# CONTEXT-AWARE PROCESS RECOMMENDATION SYSTEM FOR MEDICAL TREATMENT

by

WEIQING NI

A thesis submitted to the

School of Graduate Studies

Rutgers, The State University of New Jersey

In partial fulfillment of the requirements

For the degree of

Master of science

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 THESIS

# CONTEXT-AWARE PROCESS RECOMMENDATION SYSTEM FOR MEDICAL TREATMENT

By Weiqing Ni
Thesis Director: Dr. Ivan Marsic

AI-based recommendation systems are widely utilized in different fields including movies, music, news, social tags and products in general. Such systems may help reduce medical team errors and improve patient outcomes in treatment processes (e.g., trauma resuscitation, surgical processes) by extracting knowledge from historic data and providing online recommendations. We developed data-driven process recommender systems for trauma resuscitations process based on different models. This thesis includes three main topics: (1) process data augmentation algorithms; (2) two intention mining models; and (3) two process recommender systems. Topic (1) and (2) were developed for improving the performance of recommender systems (Topic (3)).

Our process data was collected manually by medical experts reviewing the recorded videos. The data collection was labor intensive and we coded 123 trauma patient records in the past four years. Because of the small size of our dataset, we attempted to augment it by generating synthetic data. We developed two synthetic data generators to augment our dataset: (1) alignment-based process data generator and (2) sequential generative adversarial network. Both of them can generate large amounts of semi-synthetic process data that has similar characteristics with those of real-world process data.

We used intention mining models to discover the relationship between observed treatment activities and medical team's underlying intentions. By identifying medical team's intentions, we are able to generate accurate recommendations. We developed two different intention mining algorithms, one based on Hidden Markov Models and the other based on Seq2seq models.

Last, we designed the process recommendation systems using two different models, (1) Hierarchical Hidden Markov Model (HHMM) and (2) Long Short-Term Memory (LSTM). The HHMM-based recommender system utilizes the intention mining algorithm to estimate the medical team's current intention, and then provides the process recommendation identified in that intention category. On the other hand, the LSTM-based recommender system learns the relationships from different processes. And also, the LSTM model was modified to deal with both environmental (i.e., patient demographics) and behavioral (i.e., preceding treatment activities) contextual information. To provide the process recommendation, the LSTM is iterated over the previous process trace, and uses the most likely activity as the next-step recommending process. For HHMM-based recommender system, we achieved top-1 accuracy at 34.4% and top-5 accuracy 56.9% over 102 kinds of activities. The LSTM-based recommender system showed a higher top-1 accuracy at 39.9% and top-5 accuracy 65.5%. The experimental results indicated both of out recommender systems (HHMM & LSTM) outperforms baseline models in recommendation accuracy, demonstrating the feasibility of our context-aware process recommendation systems for complex real-world medical processes.

# ACKNOWLEDGEMENTS

First and foremost, I would like to express my deepest gratitude to my advisor, Prof. Ivan Marsic, for his guidance and support over my one-year research. His great encouragements, inspirations, advice and feedback lead me to the thorough understanding of my research. I also enjoyed the freedom he gave me and benefited from the independent research opportunities.

And also, during my graduate school journal at Rutgers, I received tremendous help from my lab colleagues and friends. I would like to thank Sen Yang who is not only the tutor of my research but also a great friend shared his experience and advice on matters outside of school. I would like to thank my lab colleagues Xin Dong, Jingyuan Li, Fei Tao and Yanhao Wang. I would also like to thank all friends who have accompanies me along the journey.

Outside of Rutgers University, I would like to extend my gratitude to my mentor Jin Cao and collaborators Ran He, Wangsu Hu at Nokia Bell Labs. My internship at Bell Labs was one of the best research experiences that I had during graduate school. I benefited a lot from the collaboration, and from them I gained a clear idea about what my future career will be.

I would like to thank Prof. Shantenu Jha and Prof. Gajic Zoran, for their time to serve on my committee and provide me with suggestions on improving the dissertation.

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

# TABLE OF CONTENTS

Page

# LIST OF TABLES

Table Page

LIST OF FIGURES

Figure Page

# CHAPTER 1

# INTRODUCTION

## 1.1 Motivation

Medical teams make unavoidable errors in the 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 [1]. During trauma resuscitations, 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. Despite the use of standardized protocols for establishing treatment and management goals, deviations from these protocols are observed in up to 85% of trauma resuscitations [2]. Although most deviations are variations resulting from the flexibility or adaptability needed for managing patients with different injuries, other deviations represent significant errors that may contribute to adverse patient outcomes [3]. To reduce the chances of errors, our research explores how to remind the clinical doctors during the medical processes by data-driven recommender system. The recommender system built on artificial intelligence (AI) and data mining techniques will provide the medical team leader (or surgical coordinator) with next-step treatment recommendations through the wall displays.

## 1.2 Synthetic Data Generator

Process mining techniques have been applied to the visualization, interpretation and analysis of medical processes [4]. However, only limited process data is publicly available, especially in the medical field. Our process data was collected manually by medical experts reviewing the recorded videos. The data collection was labor intensive and we coded 123 trauma patient records in the past four years. Therefore, we need to generate synthetic trauma resuscitation process data. We achieve this goal in two different approaches: 1. Alignment-based Process Data Generator and 2. Sequential Generative Adversarial Network. Both of them can generate large amounts of semi-synthetic process data that has similar characteristics with those of real-world process data.

### 1.3 Intention Mining Algorithms

Intentions are the thoughts directed towards achieving process goals. We can infer people's intentions from their activities. For example, during trauma resuscitations, "maintain oxygenation" is a goal. Activities for addressing this goal may include "placing oxygen" or "placing an oxygen saturation monitor". The team may have the goal of "maintaining oxygenation", but not have the intention of fulfilling this goal for several minutes. When they do intend to satisfy this goal, their intention may be identified by observing the two oxygen-related activities. Previous workflow analyses in process mining have focused on simple mining of the patterns of observed activities. They did not attempt to understand the hidden (or unobservable) intentions underlying the observed processes. In addition, most previous research has focused on simple processes that include a limited number of activities. Performing these analyses in medical settings is more challenging because of the complex concurrency of associated activities. Knowing the intentions behind the activities can help simplify modeling of complex processes and provide accurate recommendations.

Intention mining has not been deployed in clinical settings. No previous research has tried to identify the intentions of a medical team during a clinical process. Hence, there is no established intention model of the trauma resuscitation process that could be used to supervise the intention mining. Although it is possible to manually perform intention mining, this approach is vulnerable to subjective bias. Experienced observers may bring useful domain knowledge to this analysis, but may also be prone to identifying only familiar or expected patterns while neglecting others. Careful review of activity lists and patient features could mitigate this bias, but require time and labor commitment. For this reason, a data-driven approach to intention discovery is more attractive.

### 1.4 Context-aware Process Recommendation System

Our application differs from traditional recommendation problems in two ways. Firstly, temporal information plays a much more important role. Some treatment activities have temporal correlations, i.e., the secondary survey of the trauma resuscitation usually follows a head-to-toe examination. Secondly, different patients need different medical treatment procedures. The medical team must act according to the different patient conditions (e.g., injury area and severity). For this reason, it is important to incorporate context attributes into the recommender system for prescriptive analytics [5]. There are two types of contextual information: environmental and behavioral. Behavioral context refers treatment workflow. It is the activities performed and the order of their performance. Environmental context can be further divided into two categories: static and dynamic. Static context is a set of features of the patient or resuscitation that is present when the patient arrives and does not change. Examples are time of day, age of patient, and mechanism of injury. Dynamic context is a feature that changes as treatment goes on. For example, attributes of the activities, e.g., descriptor and completeness of an activity, are dynamic context.

We designed process recommendation system based on different models: Hierarchical Hidden Markov Model and LSTM network.

For the Hidden Markov Model (HMM) approach, we first perform unsupervised intention discovery by finding patterns in the observed activities. Our intention discovery is based on the fact that observed activities are correlated with hidden intentions. We also assume that activities caused by the same intention will be associated with each other. We used an HMM inference algorithm to extract these associations automatically. In addition, because no general criteria are available to assess the quality of trauma resuscitations across patients with various injuries and conditions, our process recommendation is based on the assumption that the "average" process enactment is more effective and valid than those deviating from the average. The key components of this system include: mining intentions from historic data, integrating contextual information into the recommender system, and finding the "average" enactment.

For the LSTM approach, we modified the RNN to receive and incorporate patient demographics as auxiliary inputs to the network. The main 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, can be labor-intensive. Over two years, we coded 122 resuscitation cases, but our data is still too small to training a deep neural network. We attempt to address this limitation by pre-training the neural network with synthetic data.

# CHAPTER 2

# SYNTHETIC DATA GENERATOR

Process mining techniques have been applied to the visualization, interpretation, and analysis of medical processes. However, only a very limited amount of process data necessary for these analyses is publicly available, especially in the medical field. This limits novel medical process research to using insufficiently large or randomly-generated synthetic datasets. Here a model is needed to generate large amounts of synthetic process data which is trained by using a limited amount of observed data. The generated data has similar proprieties comparing to the real data, and could potentially be observed in reality. In this study, we tried to use two different methods to implement the data generator: trace alignment and generative adversarial network.

## 2.1 Alignment-based Process Data Generator

Many of today's information systems are recording an abundance of event logs. Process mining techniques attempt to extract non-trivial knowledge and interesting insights from these event logs and to exploit these for further analysis [6][32].

### 2.1.1 Pair-wise Trace Alignment

In this section, trace alignment is formally defined and the techniques for finding optimal alignments will be discussed. Firstly, the notations used in description of trace alignment algorithm are listed:

- $\sum$ is the set of activities. $|\sum|$ denotes the number of the activities.
- $\sum^+$ is the set of all sequences of activities from $\sum$ . $T \in \sum^+$ is a trace over $\sum$. $|T|$ denotes the length of trace T.

- $T(K)$ represents the $k^{th}$ activity in the trace, and $T^n$ represents the $n$ length prefix of $T$, which means $T^n = T^{n-1}T(n)$.

The trace alignment over a set of traces $T = \{T_1, T_2, \dots, T_n\}$ is defined as a mapping of the set of traces in $T$ to another set of traces $\bar{T} = \{\bar{T}_1, \bar{T}_2, \dots, \bar{T}_n\}$ where each $\bar{T}_\imath \in (\sum \cup \{-\})^+$ for $1 \leq i \leq n$ and :

- $|\bar{T}_1| = |\bar{T}_2| = \cdots = |\overline{T_n}| = m$, where *m* is the length of the alignment.
- $\bar{T}_\imath$ by removing all "—" gap symbols is equal to $|T_i|$.

An alignment over a set of traces can be represented as a matrix $\mathcal{A}(T) = \{a_{ij}\}$. Here shows an example of aligning several traces.

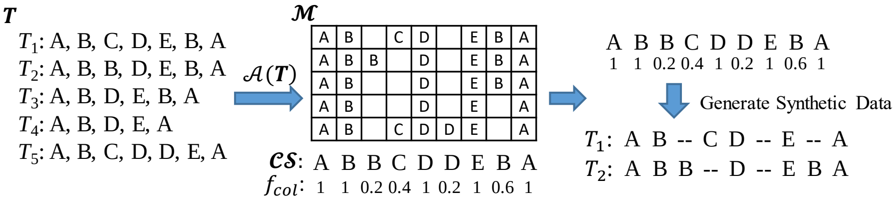

$\boldsymbol{T}$
$T_1$: A, B, C, D, E, B, A
$T_2$: A, B, B, D, E, B, A
$T_3$: A, B, D, E, B, A
$T_4$: A, B, D, E, A
$T_5$: A, B, C, D, D, E, A
$\mathcal{A}(\boldsymbol{T})$
$\mathcal{M}$
A B C D E B A
A B B D E B A
A B D E B A
A B D E A
A B C D D E A
$\mathcal{CS}$: A B B C D D E B A
$f_{col}$: 1 1 0.2 0.4 1 0.2 1 0.6 1
A B B C D D E B A
1 1 0.2 0.4 1 0.2 1 0.6 1
Generate Synthetic Data
$T_1$: A B -- C D -- E -- A
$T_2$: A B B -- D -- E B A

**Figure 1 Data Generator Based on Trace Alignment**

Here is the method of computing alignments based on a dynamic programming algorithm for finding the optimal alignment between sequences. A matrix *F* indexed by *i* and *j* is constructed and *F(i, j)* is the score of the best alignment. *F(0, 0)* is initialized to 0 and we can get the rest values by:

$$F(i,j) = max \begin{cases} F(i-1, j-1) + S\big(T_1(i), T_2(j)\big), & T(i) \rightarrow \ T(j) \\ F(i-1, j) + I\big(T_1(i), T_1(i-1)\big), & T_1(i) \rightarrow \ ' - ' \\ F(i, j-1) + I\big(T_2(j), T_2(j-1)\big), & T_2(j) \rightarrow \ ' - ' \end{cases} \qquad \textbf{(1)}$$

Where *S(a, b)* denotes the score for substitution of activity *a* with activity *b*, and *I(a, b)* denotes the score for inserting activity *a* given that the left activity is *b* which $I(a, -) =$

$I(-,a) = I(-,-) = 0$ for all $a \in \sum$. The bottom right cell of the matrix $F(|T_1|,|T_2|)$ is the best alignment score of trace $T_1$ and $T_2$. To get the alignment $\mathcal{A}(\boldsymbol{T})$**,** we backtrack from the bottom right on how the values were derived and stop at the start of the *matrix*, $i = j = 0$.

