(bottom line of Figure 8.9 (b)) of duration-aware trace alignment is the mean activity duration of the corresponding column. Compared with the previous two visualizations, the alignment view makes it easier to interpret process traces and extract insights. When considering algorithm execution time, our previous research found that for a moderately-sized dataset (e.g. 50,000 activities, ~1,000 traces and ~50 activity for each trace), the alignment can be effectively calculated in 25.5±1.5 seconds [1]. This time is not instantaneous (which would be ideal), but is still reasonable.

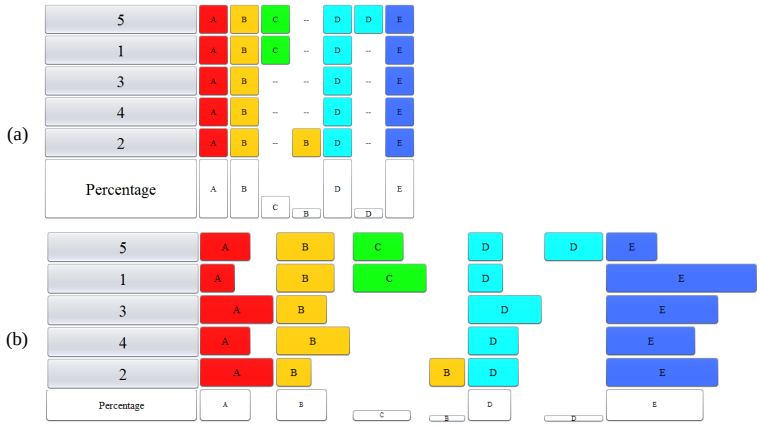


Figure 8.9. Alignment (a) Process trace alignment (b) Duration-aware trace alignment. Each row represents a single process and each block represents an activity. The bottom line of each figure is the consensus sequence. Dashes or spaces are introduced to achieve alignment of the activities. The data comes from Cluster 1 in Figure 8.5.

8.3.2 G2: Simplified Visualization of Process Traces

The first interactive visualization feature in G2 is the selection of cluster number (clicking button ① in Figure 8.1 and inputting cluster number *k* in the pop-up dialogue). A hierarchical tree structure with *k* clusters will be shown at the bottom panel (Figure 8.1 and Figure 8.10) where the non-leaf (a.k.a. internal) nodes show the current height (a.k.a. depth) and process traces included under this node. *k* leaf nodes correspond to the *k* clusters and display all the process IDs in the cluster.

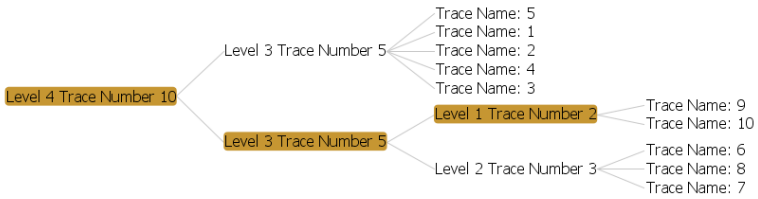


Figure 8.10. Hierarchical Tree Structure (we cited the same source code from ProM [63] here and made modifications showing only the number of clusters specified by the user). The result is based on the data in Figure 8.5.

After clustering, each cluster is represented with its own cluster prototype. By default, the cluster prototypes are visualized as activity stacks (Figure 8.11). The prototypes can also be visualized in alignment view (Figure 8.1 and Figure 8.12) by clicking on the button "Align Cluster Prototype" (② in Figure 8.1). Another interactive function allows the user to check the pre-aggregated traces under a certain cluster. This feature may be accessed by clicking on the buttons showing the cluster information (③ in Figure 8.1).

8.3.3 G3: Visualization of Statistics of Trace Clusters vs. Trace Attributes.

Users can access statistics of trace clusters and trace attributes by clicking on the button "Multi-Logistic Regression" (⑤ in Figure 8.1) or on "Binomial Logistic Regression" (④ in Figure 8.1). The number of clusters is decided by the user. The significance tests for trace attributes on trace clusters (*p*-value statistics) are shown in a chart (⑥ in Figure 8.1, JFreeChart[2] library is used). The horizontal axis represents the *p*-value, while the vertical axis represents the trace attributes. The *p*-value of different clusters is denoted with different shapes and colors. Because alpha = 0.05 is widely used as the significance level, we placed a highlighted line at this level. When performing multinomial logistic regression, the reference category is set to the last-numbered category by default. Users, however, may change the reference category manually (⑦ in Figure 8.1). In addition to *p*-values for each

[2] http://www.jfree.org/

trace attribute, the regression coefficients of the logistic regression model are also listed in a table (⑧ in Figure 8.1 and Figure 8.13).

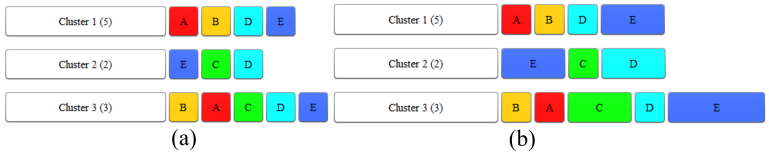


Figure 8.11. Simplified visualization of raw process traces. Each row is a cluster's prototype. The information in the white block before the prototypes shows the cluster ID that each prototype represents and the number of process traces in that cluster. (a) Cluster prototypes are consensus sequences calculated from context-aware alignment (Figure 8.9 (a)); (b) Cluster prototypes are consensus sequences calculated from duration-aware alignment ((b)). The data comes from Figure 8.5**.**

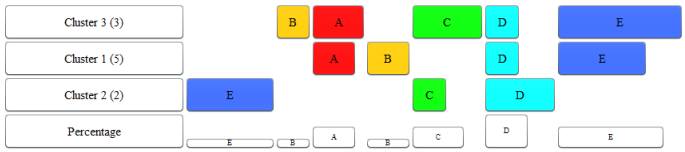
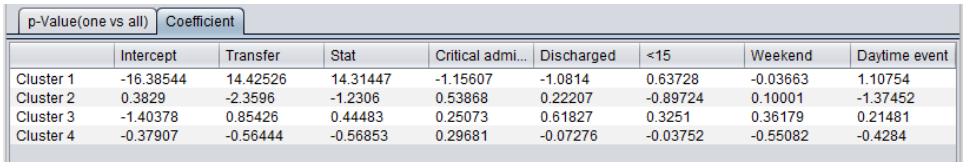


Figure 8.12. Alignment view of the cluster prototypes in Figure 8.11(a). The data comes from Figure 8.5**.**



	Intercept	Transfer	Stat	Critical admi...	Discharged	<15	Weekend	Daytime event
Cluster 1	-16.38544	14.42526	14.31447	-1.15607	-1.0814	0.63728	-0.03663	1.10754
Cluster 2	0.3829	-2.3596	-1.2306	0.53868	0.22207	-0.89724	0.10001	-1.37452
Cluster 3	-1.40378	0.85426	0.44483	0.25073	0.61827	0.3251	0.36179	0.21481
Cluster 4	-0.37907	-0.56444	-0.56853	0.29681	-0.07276	-0.03752	-0.55082	-0.4284

Figure 8.13. Statistics for regression coefficients

8.3.4 Additional supportive functions

In addition to the three main goals, VIT-PLA also includes several useful supportive functions. The Activity Filter (⑨ in Figure 8.1) allows the user to include and exclude activities in the visualization and analysis. The Color Map (⑨ in Figure 8.1) allows the

user to recolor the activity symbols. The Zoom Slider (⑩ in Figure 8.1) enables the user to resize the activity symbols in the visualization panel (the sliders in the top-right corner control the size of the activity symbols).

8.4 Case Studies

8.4.1 Case Study I: Artificial Data

8.4.1.1 Data Description

This dataset was artificially generated using the Process Log Generator (PLG) [123]. It includes 500 process traces consisting of 10 different activity types. The drawback of this artificial data is that it does not have background attributes associated with each process trace. For this reason, we only focus on the simplification of trace visualization when using this dataset.

8.4.1.2 Results and Discussion

The visualization of 500 process traces without data aggregation strategies can lead to extremely large and complex visualization results (Figure 8.14(a)). When represented this way, the symbols are too small to identify, making it difficult to extract useful information. To improve visualization, we used clustering to aggregate the original dataset into a small number of representative process traces (Figure 8.14(b). In this example, we arbitrarily chose 10 clusters, a manageable number of clusters to understand). The visualization becomes clearer when put into the alignment view (Figure 8.14(c)). From these two simplified visualizations (Figure 8.14(b) and Figure 8.14(c)), it is easy to extract some interesting insights: (1) the sequential order of consensus tasks (tasks that occur more than or equal to 50% in the column) is “ACEGFDHIB”; (2) the pattern “HIJ” is repeated in two of the ten clusters (cluster 1 and cluster 2); (3) activity C is performed late in one cluster (cluster 5); and (4) activity D is performed late in one cluster (cluster 3) and omitted in another (cluster 7).

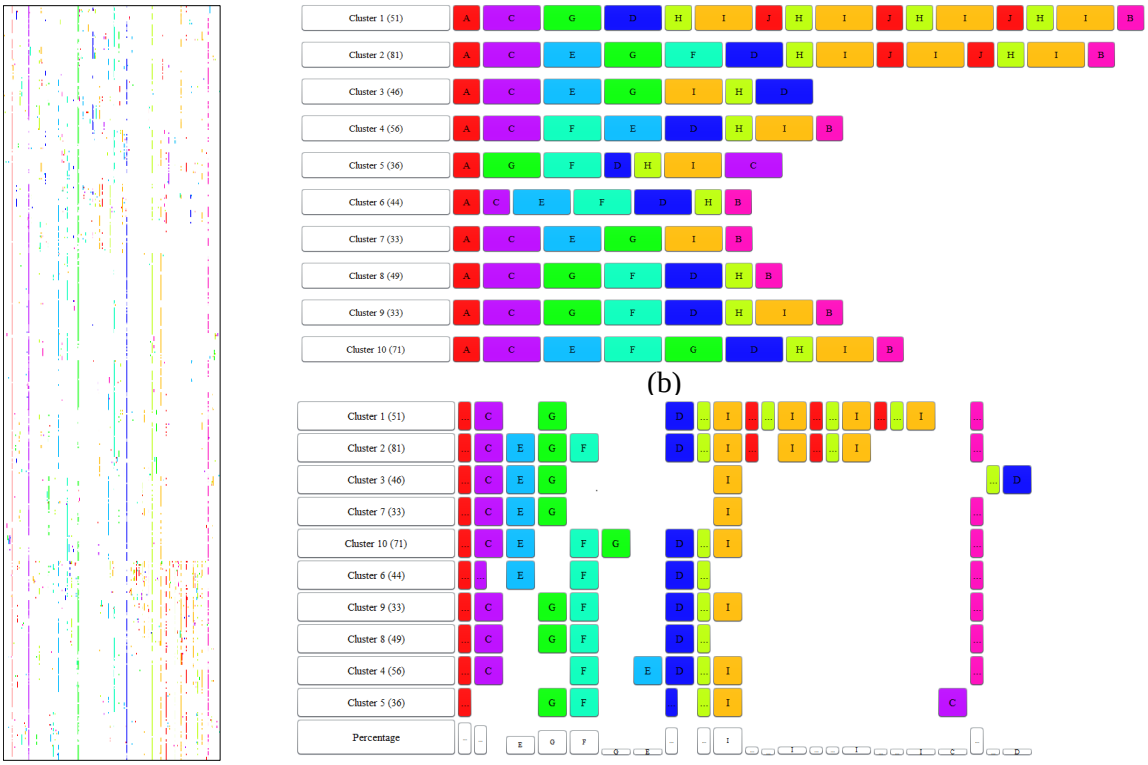


Figure 8.14. Visualization of artificially generated dataset. (a) Alignment view of all 500 process traces; (b) Simplified visualization of 500 process traces using 10 cluster prototypes; (c) Alignment view of 10 cluster prototypes.

8.4.2 Case Study II: Trauma Resuscitation Workflow Data

8.4.2.1 Data Description

We used a trace log obtained from video analysis of 171 child trauma resuscitations between May and August 2013 at Children's National Medical Center in Washington, DC. An event log of five activities typically performed during the initial evaluation was created and used as the dataset for this case study. We obtained the workflow model for these activities from domain experts (Figure 8.15(a)). Activities "Airway, Breath, Circulation" follow a sequential order. Activities "GCS" and "Pupil check" are parallel and should be performed after the previous three activities. We also obtained from the medical chart

Table 8-1 Process trace attributes

Attribute List	Values		
Weekend Event	1	0	
Daytime Event	1	0	
ISS Score	<15	≥15	
Activation Level[1]	Attending Stat	Stat	Transfer
EDDISPGroup[2]	Non-critical Admission	Critical Admission	Discharged

[1]Activation level = pre-hospital triage level
[2] EDDISPGroup = admission status of patients after ED care

review several patients and resuscitation attributes (including pre-hospital triage level, the resuscitation's time of the day and day of the week, Injury Severity Score [ISS], and patient admission status after the resuscitation) (Table 8-1). This dataset is not a "large process log," but we chose it for our preliminary analysis to demonstrate how our approach can be integrated with medical domain knowledge.