### 2.1.2 Synthetic Data Generator Based on Trace Alignment

Our synthetic data generator is backed by trace alignment, of aligning traces in an event log and shows the promise of such an approach in process diagnostics addressing some of the questions enumerated above [7]. Given process traces $\boldsymbol{T}$, the trace alignment algorithm $\mathcal{A}(\boldsymbol{T})$ forms an alignment matrix M with the traces in $\boldsymbol{T}$ 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 1). $\mathcal{A}(\boldsymbol{T})$ also returns the consensus sequence $\boldsymbol{\mathcal{CS}}$, a sequence that records the activity in each column of the alignment matrix. Our synthetic generator first calculates the alignment matrix and integrates each activity in the matrix with its associated environmental context attributes. Then we compress the 2D alignment matrix in to a 1D consensus sequence. Lastly, we try to reconstruct the 2D dataset using the consensus sequence. The reconstructed synthetic data will retain most of authentic data's characteristics. Throughout, we introduce noise to vary the synthetic data from the authentic data, helping the model generalize better to unseen data.

**Table 1 Algorithm of Synthetic Data Generator based on Trace Alignment**

| **Algorithm:** | **Synthetic Patient Record Generator Based on Trace Alignment** |
|---|---|
| **Input:** $\boldsymbol{r} = \{\boldsymbol{id}, \boldsymbol{x}, \boldsymbol{T}\}$<br>**Output:** $r^S$ | /* historic patient ids, patient attributes and treatment traces */<br>/* A synthetic patient record */ |
| Step 1. | Calculate alignment matrix $\{\boldsymbol{\mathcal{M}}, \boldsymbol{\mathcal{CS}}\} = \mathcal{A}(\boldsymbol{T})$ |
| Step 2. | **for** each column *col* in alignment matrix $\boldsymbol{\mathcal{M}}$: |

```
Step 3.      Initialize $f_{col} = 0$, $\boldsymbol{B}_{col} = \emptyset$, $\boldsymbol{X}_{col} = \emptyset$
Step 4.      Calculate column frequency: $f_{col}$ = num(non-gap cells)/num(rows)
Step 5.      **for** each non-gap activity $a$ in *col*:
Step 6.          Let $\boldsymbol{b}$ as the activity attributes of $a$ and $\boldsymbol{x}$ as patient attributes associated with $a$
Step 7.          $\boldsymbol{B}_{col} = \boldsymbol{B}_{col} \cup \boldsymbol{b}$;  $\boldsymbol{X}_{col} = \boldsymbol{X}_{col} \cup \boldsymbol{x}$
Step 8.      The prob. distribution of activity attributes is $\boldsymbol{b}_{col}$ = avg($\boldsymbol{B}_{col}$)
Step 9.      The prob. distribution of patient attributes is $\boldsymbol{x}_{col}$ = avg($\boldsymbol{X}_{col}$)
Step 10.     $\boldsymbol{\mathcal{CS}}$[col] = $\{a, f_{col}, \boldsymbol{b}_{col}, \boldsymbol{x}_{col}\}$
Step 11.  Initialize $\boldsymbol{x}^{s} = \emptyset$, $\boldsymbol{T}^{s} = \emptyset$, n=0
Step 12.  **for** *i* in range(size($\boldsymbol{\mathcal{CS}}$))
Step 13.     Let activity $a = \boldsymbol{\mathcal{CS}}$[i][0],
             $f_{col} = \boldsymbol{\mathcal{CS}}$[i][1],
             $\boldsymbol{b}_{col} = \boldsymbol{\mathcal{CS}}$[i][2],
             $\boldsymbol{x}_{col} = \boldsymbol{\mathcal{CS}}$[i][3]
Step 14.     **if** rand() < $f_{col}$
Step 15.         Randomly generate activity attributes $\boldsymbol{b}$ based on $\boldsymbol{b}_{col}$
Step 16.         $\boldsymbol{T}^{s} = \boldsymbol{T}^{s} \cup \{a, \boldsymbol{b}\}$
Step 17.         $\boldsymbol{x}^{s} = \boldsymbol{x}^{s} + \boldsymbol{x}_{col}$;   n++
Step 18.     **else continue**
Step 19.  Randomly generate $\boldsymbol{x}^{s}$ based on $\boldsymbol{x}^{s}$/n
Step 20.  **return** $r^{s} = \{0, \boldsymbol{x}^{s}, \boldsymbol{T}^{s}\}$
```

## 2.2 Generative Adversarial Network

Recently, recurrent neural networks (RNNs) with long-short-term-memory(LSTM) cells have shown excellent performance ranging from natural language generation to handwriting generation [8]. GANs were proposed as a training methodology to generative models where the training procedure is a minimax game between a generative model and a discriminative model [9]. On the other hand, a lot of efforts have been made to generate structured sequences. Recurrent neural networks can be trained to produce sequences of tokens in many applications such as machine translation. The sequence data generation can be formulated as a sequential decision-making process, which can be potentially be solved by reinforcement learning techniques [10]. In this section, we will introduce the sequence

GAN model which extends the GANs with RL-based generator to solve the sequence generation problems.

### 2.2.1 Structure of Sequence GAN

The generative adversarial networks are usually structured in two parts: generator and discriminator, which is showed in Figure 2. Given a dataset of real-world sequences, train a generative model $G_\theta$ to produce a sequence $Y_{1:T} = (y_1, y_2, \dots y_T), y_t \in \Upsilon$ , where $\Upsilon$ is the vocabulary of the set of tokens. In timestep $t$, the state $s$ is the current generated tokens $(y_1, y_2, \dots y_{t-1})$ and the action $a$ is the next token $y_t$ to select. Also, we need to train a discriminator $D_\phi$ to improve the generator $G_\theta$. $D_\phi(Y_{1:T})$ is a probability indicates how likely the sequence $Y_{1:T}$ is a real-world sequence. The discriminator $D_\phi$ is trained over the real-world sequences and synthesis sequences generated from the generator $G_\theta$. At the same time, the generative model $G_\theta$ is updated by applying a policy gradient and Monte-Carlo search on the basis of the expected and reward got from the discriminator $D_\phi$ [11].

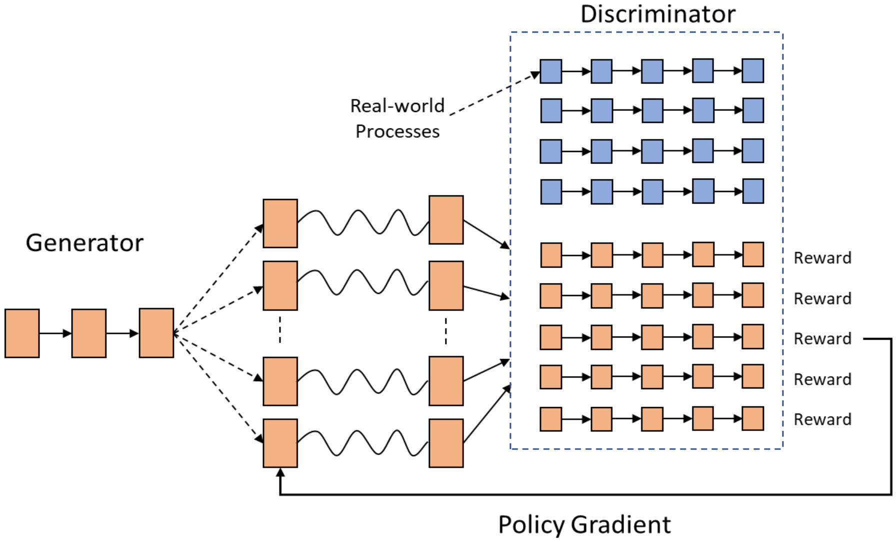


**Figure 2 The illustration of SeqGAN**

### 2.2.2 Policy Gradient

The objective of the generator (policy) $G_\theta$ is to generator a sequence from the start state $s_0$ to maximize its expected and reward.

$$J(\theta) = \mathbb{E}[R_T | s_0, \theta) = \sum_{y_1 \in Y} G_\theta(y_1 \mid s_0) \cdot Q_{D_\phi}^{G_\theta}(s_0, y_1) \qquad \textbf{(2)}$$

Where $R_T$ is the reward for a complete sequence obtained from the discriminator $D_\phi$. $Q_{D_\phi}^{G_\theta}(s, a)$ is the action-value function of a sequence, which is the excepted accumulative reward starting from state *s*, taking action *a*, and then following policy $G_\theta$. As described above, we get the value from the discriminator $D_\phi$, and thus we have:

$$Q_{D_\phi}^{G_\theta}\left(s = Y_{1:\mathrm{t}-1}, a = y_t\right)$$

$$= \begin{cases} \frac{1}{N} \sum_{n=1}^{N} D_\phi(Y_{1:T}^n), \quad Y_{1:T}^N \in MC^{G_\beta}(Y_{1:t}, N) & for\ t < T \\ D_\phi(Y_{1:T}) & for\ t = T \end{cases} \qquad \textbf{(3)}$$

In summary, Table 2 shows full details of the process. First, we pre-train the generator using maximum likelihood estimation on dataset *S*. And then the generator and discriminator trained alternatively.

**Table 2 Algorithm of Synthetic Data Generator Based on GAN**

| **Algorithm: Sequence Generative Adversarial Nets** | |
|---|---|
| **Input:** $\boldsymbol{S = \{id, T\}}$ /* historic patient ids and treatment traces */<br>**Output:** $r^S$ /* A synthetic patient record */ | |
| Step 1. | Initialize $G_\theta, D_\phi$ with random weights $\theta, \phi$ |
| Step 2. | Pretrain $G_\theta$ using MLE over $S$, $\theta \rightarrow \beta$ |
| Step 3. | Generate negative samples using $G_\theta$ for training $D_\phi$ |
| Step 4. | **do** |
| Step 5. | **for** _ in range(*g-steps*): |
| Step 6. | Generate sequence $Y_{1:T}$ using generator $G_\theta$ |
| Step 7. | Get accumulative reward by calculating $Q_{D_\phi}^{G_\theta}$ |
| Step 8. | Update generator parameters $\theta$ using policy gradient |
| Step 9. | **for** _ in range(*d-steps*): |
| Step 10. | Generate sequence $Y_{1:T}$ using generator $G_\theta$ |

| | |
|---|---|
| Step 11. | Train discriminator $D_\phi$ using $S$ and $Y_{1:T}$ |
| Step 12. | **Until** converges |

## 2.3 Experimental Result

The use of medical data for this study was approved by the Institutional Review Board at our hospital. 101 endotracheal intubation (breathing tube insertion) records were coded from surveillance videos. We evaluated the synthetic data quality based on statistics and medical expect feedback. Our results show that the synthetic data generated by trace alignment is highly similar to the real data. In addition, to assess the realism of the synthetic data, we created a mixed log with 16 authentic and 19 synthetic traces. A medical expert with experience coding our datasets was asked to identify which cases are authentic (i.e., possible to observe in practice). Our experimental results show that the medical expert correctly labelled 19 out of 35 cases. The accuracy was only 54.3 %, similar to random guessing, implying that the synthetic data was realistic and may be observed in practice. However, the sequences generated by GAN differ from the real data, which is caused by lack of the restriction on the length of the generated sequences. And also, the discriminator focuses on the single sequence rather than the whole set of synthetic data, i.e., the distribution of the length and activities is not considered in the training step.

**Table 3 Statistics of two real-world medical datasets and two synthetic datasets**

| **Dataset**<br>**Stats** | **Real Data** | **Trace Alignment** | **Sequence GAN** |
|---|---|---|---|
| **Num. Patient Records** | 101 | 2000 | 1000 |
| **Num. Total Acts** | 1239 | 24,673 | 10,520 |
| **Avg. Num. Acts in Trace** | 12.27 | 12.34 | 10.52 |
| **Var. of length of traces** | 2.54 | 2.42 | 1.67 |

# CHAPTER 3

# INTENTION MINING TECHNOLOGY

## 3.1 Background of Intention Mining

Intentions are the thoughts directed towards achieving process goals. We can infer people's intentions from their activities. For example, during trauma resuscitations (Figure 3), "maintain oxygenation" is a goal. Activities for addressing this goal may include "placing oxygen" or "placing an oxygen saturation monitor". The team may have the goal of "maintaining oxygenation", but not have the intention of fulfilling this goal for several minutes. When they do intend to satisfy this goal, their intention may be identified by observing the two oxygen-related activities. Previous workflow analyses in process mining have focused on simple mining of the patterns of observed activities. They did not attempt to understand the hidden (or unobservable) intentions underlying the observed processes. In addition, most previous research has focused on simple processes that include a limited number of activities. Performing these analyses in medical settings is more challenging because of the complex concurrency of associated activities. Knowing the intentions behind the activities can help simplify modeling of complex processes and provide accurate recommendations.

Intention mining has not been deployed in clinical settings. No previous research has tried to identify the intentions of a medical team during a clinical process. Hence, there is no established intention model of the trauma resuscitation process that could be used to supervise the intention mining. Although it is possible to manually perform intention mining, this approach is vulnerable to subjective bias. Experienced observers may bring useful domain knowledge to this analysis, but may also be prone to identifying only familiar or expected patterns while neglecting others. Careful review of activity lists and patient features could mitigate this bias, but require time and labor commitment. For this reason, a data-driven approach to intention discovery is more attractive.

**Figure 3 Emergency department (ED) trauma bay.**

### 3.2 Intention Mining using Hierarchical Hidden Markov Model

Intention model discovery includes: (1) inferring a hierarchical hidden Markov model (H-HMM) using historic process traces; (2) integrating context attributes into the induced H-HMM; (3) labeling the discovered intentions. We first perform unsupervised intention discovery by finding patterns in the observed activities. Our intention discovery is based on the fact that observed activities are correlated with hidden intentions. We also assume that activities caused by the same intention will be associated with each other. We used an HMM inference algorithm to extract these associations automatically. In addition, because no general criteria are available to assess the quality of trauma resuscitations across patients with various injuries and conditions, our process recommendation is based on the assumption that the “average” process enactment is more effective and valid than those deviating from the average.