8.4.2.2 Results and Discussion

Data Interpretation from Visual Analysis

Four cluster prototypes were generated (Figure 8.15(b) and (c)). Prototypes of clusters 1 and 3 conform to our expert model, but clusters 2 and 4 do not. From the alignment view of prototypes, we can observe that the sequential order of activity GCS (G) and pupil assessment (P) is interchangeable, which conforms with the parallel structure in our expert model. Visualizations of pre-aggregated traces for each prototype are not displayed, but users can visualize the traces by clicking on the cluster button at the front of each row (Figure 8.15(b) and (c)).

With the attribute data for these process traces, we can perform statistical analysis to explore the underlying correlation between the trace attributes and trace cluster membership. The following are examples of the statistical findings, followed by feedback from domain experts:

Observation #1: Attribute "Daytime Event" is statistically significant (p-value = 0.021, red square point in row "Daytime event" in Figure 8.15) for cluster 1. The regression coefficient of Daytime Event is 1.108 (Figure 8.13). This attribute is statistically significant

because the proportion of data objects that have this feature (daytime = 1) in this cluster is 12/31 (68%), while the proportion of data objects that have this feature (daytime = 1) in the reference category (all other clusters) is 71/140 (51%).

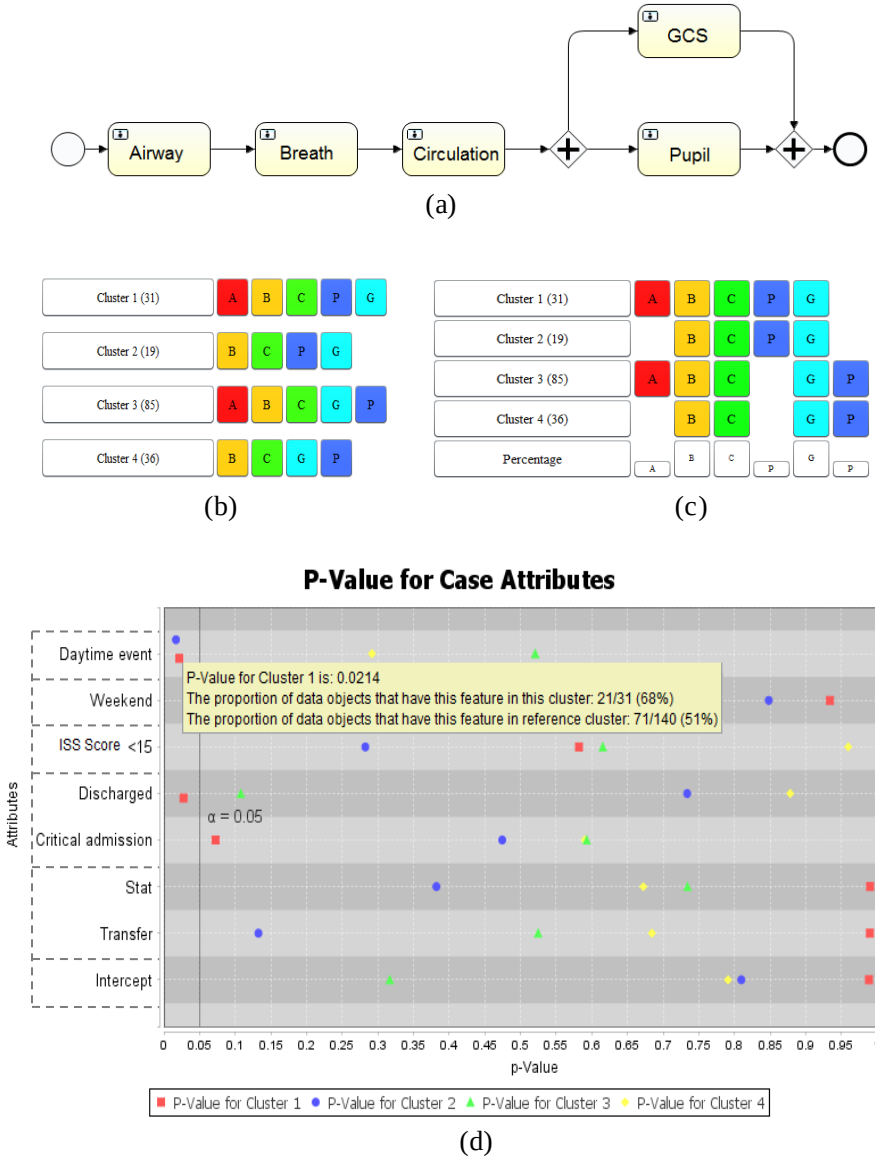


Figure 8.15. (a) Workflow model (drawn based on BPMN) given by domain expert describing the initial evaluation of trauma, (b) Simplified visualization of 171 traces using four cluster prototypes, (c) Alignment view of four cluster prototypes (d) *p*-value for binomial logistic regression coefficients

Observation #2: Attribute “Daytime Event” is statistically significant (*p*-value = 0.017, blue circle point in row “Daytime event” in Figure 8.15) for cluster 2. The regression coefficient of Daytime Event is −1.375 (Figure 8.13). This attribute is significantly significant because the proportion of data objects that have this feature (daytime = 1) in this cluster is 6/19 (31%), while the proportion of data objects that have this feature (daytime = 1) in the reference category (all other clusters) is 86/152 (57%).

Medical expert feedback: For the care of injured patients, improved outcomes are associated with compliance with the Advanced Trauma Life Support model [16], represented here as the expert model. We find that one cluster (cluster 1) whose cluster prototype follows the model occurs more often during the day and another cluster (cluster 2) whose cluster prototype deviates from the model occurs more often at night. This association finding supports previous work showing decreased compliance with trauma protocols at night [46].

Domain Expert Feedback on VIT-PLA Design:

To evaluate the quality of our design, we had two medical domain experts evaluate a prototype of VIT-PLA. Both positive and negative feedback was received.

Both domain experts liked the visualization’s flexibility and interactivity. They found that its data clustering, activity filtering, symbol resizing, and recoloring functions were very useful. They were also found that with the knowledge uncovered by the program’s statistical analysis was useful. One domain expert found it useful to switch between the aggregated data and the original traces, and also commented on the helpfulness of the cluster’s “average sequence”.

Most negative comments focused on our approach for statistical analysis. One domain expert felt that data-driven clustering approach lacked consistency because its result varied when different clustering algorithms or similarity metrics were used. Also, the domain expert found that some small clusters did not have sufficient data to support the statistical hypothesis test correlating trace clusters and trace attributes.

Chapter 9
Conclusions

This dissertation has three major contributions which correspond to the three topics of Part II, Part III, and Part IV. Specifically, the contributions are: (1) novel approaches and frameworks for applied process mining and their implementations in real-world medical processes; (2) two different process recommender systems that close the gap between process mining and recommender systems; and (3) development of a visual analytic tool for process mining. The presented approaches, frameworks and tools were evaluated with several real-world datasets. As we have a partnership with Children's National Medical Center, we were able to continuously access valuable domain knowledge and feedback on our methods. Although most of the datasets and case studies are conducted in medical domain, our methods can be easily applied to other problems with similar settings.

In Part II, we developed novel process mining methods and applied them to real-world medical process analysis. First, to enhance the existing workflow discovery algorithm, we developed a splitting-based workflow discovery method. Our method is able to tackle the duplicate-activity problem by allowing the activity nodes in the model to further split. Second, to quantify and analyze the discrepancies between work-as-done and work-as-imaged, we invented a framework for automatic process deviation detection. This framework provides a method for identifying repeated, omitted and out-of-sequence activities that can be included in the design of decision support systems for complex medical processes. Third, to analyze the differences between the medical treatment procedures of different patients, we introduced a framework for analyzing the association between treatment procedures and patient cohorts. The framework works by learning weights of context attributes by best-first search, deciding patient cohorts using clustering algorithms, discovering treatment procedures (or patterns) with process mining techniques, and analyzing the cohort-vs.-procedure through statistical analysis.

In Part III, we presented two process recommender systems which present as a bridge between process mining and recommender systems. We first designed a prototype-based

recommender system. This approach relies on mining historic data to uncover the potential association between the way of enacting a process and contextual attributes. If association tests are significant, we train a recommender system to output a prototypical enactment for the given context attributes. Later, we proposed another recommender system that is able to provide a step-by-step recommendation. The system was built on recurrent neural networks. The networks took both environmental and behavioral contextual information as input and output next-step suggestions.

In Part IV, we implemented our methods into a visual analytic tool. The tool was named as VIT-PLA, which is short for Visual Interactive Tool for Process Log Analysis. In this tool, we proposed a prototype-based process data visualization strategy. The strategy can greatly reduce the data amount to visualize but preserve the characteristics of each process cluster. Statistical analyses were also implemented and visualized to help users better understand their process data.

Several challenges can be further explored in future work. First, the data amount is a limitation of our project and a common problem in the process mining community. For processes like the trauma resuscitation, there are no automated approaches that can accurately collect the activity logs. We made some efforts trying to collect such activity logs automatically by installing sensors (camera, microphone array and RFID tags), but the current system is not accurate enough. Manual coding of the activity logs can be tedious can greatly limit the data amount. Another reason that limits the data amount is confidentiality issues. Some of the process data, like ours, need strictly evaluated by institutional review board before usage. It poses risks to the privacy of both patients and medical teams. Hence, privacy-preserving process data sharing methods are desired for the process mining research community. Second, although we proposed the process recommender systems and evaluated using our datasets and case studies, we believe this is just a start for process recommender systems and there will be plenty of applications and improvements on such systems. Hopefully, we could see more process recommender systems studied and applied in real-world processes to help simplify and standardize the procedures in the near future.

References

[1] Yang, Sen, Moliang Zhou, Rachel Webman, JaeWon Yang, Aleksandra Sarcevic, Ivan Marsic, and Randall S. Burd. "Duration-aware alignment of process traces." In Industrial Conference on Data Mining, pp. 379-393. Springer, Cham, 2016.

[2] Yang, Sen, Xin Dong, Moliang Zhou, Xinyu Li, Shuhong Chen, Rachel Webman, Aleksandra Sarcevic, Ivan Marsic, and Randall S. Burd. "VIT-PLA: Visual Interactive Tool for Process Log Analysis." In KDD Workshop on Interactive Data Exploration and Analytics. 2016.

[3] Yang, Sen, Xin Dong, Leilei Sun, Yichen Zhou, Richard A. Farneth, Hui Xiong, Randall S. Burd, and Ivan Marsic. "A Data-driven Process Recommender Framework." In Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pp. 2111-2120. ACM, 2017.

[4] Yang, Sen, Moliang Zhou, Shuhong Chen, Xin Dong, Omar Z. Ahmed, Randall S. Burd, and Ivan Marsic. "Medical Workflow Modeling Using Alignment-Guided State-Splitting HMM." In Healthcare Informatics (ICHI), 2017 IEEE International Conference on, pp. 144-153. IEEE, 2017.

[5] Li, Jingyuan, Sen Yang, Shuhong Chen, Fei Tao, Ivan Marsic, and Randall S. Burd. "Discovering Interpretable Medical Workflow Models." In 2018 IEEE International Conference on Healthcare Informatics (ICHI), pp. 437-439. IEEE, 2018.

[6] Yang, Sen, Fei Tao, Jingyuan Li, Dawei Wang, Shuhong Chen, Omar Z. Ahmed, Ivan Marsic, and Randall S. Burd. "Process Mining the Trauma Resuscitation Patient Cohorts." In 2018 IEEE International Conference on Healthcare Informatics (ICHI), pp. 29-35. IEEE, 2018.

[7] Yang, Sen, Weiqing Ni, Xin Dong, Shuhong Chen, Richard A. Farneth, Aleksandra Sarcevic, Ivan Marsic, and Randall S. Burd. "Intention Mining in Medical Process: A Case Study in Trauma Resuscitation." In 2018 IEEE International Conference on Healthcare Informatics (ICHI), pp. 36-43. IEEE, 2018.

[8] Yang, Sen, Yichen Zhou, Yifeng Guo, Richard A. Farneth, Ivan Marsic, and Randall S. Burd. "Semi-Synthetic Trauma Resuscitation Process Data Generator." In Healthcare Informatics (ICHI), 2017 IEEE International Conference on, pp. 573-573. IEEE, 2017.

[9] Zhou, Moliang, Sen Yang, Xinyu Li, Shuyu Lv, Shuhong Chen, Ivan Marsic, Richard A. Farneth, and Randall S. Burd. "Evaluation of Trace Alignment Quality and its Application in Medical Process Mining." In Healthcare Informatics (ICHI), 2017 IEEE International Conference on, pp. 258-267. IEEE, 2017.

[10] Chen, Shuhong, Sen Yang, Moliang Zhou, Randall S. Burd, and Ivan Marsic. Process-oriented Iterative Multiple Alignment for Medical Process Mining. In2017 IEEE International Conference on Data Mining Workshops (ICDMW) 2017 Nov 1 (pp. 438-445). IEEE.

[11] Yang, Sen, Aleksandra Sarcevic, Richard A. Farneth, Shuhong Chen, Omar Z. Ahmed, Ivan Marsic, and Randall S. Burd. "An Approach to Automatic Process Deviation Detection in a Time-Critical Clinical Process." Journal of Biomedical Informatics.