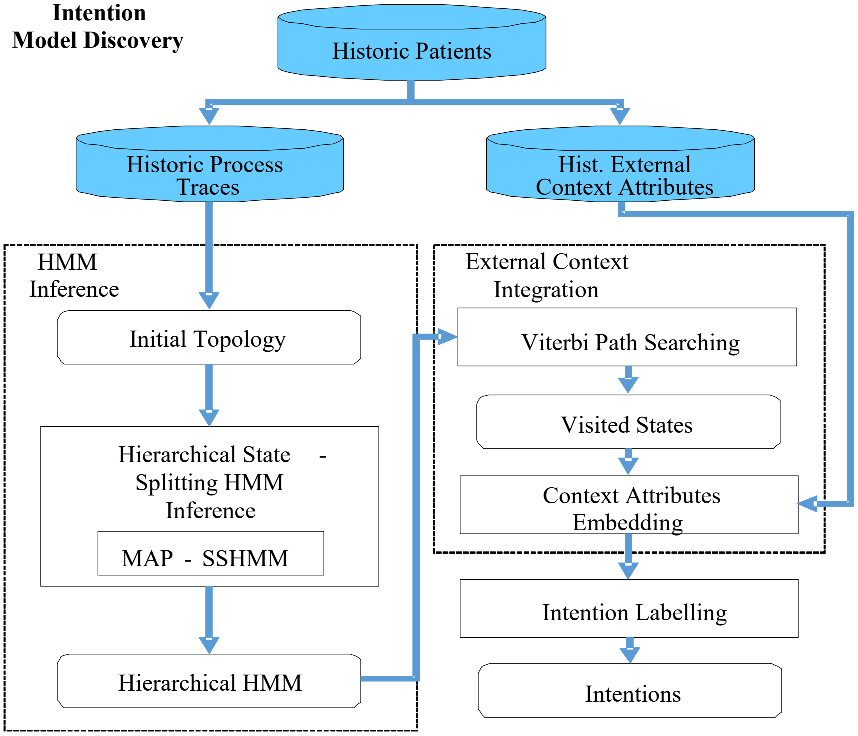


**Figure 4 Flow Chart of HMM Intention Mining Model**

We used hierarchical hidden Markov models (H-HMM) that model (Figure 4) intentions at multiple levels and high-level intentions are composed of low-level intentions. To derive this model, we introduced a state-splitting approach that avoids subjective and labor-intense parameter initialization (e.g., number of hidden states, transition probabilities). We integrated context attributes into the intention model, making it context-aware. We applied the approach on the trauma resuscitation process and the results show that we correctly labelled 86.59% of the intentions (with $F_1$ score as 0.87).

### 3.2.1 Inference of Hierarchical Hidden Markov Model

#### 3.2.1.1 State-splitting Hierarchical HMM

The trauma resuscitation log $\boldsymbol{L} = [c^{(1)}, \dots, c^{(l)}]^T$ is a vector of elements $c^{(i)}$. Each $c^{(i)} = \{id^{(i)}, \boldsymbol{x}^{(i)}, \boldsymbol{O}^{(i)}\}$ represents a trauma resuscitation case, which is indexed with a unique case id, contains the resuscitation trace $\boldsymbol{O}^{(i)}$, and has a vector $\boldsymbol{x}^{(i)}$ of context attributes. A resuscitation trace is $\boldsymbol{O}^{(i)} = [a_1^{(i)}, \dots, a_k^{(i)}]^T$, where k total activities a are ordered by activity start time. Traces of different resuscitation executions may have varying lengths, especially for complex processes with optional, omitted, or even erroneously performed activities. Context attributes $\boldsymbol{x}^{(i)} = [x_1^{(i)}, \dots, x_g^{(i)}]^T$ is a vector of $g$ recorded patient attributes (e.g., patient age, injury type) and hospital factors (e.g., day vs. night shift, prehospital triage of injury severity).

An intention $I_i^{(\ell,p)}$ is defined as a hidden (or unobservable) goal, objective, or motivation that can be achieved by a group of activities. Intentions can exist on multiple levels. Low-level intentions are called subintentions or subsubintentions. We use ℓ to denote the ℓ-th level of intention, i to denote the intention's index at the ℓ-th level, and p to denote the index of a parent intention at the (ℓ–1)-th level. At the top level (ℓ = 1), p = null[33][34].

A Hidden Markov Model $\lambda = (\boldsymbol{A}, \boldsymbol{B}, \boldsymbol{\pi}, \boldsymbol{Q}, \boldsymbol{\Sigma})$ can model temporal sequences using hidden or unobserved states. Here, $\boldsymbol{A}$ is the state transition probability matrix, so $t_{ij} \in \boldsymbol{A}$ represents the transition probability between states i and j, $\boldsymbol{B}$ represents each state's observation probability distribution, $\boldsymbol{\pi}$ is a vector of the initialized state distribution, $\boldsymbol{Q} = [q_1, \dots, q_{|\boldsymbol{Q}|}]^T$ is the vector of hidden states, and $\boldsymbol{\Sigma} = [e_1, \dots, e_{|\boldsymbol{\Sigma}|}]^T$ is the emission alphabet (i.e., observations). During trauma resuscitations, we can directly observe the team activities, but not their intentions. The unobservable intentions need to be inferred from the observations. We used hidden states of an HMM to model medical team intentions. To model the hierarchical intention structure, we adopted a hierarchical HMM. High-level intentions (or states) are composed of low-level intentions, and the lowest level of intentions are carried out by observable activities (Figure 5).

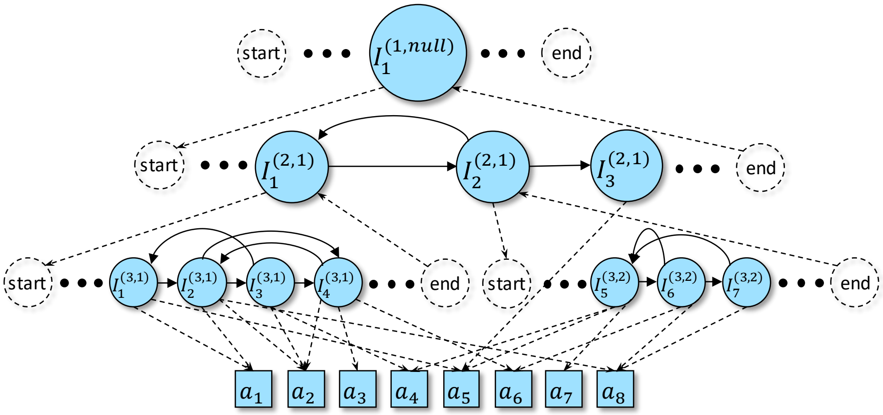


**Figure 5 Hierarchical Intention Model with 3 Layers**

To model intentions in a scenario as complex as the trauma resuscitation, we used an HMM inference model with several enhancements. First, to reduce the influence of the initial topology on the results, we used a state-splitting approach [12][13][14][15] instead of existing methods such as Baum-Welch. Second, we used in novel ways several characteristics of maximum a posteriori probability scoring [16] to guide the state-splitting. Third, to model intentions at different levels of granularity, we introduced a novel recursive algorithm for hierarchical HMM discovery.

State splitting works by initializing a general HMM (e.g., a single state) and successively splitting states until convergence is achieved (Figure 6). All candidate states for splitting are split and the best split found is used in the next iteration *i* of the model:

$$\lambda_i = \underset{\lambda_{i-1}^{q}}{\operatorname{argmax}} \operatorname{Score}(\lambda_{i-1}^{q}, \boldsymbol{O}) \quad s.t.\;\; q \in \boldsymbol{Q} \tag{4}$$

where $\lambda_{i-1}^{q}$ is the candidate model for splitting state q on the model $\lambda_{i-1}$. The scoring function Score(λ, O) quantifies how well the model λ fits the observations O, balanced against the model complexity penalty [12][13][14][15]. The algorithm terminates when further splitting does not increase the score:

$$\operatorname{Score}(\lambda_i, \boldsymbol{O}) \leq \operatorname{Score}(\lambda_{i-1}, \boldsymbol{O}) \tag{5}$$

Too many split states may lead to overfitting, trading representativeness for accuracy. Too few split states may lead to underfitting and less accurate models. We used the maximum a posteriori probability [16] scoring function to guide splitting. This method has several advantages over existing complexity metrics such as BIC [13], MDL [14], and Heuristic [14]. In our experiments, other metrics suffered from either over-penalizing (disallowing the model to split at all) or imbalanced splitting (where only a few states retain all information). MAP does not have these problems for several reasons. First, MAP penalizes complexity from three aspects: emission probabilities, transition probabilities, and number of states. The other metrics do not consider all three penalties. Second, the other metrics penalize the model linearly, whereas MAP compounds the measure of complexity (Eq. (3)). A set of observation process traces $\boldsymbol{O} = [\boldsymbol{O}_1, \ldots, \boldsymbol{O}_n]$, the MAP helps to find an HMM topology topology $\lambda$ that maximizes the posterior probability $P(\lambda|\boldsymbol{O})$:

$$\hat{\lambda}_{MAP}(\boldsymbol{O}) = \underset{\lambda}{\operatorname{argmax}} P(\lambda|\boldsymbol{O}) = \frac{P(\lambda)^{\omega_1} P(\boldsymbol{O}|\lambda)}{P(\boldsymbol{O})} \qquad \textbf{(6)}$$

where $P(\boldsymbol{O}|\lambda)$ is observation sequence probability, solved with the Forward algorithm [17]. The model prior $P(\lambda)$ can be considered as model complexity penalty. The weight (hyperparameter) $\omega_1$ is used to control model complexity. Observation $P(\boldsymbol{O})$ is fixed for a given data, so it can be ignored in the maximization.

The model prior P(λ) is composed of structural priors $P(\lambda_s)$ (i.e., prior probability distribution over all possible model topologies with a given number of states) and parameter priors $P(\theta_\lambda)$ (i.e., prior probability distribution of transitions and emissions).

$$P(\lambda) = P(\lambda_s) P(\theta_\lambda|\lambda_s) \qquad \textbf{(7)}$$

Let $P(\lambda_G)$ denote the prior for global aspects of the model structure (e.g., the number of states), where $\lambda_G$ is assumed to be unbiased and fixed. We have

$$P(\lambda) = P(\lambda_G) \prod_{q \in \boldsymbol{Q}} P\left(\lambda_s^{(q)} \middle| \lambda_G\right) P\left(\theta_\lambda^{(q)} | \lambda_G, \lambda_s^{(q)}\right) \qquad \textbf{(8)}$$

where $P(\lambda_s^{(q)})$ is a prior for the structure of state q and $P(\theta_\lambda^{(q)}|\lambda_G)$ is a prior for the parameters of state q.

$$P\left(\lambda_s^{(q)} \middle| \lambda_G\right) = p_t^{n_t^{(q)}} (1 - p_t)^{|\boldsymbol{Q}| - n_t^{(q)}} \left[ p_e^{n_e^{(q)}} (1 - p_e)^{|\boldsymbol{\Sigma}| - n_e^{(q)}} \right]^{\omega_2} \qquad \textbf{(9)}$$

where $n_t^{(q)}$ is the estimated number of outgoing transitions of state q and $n_e^{(q)}$ is the number of its emissions. $p_t = \overline{n_t}/|\boldsymbol{Q}|$ is the probability of a potential transition's existence, and $p_e = \overline{n_e}/|\boldsymbol{Q}|$ is the probability of a possible emission. $\overline{n_t}$, $\overline{n_e}$ are the expected (i.e., average) number of transitions and emissions per state. The weight ω2 serves to balance the penalty of emissions over transitions.

$$P\left(\theta_{\lambda}^{(q)} \middle| \lambda_G, \lambda_s^{(q)}\right) = \frac{1}{B(\alpha_t, \dots, \alpha_t)} \prod_{i=1}^{n_t^{(q)}} \theta_{qi}^{(\alpha_t - 1)} \left[ \frac{1}{B(\alpha_e, \dots, \alpha_e)} \prod_{j=1}^{n_e^{(q)}} \theta_{qj}^{(\alpha_e - 1)} \right]^{\omega_2} \quad \textbf{(10)}$$

where the Dirichlet distribution is used as the prior distribution over the model parameters (transition probabilities $\boldsymbol{\theta}_t^{(q)} = [\theta_{q1}, \dots, \theta_{qn}]^T$ and emission probabilities $\boldsymbol{\theta}_e^{(q)} = [\theta_{q1}, \dots, \theta_{qn}]^T$) in a given HMM structure $\lambda_s$. We set the prior weights α to 2 in the Beta function $B(\alpha, \dots, \alpha)$. The normalizing constant Beta function helps produce more balanced states (i.e., states of similar size) and avoid imbalanced states (i.e., states that greatly differ in size) in the intention model. Intuitively, we favor similar-size intentions in the same model. The hierarchical HMM  is inferred recursively (Table 4). We first find a subset of observations $\boldsymbol{O}_i^s \subseteq \boldsymbol{O}^s$ that can be emitted in hidden state *qi* in the intention model $\lambda^l$. This is done using the Viterbi algorithm [17], which finds the optimal state sequence associated with the observed activity sequence. A lower level intention model $\hat{\lambda}_i^{\ell+1}$ is then recursively inferred based on $\boldsymbol{O}_i^s$ (Table 4). The initial topology of this model is set to three nodes (one for start, one for end, and one split-able state in between). The recursion terminates at levels where the complexity does not allow any more splitting. Afterwards as a post-processing step, the inferred model $\boldsymbol{\lambda}^H$ is smoothed to flatten the emission probability distribution so that all traces can occur with some probability.

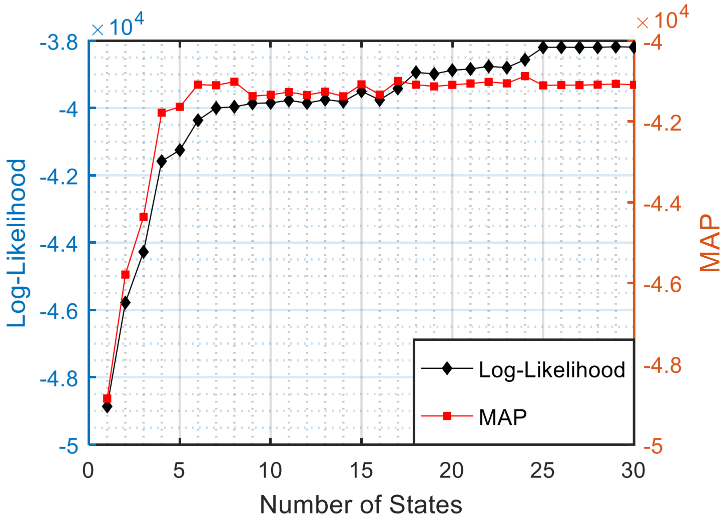


**Figure 6 Log-likelihood and MAP change as state splitting**

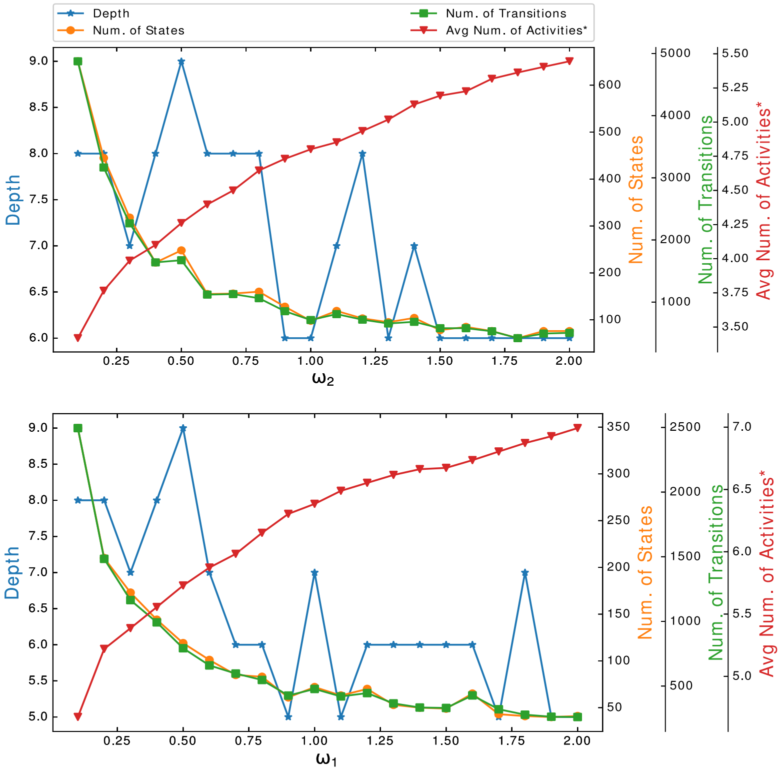


**Figure 7 (a) Influence of ω2 with a constant ω1 = 1. (b) Influence of ω1 with a constant ω2 = 1. (*The avg. num. of activities per lowest-level intentions before smoothing).**

$\omega_1$, $\omega_2$ are two weights (or hyperparameters) that control model complexity and the inferred model's topology. $\omega_1$, $\omega_2$ are 1 by default. A smaller $\omega_1$ leads to larger, deeper models. A smaller $\omega_2$ is equivalent to smaller emission penalties, allowing for more activities to be observed in the split states (Figure 7). Intuitively, we favor a hierarchical model that is smaller, simpler, and easily-interpretable for labeling purposes. As $\omega_1$ increases (with $\omega_2$ held constant at 1), the depth of the intention model drops from eight to five hierarchical levels of the model. The total number of states and transitions decreases. The average number of activities per lowest-level state increases from 4.5 to 7 per state.

**Table 4 Algorithm of State-Splitting Hierarchical HMM Inference**

```
Algorithm: State-Splitting Hierarchical HMM Inference
Input: $\boldsymbol{O}$, $\omega_1$, $\omega_2$
Output: $\boldsymbol{\lambda}^H$
/* Initialization */
Step 1.  Initialize HMM topology $\lambda_0$ as three nodes (a single node of observations and two functional nodes, start and end);
/* When intention level $\ell$ = 1 (top level) */
Step 2.  Infer top level of intention model: $\hat{\lambda}^1 = \text{MapSS}(\boldsymbol{O}, \omega_1, \omega_2)$;
/* When intention level $\ell$ >1, do recursive inference */
Step 3.  $\boldsymbol{\lambda}^H$ = RecursiveInference($\hat{\lambda}^1$, $\boldsymbol{O}$);
Step 4.  Smooth the model: $\boldsymbol{\lambda}^H$ = Smoothing($\boldsymbol{\lambda}^H$);
Step 5.  return $\boldsymbol{\lambda}^H$;

Function: RecursiveInference (inferred $\lambda^\ell$, observation $\boldsymbol{O}^s$)
Step 1.  Find subsequence $\boldsymbol{O}_i^s \subseteq \boldsymbol{O}^s$ that can be observed in hidden state $q_i$, $\{\boldsymbol{O}_i^s\} = \text{Viterbi}(\boldsymbol{O}^s, \lambda^\ell)$;
Step 2.  for each state $q_i$ in $\lambda^\ell$, do
Step 3.      Infer subintention model: $\hat{\lambda}_i^{\ell+1} = \text{MapSS}(\boldsymbol{O}_i^s, \omega_1, \omega_2)$;
Step 4.      If number of states in subintention model $|\boldsymbol{Q}_i^{\ell+1}|$>3
Step 5.          RecursiveInference($\hat{\lambda}_i^{\ell+1}$, $\boldsymbol{O}_i^s$);
Step 6.      end if
Step 7.  end for
Step 8.  $\boldsymbol{\lambda}^H = \boldsymbol{\lambda}^H \cup \{\lambda^\ell\}$ ;
end Function
```

```
Function: Smoothing (inferred model $\boldsymbol{\lambda}^H$)
for each $\lambda$ in $\boldsymbol{\lambda}^H$, do
  for each state $q_i$ in $\lambda$, do
    Let $|\boldsymbol{\Sigma}|$ denote the number of emissions in $q_i$;
    for each emission $e$ in $q_i$, do
      if $\boldsymbol{B}(e) \neq 0$, do
        $\boldsymbol{B}(e) = \boldsymbol{B}(e) \cdot (1–0.003) + 0.003/|\boldsymbol{\Sigma}|$   // 0.003 is selected based on
        three-sigma rule of thumb;
      else
        $\boldsymbol{B}(e)$ = 0. 003/$|\boldsymbol{\Sigma}|$;
      end if
    end for
  end for
return $\boldsymbol{\lambda}^h$;
end Function
```

### 3.2.1.2 Incorporation of Context Attributes

The medical team must act according to the different conditions of patients arriving at the emergency department (e.g., injury area and severity). For this reason, it is important to incorporate context attributes into the recommender system for prescriptive analytics [21][22]. The recommendations (or predictions) need to be based on the intentions or objectives the medical team forms to treat patients of different conditions.

We incorporated context attributes into the intention model by replaying the historic data on the intention model $\boldsymbol{\lambda}^H$ and recording the distribution of context attributes $\boldsymbol{x}$ at each transition and emission (Table 5). Specifically, we used the Viterbi algorithm [17] to find the optimal sequence of hidden states, $\boldsymbol{Q}^{(i)} = \{{q_1}^{(i)}, \dots, {q_t}^{(i)}, \dots, {q_T}^{(i)}\}$, given a specific observation sequence $\boldsymbol{O}^{(i)} = \{{a_1}^{(i)}, \dots, {a_t}^{(i)}, \dots, {a_T}^{(i)}\}$ and HMM $\lambda$. For each state ${q_t}^{(i)}$ and transition $({q_t}^{(i)} \rightarrow {q_{t+1}}^{(i)})$, we assigned context attribute $\boldsymbol{x}^{(i)}$. The output from Alg. 2 is a hierarchical intention model $\boldsymbol{\lambda}^{(H,c)}$ with labeled context attributes $c$ on each transition and emission.

**Table 5 Methods of Incorporation Context Attributes**

| **Algorithm: Incorporating Context Attributes** | |
|---|---|
| **Input:** $\boldsymbol{O}$, $\boldsymbol{\lambda}^{H}$, map<$\boldsymbol{O}^{(i)}$, $\boldsymbol{x}^{(i)}$> | |
| **Output:** $\boldsymbol{\lambda}^{(H,c)}$ | |
| Step 1. | **for** each trace $\boldsymbol{O}^{(i)} \in \boldsymbol{O}$, **do** |
| Step 2. | Find the most likely path $p^{(i)} = \{\boldsymbol{Q}^{(i)},\ \boldsymbol{t}^{(i)}\}$ in $\boldsymbol{\lambda}^{H}$ given an observed trace $\boldsymbol{O}^{(i)}$. $p^{(i)} = \underset{p=\{\boldsymbol{Q}^{(i)},\, \boldsymbol{t}^{(i)}\}}{\arg\max}\ P\left(p \mid \boldsymbol{O}^{(i)}, \boldsymbol{\lambda}^{H}\right)$.<br>Solve $p^{(i)}$ using Viterbi algorithm; |
| Step 3. | **for** each state $q_j \in \boldsymbol{Q}^{(i)}$ and transition $t_{kj} \in \boldsymbol{t}^{(i)}$ **do** |
| Step 4. | $q_j.\boldsymbol{x} = q_j.\boldsymbol{x} + \boldsymbol{x}^{(i)}$; |
| Step 5. | $t_{kj}.\boldsymbol{x} = t_{kj}.\boldsymbol{x} + \boldsymbol{x}^{(i)}$; |
| Step 6. | **end for** |
| Step 7. | Normalize $q_j.\boldsymbol{x}$ and $t_{kj}.\boldsymbol{x}$ into likelihood: $P_{q_j}(\boldsymbol{x})$ and $P_{t_{kj}}(\boldsymbol{x})$; |
| Step 8. | Smooth $P_{q_j}(\boldsymbol{x})$ and $P_{t_{kj}}(\boldsymbol{x})$; |
| Step 9. | **end for** |
| Step10. | $\boldsymbol{\lambda}^{(H,c)} = \boldsymbol{\lambda}^{H}$ ; |
| Step11. | **return** $\boldsymbol{\lambda}^{(H,c)}$; |

### 3.2.2 Experimental Results

To evaluate the discovered intention model, we checked whether our data-driven approach can discover meaningful intentions. We discovered the intention model in an unsupervised way, under the assumption that sequential relationships between the observed activities are correlated with medical team intentions. To validate our assumption and evaluate our intention model, we conducted two different experiments. (1) We asked medical experts whether they could manually assign the discovered intentions (Figure 8) with meaningful labels. If they could not come up with a label, they would note “intention cannot be defined”. (2) We provided medical experts 40 cases of a total 1074 activities and asked them to manually label the activities with top-level (level-1) intentions (Figure 9). We then compare the human labels to algorithm-derived labels.

#### 3.2.2.1 Qualitative Evaluation

From engineering perspective, the emission distribution of low-level intentions is sparse (matrix in Figure 8). This is because of the nature of our trauma resuscitation data. Many resuscitation activities have a strict or strong association with certain intentions. For example, activity “L-spine-BK” (Figure 9) is only associated with intention $\mathrm{I}_{13}^{(2,5)}$ “Assessment of Lumbar Spine”. For other activities, a strict or strong association with a specific intention is not observed. For example, “Palpation-H” is not only correlated with level-1 intention “Assessment of Head and Face” but also the level-1 intention “Assessment of Back and Posterior Aspect of the Head and Extremities”. This is because the back of the head is exposed once the medical team rolls the patient onto his or her side in order to assess the back. Assessing the back of the head here allows the examining provider to complete the head exam without placing stress on the neck or cervical-spine. The medical experts commented: the intention model correctly captures the intentions associated with the medical tasks. Associated activities were grouped primarily according to the expectations of medical experts. The model is advantageous because it accommodates for differences between providers and resuscitation attributes. Additionally, the model correctly groups related intentions, for example head and neck or chest and abdomen, which reflects the same groupings that medical experts would predict.