[12] Stahl, Kenneth D., and Susan E. Brien. "Reducing patient errors in trauma care." Acute Care Surgery and Trauma Care. London, UK: Informa Health Care (2009): 268-277.

[13] Demetriades, Demetrios, Brian Kimbrell, Ali Salim, George Velmahos, Peter Rhee, Christy Preston, Ginger Gruzinski, and Linda Chan. "Trauma deaths in a mature urban trauma system: is “trimodal” distribution a valid concept?." Journal of the American College of Surgeons 201, no. 3 (2005): 343-348.

[14] Gruen, Russell L., Gregory J. Jurkovich, Lisa K. McIntyre, Hugh M. Foy, and Ronald V. Maier. "Patterns of errors contributing to trauma mortality: lessons learned from 2594 deaths." Annals of Surgery 244, no. 3 (2006): 371.

[15] Yang, Sen, Jingyuan Li, Xiaoyi Tang, Shuhong Chen, Ivan Marsic, and Randall S. Burd. "Process Mining for Trauma Resuscitation." The IEEE intelligent informatics bulletin 18, no. 1 (2017): 15.

[16] Subcommittee AT, Tchorz KM, International ATLS working group. Advanced trauma life support (ATLS®): the ninth edition. The Journal Of Trauma And Acute Care Surgery. 2013;74(5):1363.

[17] Sarcevic, Aleksandra, Ivan Marsic, and Randal S. Burd. "Teamwork errors in trauma resuscitation." ACM Transactions on Computer-Human Interaction (TOCHI) 19, no. 2 (2012): 13.

[18] Van der Aalst, Wil MP. "Process Discovery: An Introduction." In Process Mining, pp. 125-156. Springer, Berlin, Heidelberg, 2011.

[19] Augusto, Adriano, Raffaele Conforti, Marlon Dumas, Marcello La Rosa, Fabrizio M. Maggi, Andrea Marrella, Massimo Mecella, and Allar Soo. "Automated discovery of process models from event logs: Review and benchmark." IEEE Transactions on Knowledge and Data Engineering (2018).

[20] De Weerdt, Jochen, Manu De Backer, Jan Vanthienen, and Bart Baesens. "A multi-dimensional quality assessment of state-of-the-art process discovery algorithms using real-life event logs." Information Systems 37, no. 7 (2012): 654-676.

[21] Murata, Tadao. "Petri nets: Properties, analysis and applications." Proceedings of the IEEE 77, no. 4 (1989): 541-580.

[22] Rovani, Marcella, Fabrizio M. Maggi, Massimiliano de Leoni, and Wil MP van der Aalst. "Declarative process mining in healthcare." Expert Systems with Applications 42, no. 23 (2015): 9236-9251.

[23] Chinosi, Michele, and Alberto Trombetta. "BPMN: An introduction to the standard." Computer Standards & Interfaces 34, no. 1 (2012): 124-134.

[24] Adriansyah, Arya, Boudewijn F. van Dongen, and Wil MP van der Aalst. "Conformance checking using cost-based fitness analysis." In Enterprise Distributed Object Computing Conference (EDOC), 2011 15th IEEE International, pp. 55-64. IEEE, 2011.

[25] Fahland, Dirk, and Wil MP van der Aalst. "Model repair—aligning process models to reality." Information Systems 47 (2015): 220-243.

[26] Clarke, John R., Beverly Spejewski, Abigail S. Gertner, Bonnie L. Webber, Catherine Z. Hayward, Thomas A. Santora, David K. Wagner et al. "An objective analysis of process errors in trauma resuscitations." Academic Emergency Medicine 7, no. 11 (2000): 1303-1310.

[27] Takami, Jun-ichi, and Shigeki Sagayama. "A successive state splitting algorithm for efficient allophone modeling." In Acoustics, Speech, and Signal Processing, 1992. ICASSP-92., 1992 IEEE International Conference on, vol. 1, pp. 573-576. IEEE, 1992.

[28] Herbst, Joachim, and Dimitris Karagiannis. "Integrating machine learning and workflow

management to support acquisition and adaptation of workflow models." Intelligent Systems in Accounting, Finance & Management 9, no. 2 (2000): 67-92.

[29] Lu, Xixi, Dirk Fahland, Frank JHM van den Biggelaar, and Wil MP van der Aalst. "Handling duplicated tasks in process discovery by refining event labels." In International Conference on Business Process Management, pp. 90-107. Springer, Cham, 2016.

[30] Bose, RP Jagadeesh Chandra, and Wil MP van der Aalst. "Process diagnostics using trace alignment: opportunities, issues, and challenges." Information Systems 37, no. 2 (2012): 117-141.

[31] Rabiner, Lawrence R. "A tutorial on hidden Markov models and selected applications in speech recognition." Proceedings of the IEEE 77, no. 2 (1989): 257-286.

[32] Stolcke, Andreas, and Stephen M. Omohundro. "Best-first model merging for hidden Markov model induction." arXiv preprint cmp-lg/9405017 (1994). https://arxiv.org/abs/cmp-lg/9405017

[33] Blum, Tobias, Nicolas Padoy, Hubertus Feußner, and Nassir Navab. "Workflow mining for visualization and analysis of surgeries." International Journal of Computer Assisted Radiology and Surgery 3, no. 5 (2008): 379-386.

[34] Singer, Harald, and Mari Ostendorf. "Maximum likelihood successive state splitting." In Acoustics, Speech, and Signal Processing, 1996. ICASSP-96. Conference Proceedings., 1996 IEEE International Conference on, vol. 2, pp. 601-604. IEEE, 1996.

[35] Ostendorf, Mari, and Harald Singer. "HMM topology design using maximum likelihood successive state splitting." Computer Speech & Language 11, no. 1 (1997): 17-41.

[36] Mavromatis, Panayotis. "Minimum description length modelling of musical structure." Journal of Mathematics and Music 3, no. 3 (2009): 117-136.

[37] Siddiqi, Sajid M., Geoffrey J. Gordon, and Andrew W. Moore. "Fast state discovery for HMM model selection and learning." International Conference on Artificial Intelligence and Statistics. 2007.

[38] Levenshtein, Vladimir I. "Binary codes capable of correcting deletions, insertions, and reversals." In Soviet physics doklady, vol. 10, no. 8, pp. 707-710. 1966.

[39] Hsu, Pao-Lu, and Herbert Robbins. "Complete convergence and the law of large numbers." Proceedings of the National Academy of Sciences 33, no. 2 (1947): 25-31.