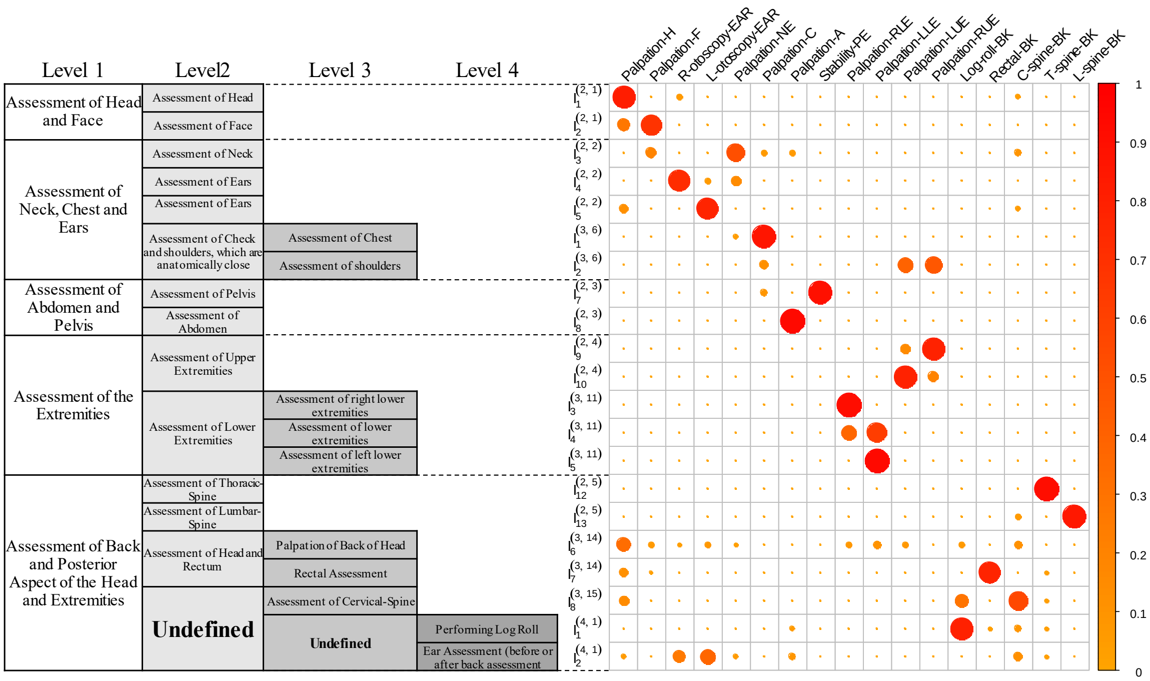


**Figure 8. Multi-level intentions (left) labeled by medical experts. The column of the matrix represents activity type and the row represents the lowest level of intentions. The size and color of the dots in the emission matrix represents the probability.**

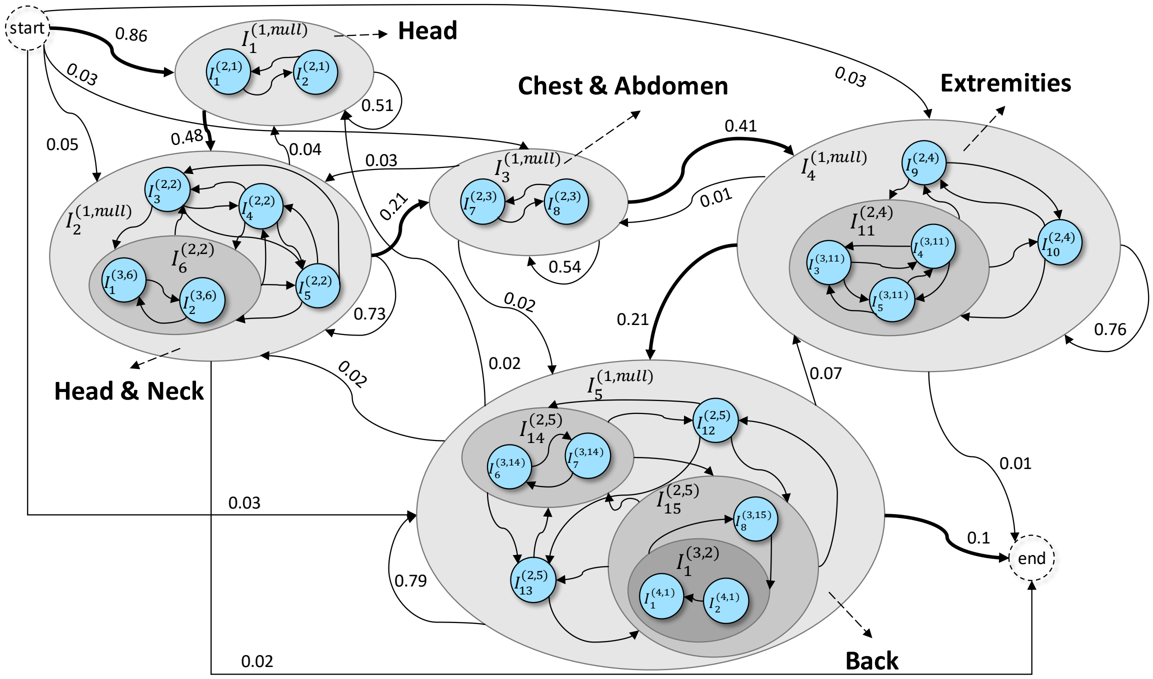
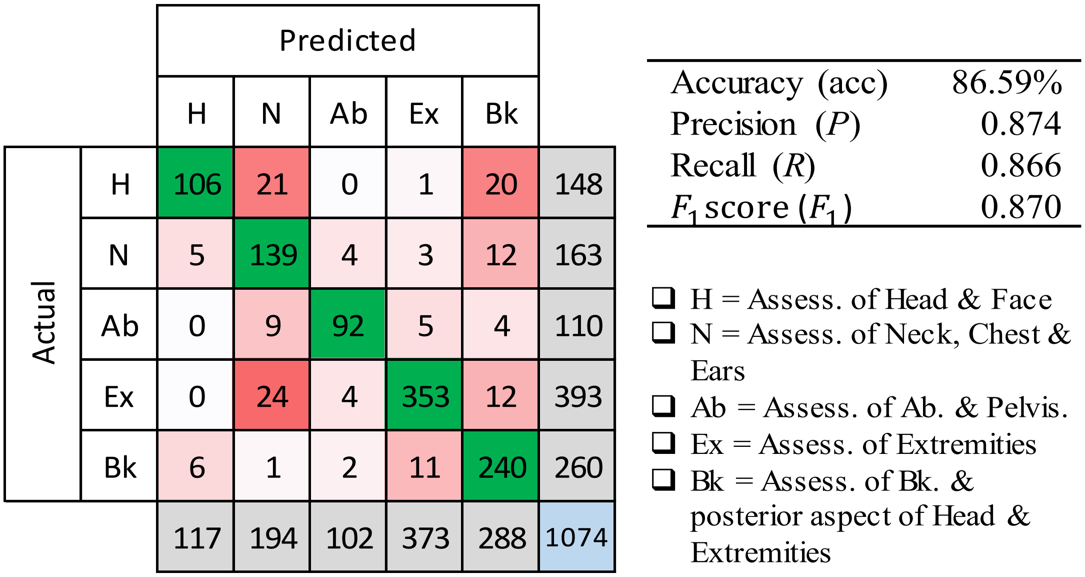


**Figure 9. Discovered intention model. The model is trained based on a sample data with 17 secondary survey activities in 123 resuscitations.**

One limitation however is that the model only provides a limited description of provider intentions. It is unable to distinguish between deviations and innovations. For example, medical experts were unable to assign some level 2 and level 3 intentions under the level 1 intention of assessment of back and posterior aspect of the head and extremities. Most of the activities that occur under this intention take place while the patient is rolled to her side, exposing the back. Given this context, medical experts were unable to determine why inner ear examination (otoscopy) was included as it is typically performed while the patient is lying on her back. That an explanation was unavailable should not be considered evidence of no association. However, future work should attempt to uncover reasons for these data driven results.

#### 3.2.2.2 Quantitative Evaluation

The intention labeling (algorithm vs. hand) results (Figure 10) show high intention mining accuracy, with 86.59% $acc$ and 0.87 $F_1$ score, indicating the feasibility of data-driven algorithms for intention mining. The misclassification mainly occurs at the “N” and “Bk” intentions. There most likely cause for this is the proximity of one body region to another. The neck, chest, shoulders (upper extremities), and abdomen are located very close to each other and providers will frequently move between these regions during an assessment. This is less true of these regions and the head because examiners typically complete the head exam before moving onto the thorax and abdomen. It makes sense, however, that there is some confusion associated with the head and back exams as the back of the head is commonly assessed during the back exam.



| | | Predicted | | | | | |
|---|---|---|---|---|---|---|---|
| | | H | N | Ab | Ex | Bk | |
| Actual | H | 106 | 21 | 0 | 1 | 20 | 148 |
| | N | 5 | 139 | 4 | 3 | 12 | 163 |
| | Ab | 0 | 9 | 92 | 5 | 4 | 110 |
| | Ex | 0 | 24 | 4 | 353 | 12 | 393 |
| | Bk | 6 | 1 | 2 | 11 | 240 | 260 |
| | | 117 | 194 | 102 | 373 | 288 | 1074 |

| | |
|---|---|
| Accuracy (acc) | 86.59% |
| Precision ($P$) | 0.874 |
| Recall ($R$) | 0.866 |
| $F_1$ score ($F_1$) | 0.870 |

- □ H = Assess. of Head & Face
- □ N = Assess. of Neck, Chest & Ears
- □ Ab = Assess. of Ab. & Pelvis.
- □ Ex = Assess. of Extremities
- □ Bk = Assess. of Bk. & posterior aspect of Head & Extremities

**Figure 10 Confusion matrix for algorithm-derived (predicted) intentions vs. hand (actual) intentions.**

### 3.3 Intention Mining Using Seq2seq Model

#### 3.3.1 Background of Seq2seq Model

Critically injured patients have up to a four-fold higher risk of death from errors than general hospital patients, and nearly half of these preventable deaths are related to human errors during the initial resuscitation phase of treatment. We are developing a computerized decision support system in the trauma bay that gives real-time alerts or recommendations to medical teams based on their intentions (defined as hidden goals or objectives). To capture such intentions, we developed an activity-to-intention model using the sequence-to-sequence model from deep learning. Our model includes two recurrent neural networks (RNNs): an encoder that processes the observed medical treatment procedures and a decoder that generates the intention sequences. We tested our model on 35 trauma resuscitation cases from Children's National Medical Center, a level 1 trauma center. Our preliminary analyses showed the feasibility of automated discovery of medical team's intentions in the trauma resuscitation process.

### 3.3.2 Seq2seq Model Structures

#### 3.3.2.1 Recurrent Neural Networks (RNN)

RNNs are powerful at modeling temporal sequences. The standard RNN, however, still suffers vanishing or exploding gradients when learning long-term dependencies. So we used two RNN variations, Long Short Term Memory (LSTM, [20]) networks and Gated Recurrent Unit (GRU, [21]). A LSTM unit at time step *t* is composed of an input gate $i_t$ , a forget gate $f_t$, an output gate $o_t$, a memory cell $c_t$ and a hidden state $h_{t-1}$. The LSTM transition equations are:

$$\begin{aligned}
i_t &= \sigma\left(W^{(i)}x_t + U^{(i)}h_{t-1} + b^{(i)}\right),\\
f_t &= \sigma\left(W^{(f)}x_t + U^{(f)}h_{t-1} + b^{(f)}\right),\\
o_t &= \sigma\left(W^{(o)}x_t + U^{(o)}h_{t-1} + b^{(o)}\right),\\
u_t &= tanh\left(W^{(u)}x_t + U^{(u)}h_{t-1} + b^{(u)}\right),\\
c_t &= i_t \odot u_t + f_t \odot c_{t-1},\\
h_t &= o_t \odot \tanh(c_t)
\end{aligned} \tag{11}$$

A GRU unit at time step *t* consists of an update gate $z_t$, a reset gate $r_t$ and a hidden state $h_{t-1}$. The GRU transition equations are:

$$z_t = \sigma\left(W^{(z)}x_t + U^{(z)}h_{t-1} + b^{(z)}\right), \tag{12}$$

$$r_t = \sigma\left(W^{(r)}x_t + \ U^{(r)}h_{t-1} + b^{(r)}\right),$$

$$u_t = tanh\left(W^{(u)}x_t + \ \ U^{(u)}(h_{t-1} \odot \ r_t) + b^{(u)}\right),$$

$$h_t = z_t \odot h_{t-1} + (1 - \ z_t) \odot u_t$$

where $x_t$ is the input at the current time step, $\sigma$ denotes the sigmoid function and $\odot$ denotes element-wise multiplicationGRUs have simpler internal structure (3 gates vs. 2 gates) and are easier to train with less training data. Existing research shows GRUs outperforming LSTMs in most tasks [21]. We tried both in our framework.

**Loss Function**: We used categorical cross-entropy as the loss function (Eq. 13).

$$\mathcal{L}(\hat{y}, y) = -\frac{1}{N}\sum_{i}^{N}\sum_{j}^{M} w_j(y_{ij}\log\hat{y}_{ij}) \qquad \textbf{(13)}$$

where *N* is the total number of predictions and *M* is the number of activity types. $\hat{y}_{ij}$ is the predicted probability for the activity type of index $j$. $y_{ij}$ is either 0 or 1 stating if the activity type of index *j* is the correct class. To handle the imbalanced distribution of activity types, we included the class weights $\boldsymbol{w} = [w_1, \dots, w_M]$ (Eq.14) so that the model can pay more attention to the underrepresented classes.

$$w_j = n_{samples}/(n_{classes} * n_{y_j}) \qquad \textbf{(14)}$$

### 3.3.2.2 Framework of Seq2seq Intention Mining Model

Seq2seq models are usually used in language translation tasks. Here we show a straightforward application of the LSTM architecture can solve general sequence to sequence problems. The idea is to use one LSTM to read the input sequence, one timestep at a time, to obtain large fixed dimensional vector representation, and then to use another

LSTM to extract the output sequence from that vector (Figure 11). The second LSTM is essentially a recurrent neural network language model [22][23][24] except that it is conditioned on the input sequence. The LSTM's ability to successfully learn on data with long range temporal dependencies makes it a natural choice for this application due to the considerable time lag between the inputs and their corresponding outputs (Figure 11)[25].

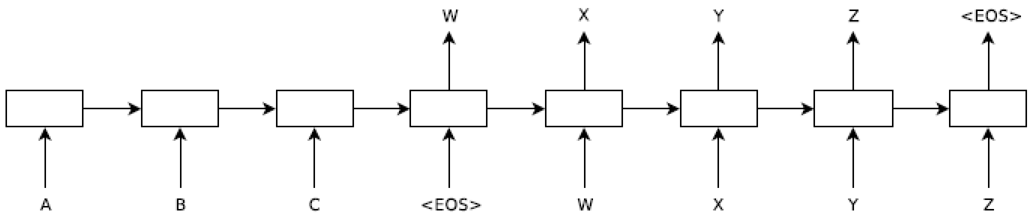


**Figure 11 Structure of Seq2seq Models**

Here we applied the Seq2seq model to the intention mining procedures. The inputs are traces of the activities, and the outputs are the intentions extracted in traces. In Figure 12, we show an example trace to illustrate the procedure. The activity trace 'palpation-h, r-otoscopy-ear … t-spine-bk' is feed into the encoder part of the seq2seq model. After the LSTM iterate over the input trace, we can get the latent representation *o* of the input trace. And then the decoder extracts the intentions from the latent representation which is 'asses of head, asses of face, …, asses of bk'.

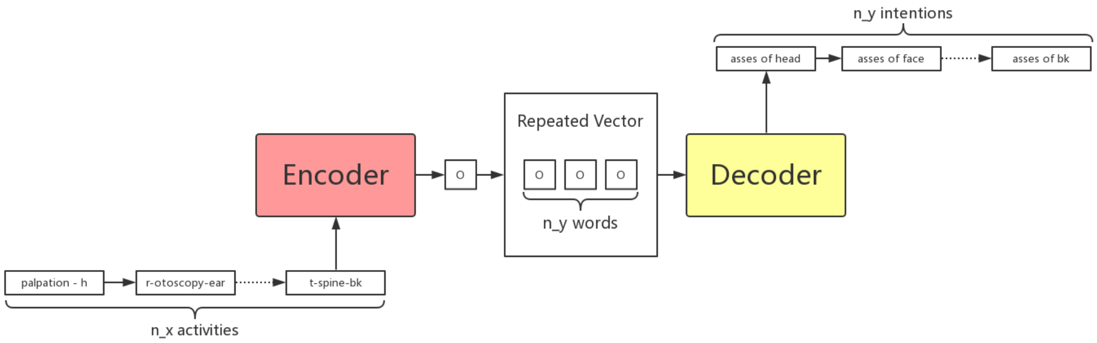


**Figure 12 Framework of Medical Intention Mining System**

We also applied the Act2Vec in the input activity traces, which is similar to the Word2Vec used in the translation models. For tasks like object or speech recognition we know that all the information required to successfully perform the task is encoded in the data. However, natural language processing systems traditionally treat words as discrete atomic symbols, these encodings are arbitrary, and provide no useful information to the system regarding the relationships that may exist between the individual symbols. Act2vec is a particularly computationally efficient predictive model for learning word embeddings from raw text. We constructed an activity dictionary and converted the sequences to related vectors.

Table 6 Shows the details of the seq2seq algorithm. In our approach, we tried to learn a Seq2seq model $\lambda$ from labelled process traces $O_b$ and use $\lambda$ extract intention from new process data $O_t$. Our method can be summarized in three steps.

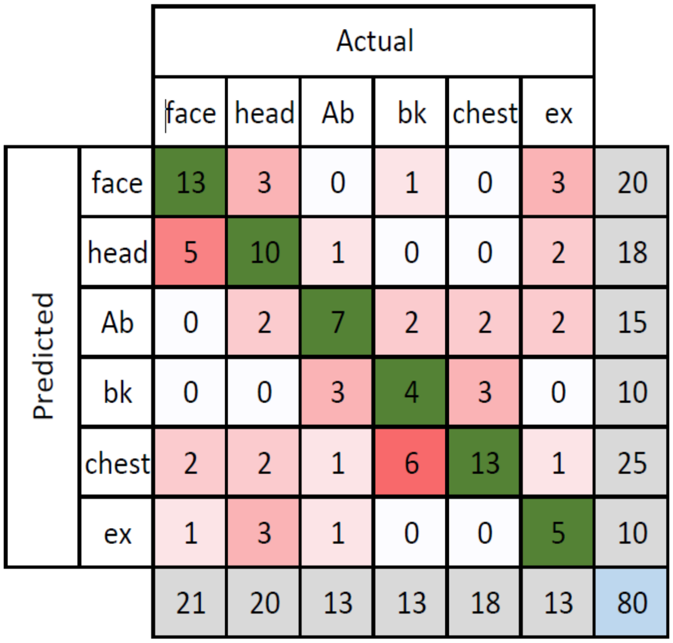
**Table 6 Algorithm of Seq2seq Intention Mining Model**

| **Algorithm: Intention Mining Using Seq2seq Model** | |
|---|---|
| **Input: $\boldsymbol{O^b}$, $\boldsymbol{O^t}$, $\boldsymbol{I_b}$** | /* Medical Process and labelled intentions */ |
| **Output: $\boldsymbol{I^t}$** | /* Extracted Intentions from new medical process */ |

| | |
|---|---|
| Step 1. | Convert the medical process data sequences $\boldsymbol{O}^{b}$ to vectors $\boldsymbol{V}^{x}$, and labelled intentions to vectors $\boldsymbol{V}^{y}$ |
| Step 2. | Infer the Seq2seq model $\boldsymbol{\lambda}$ on medical process vectors $\boldsymbol{V}^{x}$ and intentions vectors $\boldsymbol{V}^{y}$ |
| Step 3. | Extract intentions $\boldsymbol{I}^{t}$ from new medical process $\boldsymbol{O}^{t}$ |
| Step 3.1 | Convert the process $\boldsymbol{O}^{t}$ to vectors $\boldsymbol{V}^{x\prime}$ |
| Step 3.2 | Generate intentions vectors $\boldsymbol{V}^{y\prime}$by trained model $\boldsymbol{\lambda}$ |
| Step 3.3 | Recover vectors $\boldsymbol{V}^{y\prime}$to intention sequences $\boldsymbol{I}^{t}$ |
| Step 4. | **return** $\boldsymbol{I}^{t}$; |

### 3.3.3 Experimental Results

To evaluate the seq2seq model, here we tested our model on 35 trauma resuscitation cases from Children's National Medical Center, a level 1 trauma center. Training accuracy goes up to 80% in after 300 epochs of training, and the accuracy on the testing set is 65%. Figure 13 shows the confusion matrix of the testing result. For the values $v(i,j)$ in matrix, indicates the number of occurrence that the actual intention is $j$ and the predicted intention is $i$, e.g. the values on the diagonal line indicated the number of correct predictions.



| | | Actual | | | | | | |
|---|---|---|---|---|---|---|---|---|
| | | face | head | Ab | bk | chest | ex | |
| Predicted | face | 13 | 3 | 0 | 1 | 0 | 3 | 20 |
| | head | 5 | 10 | 1 | 0 | 0 | 2 | 18 |
| | Ab | 0 | 2 | 7 | 2 | 2 | 2 | 15 |
| | bk | 0 | 0 | 3 | 4 | 3 | 0 | 10 |
| | chest | 2 | 2 | 1 | 6 | 13 | 1 | 25 |
| | ex | 1 | 3 | 1 | 0 | 0 | 5 | 10 |
| | | 21 | 20 | 13 | 13 | 18 | 13 | 80 |

**Figure 13 Confusion Matrix of Seq2seq Intention Mining Results**

As mentioned above, overfitting is a common problem of deep learning models. When the hidden layer size increase, the training accuracy can goes higher (up to 97% for hidden layer size = 1000). However, the model cannot output reasonable intentions and the accuracy decreased. The model is overfitted.

# CHAPTER 4

# MEDICAL TREATMENT PROCESS RECOMMENDATION

## 4.1 Background of Medical Treatment Process Recommendation

Medical teams make unavoidable errors in the 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 [26]. During trauma resuscitations, 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. Despite the use of standardized protocols for establishing treatment and management goals, deviations from these protocols are observed in up to 85% of trauma resuscitations [27]. Although most deviations are variations resulting from the flexibility or adaptability needed for managing patients with different injuries, other deviations represent significant errors that may contribute to adverse patient outcomes [28]. To reduce the chances of errors, our research explores how to remind the clinical doctors during the medical processes by data-driven recommender system. The recommender system built on artificial intelligence (AI) and data mining techniques will provide the medical team leader (or surgical coordinator) with next-step treatment recommendations through the wall displays (monitors) (Figure 14).

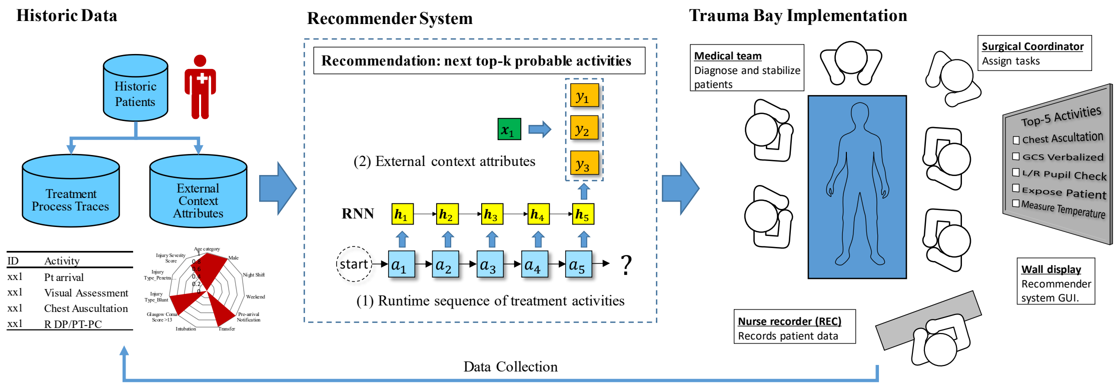


**Figure 14 Overview of Recommendation System in Trauma Bay**

## 4.2 Process Recommendation Based on Hierarchical Hidden Markov Model

We’ve already introduced the HHMM technology in Chapter 3.2. We first perform unsupervised intention discovery by finding patterns in the observed activities. Our intention discovery is based on the fact that observed activities are correlated with hidden intentions. We also assume that activities caused by the same intention will be associated with each other. We used an HMM inference algorithm to extract these associations automatically. In addition, because no general criteria are available to assess the quality of trauma resuscitations across patients with various injuries and conditions, our process recommendation is based on the assumption that the “average” process enactment is more effective and valid than those deviating from the average.

The process recommendation is done in two steps. First, based on the given observation sequence, we estimate the current intention, i.e., the state in the intention model. Second, knowing the estimated intention, we provide several different types of recommendations.

Intention estimation is done using our context-aware Viterbi algorithm. Extended from the classic Viterbi algorithm, our context-aware version considers the context attributes while finding the optimal state sequence associated with a given observation sequence. We modified the objective function of the Viterbi algorithm (Eq.31 in [16]) to include the context attribute probabilities:

$$\begin{aligned}&\delta_{t+1}(j)\\&=\left[\max_{k}\delta_t(j)\,P_{t_{kj}}(\boldsymbol{x}^{(i)})\right]\cdot e_j(a_{t+1}{}^{(i)})\\&\cdot P_{q_j}(\boldsymbol{x}^{(i)})\end{aligned} \tag{15}$$

where $\delta_t(j)$ is the score (i.e., the probability) of the optimal state sequence, at time $t$, associated with the first $t$ observations $\boldsymbol{O}^{(i)} = \{a_1{}^{(i)}, \dots, a_t{}^{(i)}\}$ and ending at state $q_t^{(i)}$. $P_{t_{kj}}(\boldsymbol{x}^{(i)})$ and $P_{q_j}(\boldsymbol{x}^{(k)})$ are calculated as:

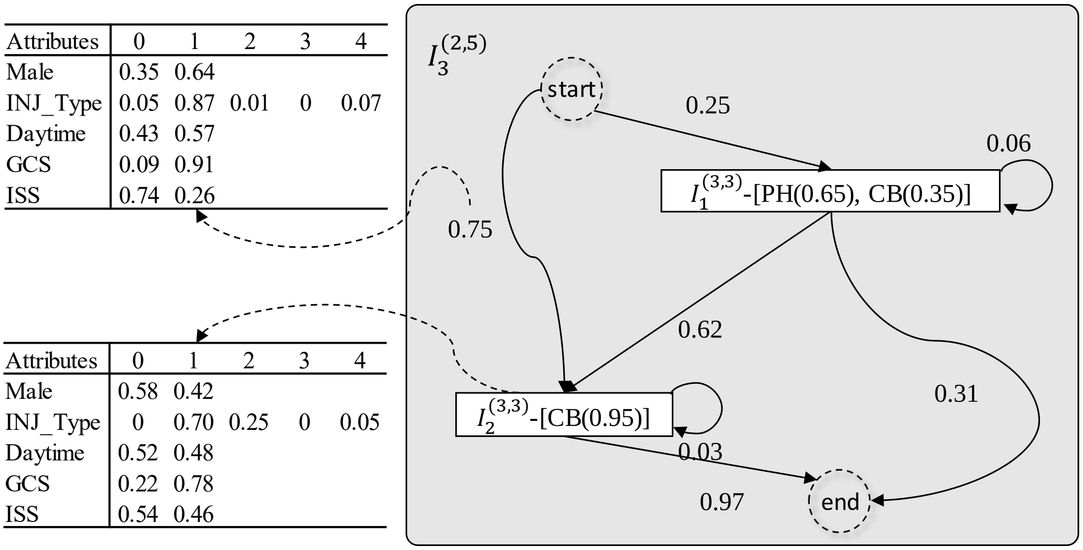
$$P_{t_{kj}}(\boldsymbol{x}^{(i)}) = \sqrt[|\boldsymbol{x}|]{\prod_{m} P_{t_{kj}}\left(x_m^{(i)}\right)} \tag{16}$$

and

$$P_{q_j}(\boldsymbol{x}^{(i)}) = \sqrt[|\boldsymbol{x}|]{\prod_{m} P_{q_j}\left(x_m^{(i)}\right)} \tag{17}$$

The estimated intention is:

$$q_t^{(i)} = \underset{0\le j\le|\boldsymbol{Q}|}{\operatorname{argmax}}\,\delta_t(j) \tag{18}$$



| Attributes | 0 | 1 | 2 | 3 | 4 |
|---|---|---|---|---|---|
| Male | 0.35 | 0.64 | | | |
| INJ_Type | 0.05 | 0.87 | 0.01 | 0 | 0.07 |
| Daytime | 0.43 | 0.57 | | | |
| GCS | 0.09 | 0.91 | | | |
| ISS | 0.74 | 0.26 | | | |

| Attributes | 0 | 1 | 2 | 3 | 4 |
|---|---|---|---|---|---|
| Male | 0.58 | 0.42 | | | |
| INJ_Type | 0 | 0.70 | 0.25 | 0 | 0.05 |
| Daytime | 0.52 | 0.48 | | | |
| GCS | 0.22 | 0.78 | | | |
| ISS | 0.54 | 0.46 | | | |

*Injury Type: 1-Blunt, 2-Burn, 3-Penetrating, 4-Animal Bite, 0-other
*ISS (Injury Severity Score): 0-<2, 1->=2
*GCS : 0-<13, 1->13

**Figure 15 Incorporating Context Attributes to Intention Model**

We use three different procedure recommendation strategies, two at the activity level (Obj.1 and Obj.2) and one at the intention level (Obj.3).