[40] Mans, Ronny S., M. H. Schonenberg, Minseok Song, Wil MP van der Aalst, and Piet JM Bakker. "Application of process mining in healthcare–a case study in a dutch hospital." In International Joint Conference on Biomedical Engineering Systems and Technologies, pp. 425-438. Springer, Berlin, Heidelberg, 2008.

[41] Kelleher, Deirdre C., Elizabeth A. Carter, Lauren J. Waterhouse, Samantha E. Parsons, Jennifer L. Fritzeen, and Randall S. Burd. "Effect of a checklist on advanced trauma life support task performance during pediatric trauma resuscitation." Academic Emergency Medicine 21, no. 10 (2014): 1129-1134.

[42] Shmueli, Galit. "To explain or to predict?." Statistical Science (2010): 289-310.

[43] Yu, Lu, Jason M. Schwier, Ryan M. Craven, Richard R. Brooks, and Christopher Griffin. "Inferring statistically significant hidden markov models." IEEE Transactions on Knowledge and Data Engineering 25, no. 7 (2013): 1548-1558.

[44] Günther, Christian W., and Wil MP Van Der Aalst. "Fuzzy mining–adaptive process

simplification based on multi-perspective metrics." In International Conference on Business Process Management, pp. 328-343. Springer, Berlin, Heidelberg, 2007.

[45] Van der Aalst, Wil MP, AK Alves De Medeiros, and A. J. M. M. Weijters. "Genetic process mining." In International Conference on Application and Theory of Petri Nets, pp. 48-69. Springer, Berlin, Heidelberg, 2005.

[46] Carter, Elizabeth A., Lauren J. Waterhouse, Mark L. Kovler, Jennifer Fritzeen, and Randall S. Burd. "Adherence to ATLS primary and secondary surveys during pediatric trauma resuscitation." Resuscitation 84, no. 1 (2013): 66-71.

[47] Fitzgerald, Mark, Peter Cameron, Colin Mackenzie, Nathan Farrow, Pamela Scicluna, Robert Gocentas, Adam Bystrzycki et al. "Trauma resuscitation errors and computer-assisted decision support." Archives of Surgery 146, no. 2 (2011): 218-225.

[48] Clarke, John R., Beverly Spejewski, Abigail S. Gertner, Bonnie L. Webber, Catherine Z. Hayward, Thomas A. Santora, David K. Wagner et al. "An objective analysis of process errors in trauma resuscitations." Academic Emergency Medicine 7, no. 11 (2000): 1303-1310.

[49] Wiegmann, Douglas A., Andrew W. ElBardissi, Joseph A. Dearani, Richard C. Daly, and Thoralf M. Sundt. "Disruptions in surgical flow and their relationship to surgical errors: an exploratory investigation." Surgery 142, no. 5 (2007): 658-665.

[50] Alvarez, Camilo, Eric Rojas, Michael Arias, Jorge Munoz-Gama, Marcos Sepúlveda, Valeria Herskovic, and Daniel Capurro. "Discovering role interaction models in the Emergency Room using Process Mining." Journal of Biomedical Informatics (2017).

[51] Rojas, Eric, Jorge Munoz-Gama, Marcos Sepúlveda, and Daniel Capurro. "Process mining in healthcare: A literature review." Journal of Biomedical Informatics 61 (2016): 224-236.

[52] Bouarfa, Loubna, and Jenny Dankelman. "Workflow mining and outlier detection from clinical activity logs." Journal of Biomedical Informatics 45, no. 6 (2012): 1185-1190.

[53] Caron, Filip, Jan Vanthienen, Kris Vanhaecht, Erik Van Limbergen, Jochen De Weerdt, and Bart Baesens. "Monitoring care processes in the gynecologic oncology department." Computers in Biology and Medicine 44 (2014): 88-96.

[54] Lu, Xixi, Dirk Fahland, Frank JHM van den Biggelaar, and Wil MP van der Aalst. "Detecting deviating behaviors without models." In International Conference on Business Process Management, pp. 126-139. Springer, Cham, 2015.

[55] Christov, Stefan C., George S. Avrunin, and Lori A. Clarke. "Online deviation detection for medical processes." In AMIA Annual Symposium Proceedings, vol. 2014, p. 395. American Medical Informatics Association, 2014.

[56] Kirchner, Kathrin, Nico Herzberg, Andreas Rogge-Solti, and Mathias Weske. "Embedding conformance checking in a process intelligence system in hospital environments." In Process Support and Knowledge Representation in Health Care, pp. 126-139. Springer, Berlin, Heidelberg, 2013.

[57] Swinnen, Jo, Benoît Depaire, Mieke J. Jans, and Koen Vanhoof. "A process deviation analysis–a case study." In International Conference on Business Process Management, pp. 87-98. Springer, Berlin, Heidelberg, 2011.

[58] Grando, María Adela, Wil MP Van Der Aalst, and Ronny S. Mans. "Reusing a declarative specification to check the conformance of different CIGs." In International Conference on

Business Process Management, pp. 188-199. Springer, Berlin, Heidelberg, 2011.

[59] Rozinat, Anne, and Wil MP Van der Aalst. "Conformance checking of processes based on monitoring real behavior." Information Systems 33, no. 1 (2008): 64-95.

[60] Oakley, Ed, Sergio Stocker, Georg Staubli, and Simon Young. "Using video recording to identify management errors in pediatric trauma resuscitation." Pediatrics 117, no. 3 (2006): 658-664.

[61] Webman, Rachel, Jennifer Fritzeen, JaeWon Yang, Grace F. Ye, Paul C. Mullan, Faisal G. Qureshi, Sarah H. Parker, Aleksandra Sarcevic, Ivan Marsic, and Randall S. Burd. "Classification and team response to non-routine events occurring during pediatric trauma resuscitation." The Journal of Trauma and Acute Care Surgery 81, no. 4 (2016): 666.

[62] Houshian, Shirzad, Morten S. Larsen, and Carsten Holm. "Missed injuries in a level I trauma center." Journal of Trauma and Acute Care Surgery 52, no. 4 (2002): 715-719.

[63] Van Dongen, Boudewijn F., Ana Karla A. de Medeiros, H. M. W. Verbeek, A. J. M. M. Weijters, and Wil MP Van Der Aalst. "The ProM framework: A new era in process mining tool support." In International Conference on Application and Theory of Petri Nets, pp. 444-454. Springer, Berlin, Heidelberg, 2005.

[64] Ward Jr, Joe H. "Hierarchical grouping to optimize an objective function." Journal of the American Statistical Association 58, no. 301 (1963): 236-244.

[65] Liu, Yanchi, Zhongmou Li, Hui Xiong, Xuedong Gao, and Junjie Wu. "Understanding of internal clustering validation measures." In Data Mining (ICDM), 2010 IEEE 10th International Conference on, pp. 911-916. IEEE, 2010.

[66] Clay-Williams, Robyn, Jeanette Hounsgaard, and Erik Hollnagel. "Where the rubber meets the road: using FRAM to align work-as-imagined with work-as-done when implementing clinical guidelines." Implementation Science 10, no. 1 (2015): 125.

[67] Kleinman, Nathan, Safiya Abouzaid, Lenae Andersen, Zhixiao Wang, and Annette Powers. "Cohort analysis assessing medical and nonmedical cost associated with obesity in the workplace." Journal of Occupational and Environmental Medicine 56, no. 2 (2014): 161-170.

[68] Delgado, João, Jane AH Masoli, Kirsty Bowman, W. David Strain, George A. Kuchel, Kate Walters, Louise Lafortune, Carol Brayne, David Melzer, and Alessandro Ble. "Outcomes of treated hypertension at age 80 and older: Cohort analysis of 79,376 individuals." Journal of the American Geriatrics Society 65, no. 5 (2017): 995-1003.

[69] Nelson, Jana, Adrian T. Billeter, Burkhardt Seifert, Valentin Neuhaus, Otmar Trentz, Christoph K. Hofer, and Matthias Turina. "Obese trauma patients are at increased risk of early hypovolemic shock: a retrospective cohort analysis of 1,084 severely injured patients." Critical Care 16, no. 3 (2012): R77.

[70] Kelleher, Deirdre C., RP Jagadeesh Chandra Bose, Lauren J. Waterhouse, Elizabeth A. Carter, and Randall S. Burd. "Effect of a checklist on advanced trauma life support workflow deviations during trauma resuscitations without pre-arrival notification." Journal of the American College of Surgeons 218, no. 3 (2014): 459-466.

[71] Hand, David J., Heikki Mannila, and Padhraic Smyth. Principles of data mining (adaptive computation and machine learning). Cambridge, MA: MIT press, 2001.

[72] Cormen, Thomas H. Introduction to algorithms. MIT press, 2009.

[73] MacQueen, James. "Some methods for classification and analysis of multivariate observations." In Proceedings of the Fifth Berkeley Symposium on Mathematical Statistics and Probability, vol. 1, no. 14, pp. 281-297. 1967.

[74] Rokach, Lior, and Oded Maimon. "Clustering methods." In Data mining and knowledge discovery handbook, pp. 321-352. Springer, Boston, MA, 2005.

[75] Zaki, Mohammed J. "SPADE: An efficient algorithm for mining frequent sequences." Machine Learning 42, no. 1-2 (2001): 31-60.

[76] De Winter, Joost CF. "Using the Student's t-test with extremely small sample sizes." Practical Assessment, Research & Evaluation 18, no. 10 (2013).

[77] Shlens, Jonathon. "A tutorial on principal component analysis." arXiv preprint arXiv:1404.1100 (2014). https://arxiv.org/abs/1404.1100

[78] Clarke, John R., Catherine Z. Hayward, Thomas A. Santora, David K. Wagner, and Bonnie L. Webber. "Computer-generated trauma management plans: comparison with actual care." World Journal of Surgery 26, no. 5 (2002): 536-538.

[79] Aghabozorgi, Saeed, Ali Seyed Shirkhorshidi, and Teh Ying Wah. "Time-series clustering–A decade review." Information Systems 53 (2015): 16-38.

[80] Huaulmé, Arnaud, Sandrine Voros, Laurent Riffaud, Germain Forestier, Alexandre Moreau-Gaudry, and Pierre Jannin. "Distinguishing surgical behavior by sequential pattern discovery." Journal of Biomedical Informatics 67 (2017): 34-41.

[81] Liu, Chuanren, Fei Wang, Jianying Hu, and Hui Xiong. "Temporal phenotyping from longitudinal electronic health records: A graph based framework." In Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pp. 705-714. ACM, 2015.

[82] Forestier, Germain, Florent Lalys, Laurent Riffaud, Brivael Trelhu, and Pierre Jannin. "Classification of surgical processes using dynamic time warping." Journal of Biomedical Informatics 45, no. 2 (2012): 255-264.

[83] Forestier, Germain, Francois Petitjean, Laurent Riffaud, and Pierre Jannin. "Non-linear temporal scaling of surgical processes." Artificial Intelligence in Medicine 62, no. 3 (2014): 143-152.

[84] Jung, Jae-Yoon, Joonsoo Bae, and Ling Liu. "Hierarchical clustering of business process models." International Journal of Innovative Computing, Information and Control 5, no. 12 (2009): 1349-4198.

[85] Liu, Chuanren, Kai Zhang, Hui Xiong, Guofei Jiang, and Qiang Yang. "Temporal skeletonization on sequential data: patterns, categorization, and visualization." IEEE Transactions on Knowledge and Data Engineering 28, no. 1 (2016): 211-223.

[86] Bose, RP Jagadeesh Chandra, and Wil MP Van der Aalst. "Context aware trace clustering: Towards improving process mining results." In Proceedings of the 2009 SIAM International Conference on Data Mining, pp. 401-412. Society for Industrial and Applied Mathematics, 2009.

[87] Frey, Brendan J., and Delbert Dueck. "Clustering by passing messages between data points." Science 315, no. 5814 (2007): 972-976.

[88] Rodriguez, Alex, and Alessandro Laio. "Clustering by fast search and find of density peaks." Science 344, no. 6191 (2014): 1492-1496.

[89] Ward Jr, Joe H. "Hierarchical grouping to optimize an objective function." Journal of the

American Statistical Association 58, no. 301 (1963): 236-244.

[90] Wasserman, Larry. All of statistics: a concise course in statistical inference. Springer Science & Business Media, 2013.

[91] He, Haibo, and Edwardo A. Garcia. "Learning from imbalanced data." IEEE Transactions on Knowledge and Data Engineering 21, no. 9 (2009): 1263-1284.

[92] Fournier-Viger, Philippe, Antonio Gomariz, Manuel Campos, and Rincy Thomas. "Fast vertical mining of sequential patterns using co-occurrence information." In Pacific-Asia Conference on Knowledge Discovery and Data Mining, pp. 40-52. Springer, Cham, 2014.

[93] Clay-Williams, Robyn, Jeanette Hounsgaard, and Erik Hollnagel. "Where the rubber meets the road: using FRAM to align work-as-imagined with work-as-done when implementing clinical guidelines." Implementation Science 10, no. 1 (2015): 125.

[94] Demetriades, Demetrios, Brian Kimbrell, Ali Salim, George Velmahos, Peter Rhee, Christy Preston, Ginger Gruzinski, and Linda Chan. "Trauma deaths in a mature urban trauma system: is “trimodal” distribution a valid concept?." Journal of the American College of Surgeons 201, no. 3 (2005): 343-348.