One-step activity recommendation (Obj.1) is the most common and simplest recommendation strategy. This approach has limitations of low accuracy and is short-sighted (can only see one activity ahead). It makes recommendations mainly based on the sequential relationships between adjacent activities. In complex real-world processes, a certain set of closely-knitted activities for an intention might not need to be performed in a strict sequence. Instead, they may occur in any order so long as they are close to each other. A strict sequential recommender would fail to model this flexibility. Hence, we presented the second approach (Obj.2), recommending a sequence of activities to follow the current observation, which considers fulfilling an intention with a group of activities. The recommendation is found on by a greedy search of the next activity that maximizes probability to be observed next from the current state in the model (Eq. 18). To avoid being trapped in loops (e.g., self-transitions), we updated the probabilities of recommended transitions and emissions based on multinomial distribution. Compared to the first (Obj.1), recommending several next-step activities (Obj.2) can facilitate planning and execution. In practice, trauma resuscitations are fast paced and dynamic processes, making anticipating future steps essential to care. Identifying multiple next-steps also provides model flexibility. Although a standardized protocol for trauma care has been developed (Advanced Trauma Life Support, or ATLS [3]) the order of activity performance may change depending on patient and resuscitation attributes. Recommending only one activity ahead, therefore, would not reflect how lower-level intentions can dynamically change.

It may also be useful to recommend the next intention (Obj.3) instead of specific activities. It is possible to satisfy a particular intention with more than one activity. Focusing only on activity recommendation may obfuscate the fulfillment of a given intention if a medical provider performed an activity other than those the system recommends. Intention recommendation could allow providers to perform a range of activities to fulfill an intention without relying on specific activity recommendations.

For Obj.1, we recommend a next activity $a_{t+1}$ to the medical team that maximizes:

$$a_{t+1} = \underset{e \in \boldsymbol{\Sigma}}{\operatorname{argmax}} P(e) = \sum_{k=1}^{|\boldsymbol{Q}|} P\left(e \middle| q_{t+1}^{k}\right) P(q_{t+1}^{k}) P_{q_{t+1}^{k}}(\boldsymbol{x}') \qquad \textbf{(19)}$$

where $q_{t+1}^{k}$ is the hidden state indexed as $k$ at time $t+1$. The key is to solve $P(q_{t+1}^{k})$, which can be done by the forward algorithm, $P_{q_{t+1}^{k}}(\boldsymbol{x}')$ is the context probability (Eq.10).

For Obj.2, we provide the medical team with a treatment prototype. This prototype $\boldsymbol{T}_r = [a_{t+1}, \dots, a_k]^T$ is a recommended sequence of activities obtained by greedy iterative maximization:

$$\boldsymbol{T}_r = \boldsymbol{T}_r \cup \{\arg\max_{e\in\boldsymbol{\Sigma}} P\big(q^j_{t+1}|q^i_t\big)P\big(e\big|q^i_t\big)P_{t_{ij}}(\boldsymbol{x}')\, P_{q_j}(\boldsymbol{x}')\} \qquad \textbf{(20)}$$

For Obj.3, we recommend a next intention $I_{t+1}$ to the medical team that maximizes:

$$I_{t+1} = \underset{q^j_{t+1}}{\operatorname{argmax}} P\big(q^j_{t+1}|q^i_t\big)\, P_{t_{ij}}(\boldsymbol{x}') \quad s.t.\ j = 1, \dots, |\boldsymbol{Q}|, \qquad \textbf{(21)}$$

**Table 7 Algorithm of Recommending Prototypical Process Enactment**

| **Algorithm: Recommending Prototypical Process Enactment** | |
|---|---|
| **Input:** $\boldsymbol{O}'$ , $\boldsymbol{\lambda}^{(H,c)}$, $\boldsymbol{x}'$ **,** current intention $I$ | |
| **Output:** $\boldsymbol{T}_r = [a_{t+1}, \dots, a_k]^T$ | |
| Step 1. | Find next level HMM $\lambda_i^{l+1}$; |
| Step 2. | Find the most likely path $p^{(i)} = \{\boldsymbol{Q}^{(i)},\ \boldsymbol{t}^{(i)}\}$ in $\lambda_i^{l+1}$ given an observed trace $\boldsymbol{O}'$ ; |
| Step 3. | Find current sub-intention $I_i = \underset{q^j\in\lambda_i^{l+1}}{\operatorname{argmax}} P_{q^j}(x')$ |
| Step 4. | **while** $a \neq$ "end", **do** |
| Step 5. | Find activity $a = \underset{e\in\boldsymbol{\Sigma}}{\operatorname{argmax}} P\big(e\big|I^j_{t+1}\big)\, P(I^j_{t+1}|I^i_t)$ |
| Step 6. | Add $\boldsymbol{T}_r = \boldsymbol{T}_r \cup \{a\}$; |
| Step 7. | Update the probability of recommended emissions and transitions; |
| Step 8. | **end while** |
| Step 9. | return $\boldsymbol{T}_r$; |

### 4.3 Process Recommendation Based on LSTM

In this section, we used long short-term memory (LSTM) to model temporal dependencies. Moreover, we modified the LSTM to incorporate with the patient demographics to provide the process recommendation. The goal of the proposed algorithm is to recommend the next-step treatment activities to the medical team based on the observed behavioral contextual information. The recommender system is mainly built on a RNN (Figure 16).The LSTM cell takes the concatenation of the activity vectors $v^{\alpha}$ and the activity attribute vectors $v^{\beta}$ as the input. The latent vector output from RNN cell is then merged again with the patient attribute vector $\boldsymbol{v}^{x}$ (auxiliary input, static environmental context). For the final output, we applied a fully-connected layer after the concatenation 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 next step (*t+1*).

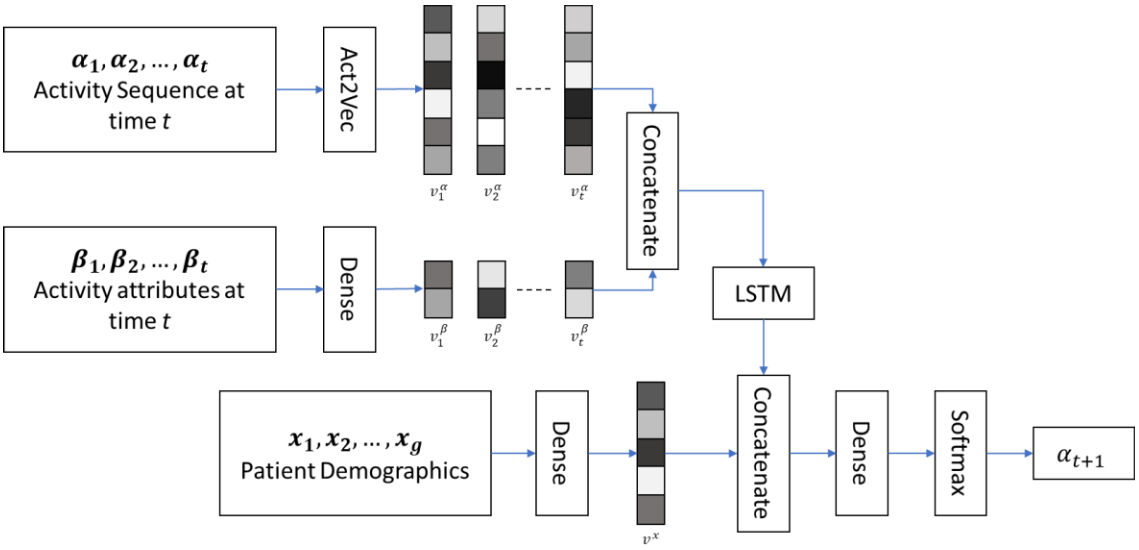

$\boldsymbol{\alpha_1, \alpha_2, \dots, \alpha_t}$
Activity Sequence at time *t*
Act2Vec
$v_1^\alpha$
$v_2^\alpha$
$v_t^\alpha$
Concatenate
$\boldsymbol{\beta_1, \beta_2, \dots, \beta_t}$
Activity attributes at time *t*
Dense
$v_1^\beta$
$v_2^\beta$
$v_t^\beta$
LSTM
$\boldsymbol{x_1, x_2, \dots, x_g}$
Patient Demographics
Dense
$v^x$
Concatenate
Dense
Softmax
$\alpha_{t+1}$

**Figure 16 Treatment Recommendation Framework Based on RNN**

And also, we use multiple contextual information as input. Given the treatment activity trace from time 1 to *t*: $\boldsymbol{T} = [\alpha_1, \dots, \alpha_t]^T$, the *i*-th activity $\alpha_i \in \boldsymbol{T}$ is embedded into a vector representation $\boldsymbol{v_t^\alpha}$. 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 completeness of the task. As this environmental context is also dynamically changing over time, we combined the environmental context $\boldsymbol{v_t^\beta}$ with activity vector $\boldsymbol{v_t^\alpha}$ at the same timestep *t* as the input of the LSTM network. The other auxiliary input, patient attribute $\boldsymbol{v^x}$s, is static over time. We thus integrate this information after the LSTM iteration over the input sequence. 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's gender and weight, may have little or no predictive power. Hence, we add a dense layer after the auxiliary input layer to help select the useful features from all of the patient demographic.

We used categorical cross-entropy as the loss function (Eq. 22).

$$\mathcal{L}(\hat{y}, y) = -\frac{1}{N}\sum_{i}^{N}\sum_{j}^{M} w_j (y_{ij} \log \hat{y}_{ij}) \qquad \textbf{(22)}$$

where *N* is the total number of predictions and *M* is the number of activity types. $\hat{y}_{ij}$ is the predicted probability for the activity type of index $j$. $y_{ij}$ is either 0 or 1 stating if the activity type of index $j$ is predicted correctly. Because of the imbalance distribution of activity types, we included the class weights $\boldsymbol{w} = [w_1, \dots, w_M]$ (Eq.23) so that the model can pay more attention to the underrepresented classes.

$$w_j = n_{samples}/(n_{classes} * n_{y_j}) \qquad \textbf{(23)}$$

## 4.4 Computational Result

Our goal is to correctly recommend the top-k 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. We adopted standard accuracy and top-k accuracy for evaluation. Standard accuracy measures the percentage of correct recommendations. Top-k accuracy the fraction of recommendations for which the correct label is among the top-k most probable predicted. Standard accuracy can be treated as top-1 accuracy.

To get a comprehensive evaluation of the proposed method, we employed 6 popular sequential modelling methods as baselines. POP and Act-KNN [29] are frequently used baselines in recommendation research. MC (Markov model) and HMM [31] (hidden Markov model) are classical sequential modelling methods. LSTM and GRU are state-of-the-art techniques recently introduced to recommender problems [29][30]. POP always recommends the most popular activities in the training data. Act-KNN recommends activities most similar to the current activity. The similarity here is the Euclidean distance between Act2vec embeddings. HMMs are able to model observations driven by latent (i.e., hidden) variables. The latent variable in medical processes can be understood as the treatment goals of the medical team.

We first evaluated the intermediate output of the Act2vec. We encoded 102 trauma activities into 100D vectors where the embedding window size is 5. And then, we applied TSNE to project the vectors onto a 2D plane (Figure 17), where each dot represents an activity. The color of the dots indicates the probable medical goals of the activity and the distance between dots shows the similarity of the activities. We also invited our medical experts to group the activities based on their domain knowledge to test whether the data-driven results make sense. The colors in Figure 17 indicate 12 different groups.

The visualization results reveal several insights here. First, activities under the same medical goal stay closer than under different medical goals. This finding indicates that the medical team usually performs the trauma resuscitation by finishing medical goals step by step rather than simultaneously. Representative exceptions are points 75 (visual inspection of left eye), 77 (visual inspection of right eye), 83 (assess left pupil), and 84 (assess right pupil). They belong to medical goal “assess head and face”, but are not close to other points under the same goal. Our medical expert explains that these four activities are usually skipped because the patients’ pupils are often checked ahead to calculate the patient’s Glasgow coma scale (i.e., a metric to assess whether the patient is in a coma).

Second, without considering the low-level groups (12 groups clustered by the medical experts), the activities points can be grouped into four major clusters (dashed circles in Figure 17). The clusters reveal the high-level medical goals. The left cluster constitutes 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.