[95] Bernhard, Michael, Torben K. Becker, Tim Nowe, Marko Mohorovicic, Marcus Sikinger, Thorsten Brenner, Goetz M. Richter et al. "Introduction of a treatment algorithm can improve the early management of emergency patients in the resuscitation room." Resuscitation 73, no. 3 (2007): 362-373.

[96] Yu, Feng, Qiang Liu, Shu Wu, Liang Wang, and Tieniu Tan. "A dynamic recurrent model for next basket recommendation." In Proceedings of the 39th International ACM SIGIR conference on Research and Development in Information Retrieval, pp. 729-732. ACM, 2016.

[97] Lu, Jie, Dianshuang Wu, Mingsong Mao, Wei Wang, and Guangquan Zhang. "Recommender system application developments: a survey." Decision Support Systems 74 (2015): 12-32.

[98] Choi, Edward, Mohammad Taha Bahadori, Andy Schuetz, Walter F. Stewart, and Jimeng Sun. "Doctor ai: Predicting clinical events via recurrent neural networks." In Machine Learning for Healthcare Conference, pp. 301-318. 2016.

[99] Sun, Leilei, Chuanren Liu, Chonghui Guo, Hui Xiong, and Yanming Xie. "Data-driven automatic treatment regimen development and recommendation." In Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pp. 1865-1874. ACM, 2016.

[100] Gröger, Christoph, Holger Schwarz, and Bernhard Mitschang. "Prescriptive analytics for recommendation-based business process optimization." In International Conference on Business Information Systems, pp. 25-37. Springer, Cham, 2014.

[101] Mikolov, Tomas, Kai Chen, Greg Corrado, and Jeffrey Dean. "Efficient estimation of word representations in vector space." arXiv preprint arXiv:1301.3781 (2013). https://arxiv.org/abs/1301.3781

[102] Hochreiter, Sepp, and Jürgen Schmidhuber. "Long short-term memory." Neural Computation 9, no. 8 (1997): 1735-1780.

[103] Cho, Kyunghyun, Bart Van Merriënboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. "Learning phrase representations using RNN encoder-decoder for statistical machine translation." arXiv preprint arXiv:1406.1078 (2014).

https://arxiv.org/abs/1406.1078

[104] Luong, Minh-Thang, Hieu Pham, and Christopher D. Manning. "Effective approaches to attention-based neural machine translation." arXiv preprint arXiv:1508.04025 (2015). https://arxiv.org/abs/1508.04025

[105] Bahdanau, Dzmitry, Kyunghyun Cho, and Yoshua Bengio. "Neural machine translation by jointly learning to align and translate." arXiv preprint arXiv:1409.0473 (2014). https://arxiv.org/abs/1409.0473

[106] Zhou, Peng, Wei Shi, Jun Tian, Zhenyu Qi, Bingchen Li, Hongwei Hao, and Bo Xu. "Attention-based bidirectional long short-term memory networks for relation classification." In Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers), vol. 2, pp. 207-212. 2016.

[107] Ma, Fenglong, Radha Chitta, Jing Zhou, Quanzeng You, Tong Sun, and Jing Gao. "Dipole: Diagnosis prediction in healthcare via attention-based bidirectional recurrent neural networks." In Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pp. 1903-1911. ACM, 2017.

[108] Cramér, Harald. Mathematical methods of statistics (PMS-9). Vol. 9. Princeton University Press, 2016.

[109] Demirtas, Hakan, and Beyza Doganay. "Simultaneous generation of binary and normal data with specified marginal and association structures." Journal of Biopharmaceutical Statistics 22, no. 2 (2012): 223-236.

[110] Steck, Harald. "Item popularity and recommendation accuracy." In Proceedings of the Fifth ACM Conference on Recommender Systems, pp. 125-132. ACM, 2011.

[111] Lee, Jongwuk, Dongwon Lee, Yeon-Chang Lee, Won-Seok Hwang, and Sang-Wook Kim. "Improving the accuracy of top-N recommendation using a preference model." Information Sciences 348 (2016): 290-304.

[112] Gagniuc, Paul A. Markov Chains: From Theory to Implementation and Experimentation. John Wiley & Sons, 2017.

[113] Van der Aalst, Wil, Ton Weijters, and Laura Maruster. "Workflow mining: Discovering process models from event logs." IEEE Transactions on Knowledge and Data Engineering 16, no. 9 (2004): 1128-1142.

[114] Monroe, Megan, Rongjian Lan, Hanseung Lee, Catherine Plaisant, and Ben Shneiderman. "Temporal event sequence simplification." IEEE Transactions on Visualization and Computer Graphics 19, no. 12 (2013): 2227-2236.

[115] Malik, Sana, Fan Du, Megan Monroe, Eberechukwu Onukwugha, Catherine Plaisant, and Ben Shneiderman. "Cohort comparison of event sequences with balanced integration of visual analytics and statistics." In Proceedings of the 20th International Conference on Intelligent User Interfaces, pp. 38-49. ACM, 2015.

[116] Wongsuphasawat, Krist, and David Gotz. "Exploring flow, factors, and outcomes of temporal event sequences with the outflow visualization." IEEE Transactions on Visualization and Computer Graphics 18, no. 12 (2012): 2659-2668.

[117] Perer, Adam, and Fei Wang. "Frequence: Interactive mining and visualization of temporal frequent event sequences." In Proceedings of the 19th International Conference on Intelligent User

Interfaces, pp. 153-162. ACM, 2014.

[118] Perer, Adam, Fei Wang, and Jianying Hu. "Mining and exploring care pathways from electronic medical records with visual analytics." Journal of Biomedical Informatics 56 (2015): 369-378.

[119] Song, Minseok, and Wil MP van der Aalst. "Supporting process mining by showing events at a glance." In Proceedings of the 17th Annual Workshop on Information Technologies and Systems (WITS), pp. 139-145. 2007.

[120] Jain, Anil K., and Richard C. Dubes. "Algorithms for clustering data." (1988) https://homepages.inf.ed.ac.uk/rbf/BOOKS/JAIN/Clustering_Jain_Dubes.pdf.

[121] Tan, Pang-Ning. Introduction to data mining. Pearson Education India, 2006.

[122] Czepiel, Scott A. "Maximum likelihood estimation of logistic regression models: theory and implementation." Available at czep. net/stat/mlelr. pdf (2002).

[123] Burattin, Andrea, and Alessandro Sperduti. "PLG: A framework for the generation of business process models and their execution logs." In International Conference on Business Process Management, pp. 214-219. Springer, Berlin, Heidelberg, 2010.