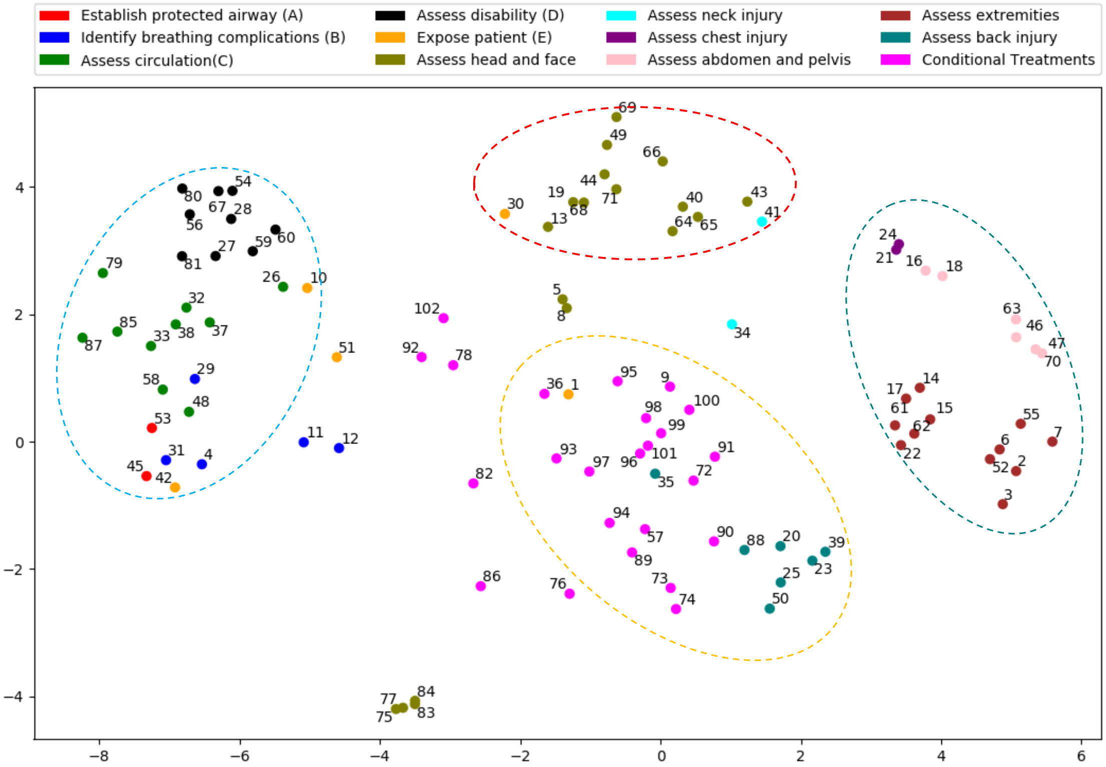


**Figure 17 Activity embedding (or Act2vec) visualization**

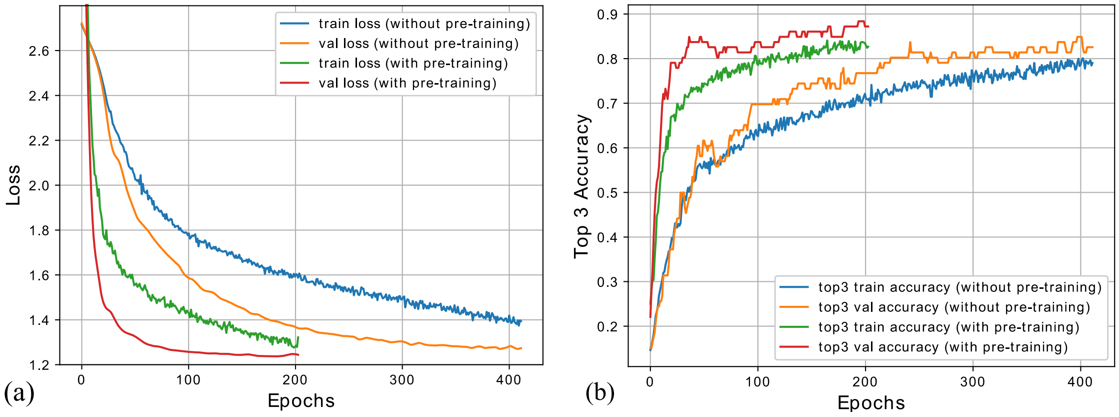


**Figure 18 Training and validation (a) loss and (b) accuracy plot for model with and without pre-training.**

We performed offline evaluation of our recommender model. For our method and all baseline models, we divided the dataset into the training, validation, and testing sets in a 0.8:0.1:0.1 ratio. Our experimental results (Table 8) show that deep learning (models 5-14) outperforms conventional recommendation methods (models 1-5). As an important baseline, POP only achieved 0.031 accuracy and 0.136 top-5 accuracy 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 associations between activities. Without a predictive model, Act-KNN using proximity alone has limited prediction power. Classical sequential models like MC and HMM achieved much higher accuracy. As they depend on first-order Markov assumptions, each prediction is only based on the immediate previous state. Their high accuracy reveals the strong dependency between adjacent activities. By considering both adjacent and long-term dependencies, the RNN methods achieved the best performance. However, the design of the network also affects performance.

**Table 8 Model performance comparison on two real-world medical datasets**

| **Datasets**<br>**Models** | | **Trauma** | | **Intubation** | |
|---|---|---|---|---|---|
| | | top-1 | top-5 | top-1 | top-3 |
| **1** | POP | 0.031 | 0.136 | 0.098 | 0.302 |
| **2** | Act-KNN | 0.177 | 0.479 | 0.125 | 0.280 |
| **3** | MC | 0.324 | 0.601 | 0.427 | 0.648 |
| **4** | HMM | 0.344 | 0.569 | 0.395 | 0.750 |
| **5** | LSTM + One-hot | 0.155 | 0.633 | 0.260 | 0.753 |
| **6** | GRU + One-hot | 0.159 | 0.627 | 0.225 | 0.711 |
| **7** | LSTM + Embedding | 0.393 | 0.631 | 0.410 | 0.773 |
| **8** | GRU + Embedding | 0.395 | 0.634 | 0.436 | 0.680 |
| **9** | LSTM + Embedding + Environmental Context | 0.387 | 0.640 | 0.350 | 0.722 |
| **10** | GRU + Embedding + Environmental Context | 0.393 | 0.645 | 0.385 | 0.711 |
| **11** | LSTM + Embedding + Environmental Context (Dense) | 0.397 | 0.644 | 0.444 | 0.701 |
| **12** | GRU + Embedding + Environmental Context (Dense) | **0.405** | 0.646 | 0.461 | 0.742 |
| **13** | Pre-train(LSTM+Embedding+Environmental Context (Dense)) | 0.400 | 0.643 | 0.436 | 0.753 |
| **14** | Pre-train (GRU+Embedding+Environmental Context (Dense)) | 0.399 | **0.655** | **0.462** | **0.773** |

The RNN with embedding layer (Act2vec, models 7&8) achieved much higher top-1 accuracy than the RNN with one-hot vector representation (models 5&6). This is expected, as the skip-gram training of Act2vec take neighboring activities (a form of low-order logic) into account. HMM/MC performs well with just first-order logic, so Act2vec would intuitively help the deep model.

Our results (models 9&10) also show that simply concatenating the attribute vectors into the input did not help prediction. Redundant attributes do not usually help prediction and may actually harm performance. We thus placed a fully-connected layer before the merging layer to reduce the dimensionality of the attribute vector. The improved model increased top-1 prediction accuracy by ~1% on trauma data and ~8% on intubation data respectively (models 11&12 vs. models 9&10).

Pre-training on the synthetic data may also improve model performance. On intubation data, the pre-trained model (models 13&14) of setups “GRU, Embedding, and Environmental Context” achieved a higher top-1 and top-3 accuracy than models of the same setups but without pre-training (models 13&14). This implies the synthetic data helps generalize the model. Compared with randomly initialized model, the validation loss of the pre-trained model converges faster (Figure 18). But on trauma data, pre-training with synthetic data did not improve the performance. This may be caused by the complexity of the trauma resuscitation process. Each trauma treatment sequence has 95 activities in average and each activity belongs to one of the 102 types. Besides, the order of treatment activities can be performed in numerous ways. Compared to such complexity, our observed 122 cases may still be too sparse. And the variations learnt from the synthetic data in the pre-training may still be insufficient to regularize the model.

# CHAPTER 5

# CONCLUSION

## 5.1 Summary of Thesis Contribution

This thesis began by developing the medical treatment process recommendation system. Firstly, we extended the Hidden Markov Model (HMM) into hierarchical structure and thus tried mining different levels of intentions. Furthermore, context attributes were then incorporated using the Viterbi algorithm so that the model offers recommendations better tailored to specifics of the situation. Recommended treatment procedures were generated step-by-step during the resuscitation. Process deviations were identified by comparing practical procedures to intention-specific treatment prototypes. We have shown that the discovered intention groupings align with medical expert knowledge. Our results showed the potential for intention mining at making adaptable recommendations to the medical team and helping them reduce errors.

Secondly, we tried to provide the process recommendation in deep learning approach. Our system was built on recurrent neural networks. The networks took both environmental and behavioral contextual information as input, and outputs next-step treatment suggestions. We contributed different approaches to enhance our model. First, we proposed Act2vec to embed different activity types into numerical vectors. Second, we designed the sliding-window attention to assist model prediction. Third, we developed a novel synthetic patient data generation algorithm. We used the generated synthetic data to pre-train our neural networks, addressing our problem of limited amount of data. Our numerical results showed our recommender system achieved improvement compared to the baseline methods. Our visual analytics extracted interesting knowledge and insights.

## 5.2 Future Work

Some computer-aided decision support systems and expert-derived algorithms have been proposed to reduce medical team errors and improve patient outcomes for trauma resuscitations. These initial attempts, despite being carefully designed by medical experts, have had limited success because: (a) expert-derived knowledge-based models can be biased, (b) the algorithms' rules were meant for general trauma patients and so ignored each resuscitation's contextual attributes, and (c) the approaches lacked generalizability and applicability to other practice settings.

We presented a data-driven intention-aware context-based trauma resuscitation process recommender and diagnostic system. However, the accuracy does not meet the requirement in the trauma resuscitation field. Due to the lack of data, lots of the transition between activities was not discovered by our models. It might be a solution that we develop a system based on a simple model proposed by medical experts, and thus we optimize the model by training it using large scale of datasets.

# REFERENCE

1. Demetriades D, et.al. Trauma deaths in a mature urban trauma system: is "trimodal" distribution a valid concept? J Am Coll Surg. 2005 Sep;201(3):343-8. PubMed PMID: 16125066.
2. Carter EA, et al. Adherence to ATLS primary and secondary surveys during pediatric trauma resuscitation. Resuscitation. 2013 Jan;84(1):66-71. PubMed PMID: 22781213.
3. Clarke JR, et al. An objective analysis of process errors in trauma resuscitations. Acad Emerg Med. 2000 Nov;7(11):1303-10. PubMed PMID: 11073483.
4. Khodabandelou, G., Hug, C., Deneckère, R., & Salinesi, C. (2014, May). Unsupervised discovery of intentional process models from event logs. In Proceedings of the 11th Working Conference on Mining Software Repositories (pp. 282-291). ACM.Khodabandelou, G., Hug, C.,
5. Gröger, Christoph, et al. "Prescriptive analytics for recommendation-based business process optimization." International Conference on Business Information Systems. Springer International Publishing, 2014.
6. van der Aalst, W.M.P., Weijters, A.J.M.M., Maruster, L.: Workflow mining: Discovering process models from event logs. IEEE Transactions on Knowledge and Data Engineering 16(9), 1128–1142 (2004)
7. Bose, R. J. C., & van der Aalst, W. (2010, September). Trace alignment in process mining: opportunities for process diagnostics. In International Conference on Business Process Management (pp. 227-242). Springer, Berlin, Heidelberg.
8. Wen, T. H., Gasic, M., Mrksic, N., Su, P. H., Vandyke, D., & Young, S. (2015). Semantically conditioned lstm-based natural language generation for spoken dialogue systems. arXiv preprint arXiv:1508.01745.
9. Goodfellow, I., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., ... & Bengio, Y. (2014). Generative adversarial nets. In Advances in neural information processing systems (pp. 2672-2680).
10. Bachman, P., & Precup, D. (2015). Data generation as sequential decision making. In Advances in Neural Information Processing Systems (pp. 3249-3257).
11. Yu, L., Zhang, W., Wang, J., & Yu, Y. (2017, March). SeqGAN: Sequence Generative Adversarial Nets with Policy Gradient. In AAAI (pp. 2852-2858).
12. 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.

13. Mavromatis, Panayotis. "Minimum description length modelling of musical structure." Journal of Mathematics and Music 3.3 (2009): 117-136.

14. Herbst, Joachim, and Dimitris Karagiannis. "Integrating machine learning and workflow management to support acquisition and adaptation of workflow models." Database and Expert Systems Applications, 1998. Proceedings. Ninth International Workshop on. IEEE, 1998.

15. Stolcke, Andreas, and Stephen M. Omohundro. "Best-first model merging for hidden Markov model induction." arXiv preprint cmp-lg/9405017 (1994).

16. Rabiner LR. A tutorial on hidden Markov models and selected applications in speech recognition. Proc. IEEE, vol.77, no.2, pp. 257-286, February 1989.

17. Gröger, Christoph, Holger Schwarz, and Bernhard Mitschang. "Prescriptive analytics for recommendation-based business process optimization." International Conference on Business Information Systems. Springer International Publishing, 2014.

18. Sun, Leilei, et al. "Data-driven Automatic Treatment Regimen Development and Recommendation." Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining. ACM, 2016.

19. Bernhard, Michael, et al. "Introduction of a treatment algorithm can improve the early management of emergency patients in the resuscitation room." Resuscitation 73.3 (2007): 362-373.

20. Hochreiter, Sepp, and Jürgen Schmidhuber. "Long short-term memory." Neural computation 9.8 (1997): 1735-1780.

21. Cho, Kyunghyun, et al. "Learning phrase representations using RNN encoder-decoder for statistical machine translation." arXiv preprint arXiv:1406.1078 (2014).

22. T. Mikolov, M. Karafi´at, L. Burget, J. Cernock`y, and S. Khudanpur. Recurrent neural network based language model. In INTERSPEECH, pages 1045–1048, 2010.

23. D. Rumelhart, G. E. Hinton, and R. J. Williams. Learning representations by back-propagating errors. Nature, 323(6088):533–536, 1986.

24. M. Sundermeyer, R. Schluter, and H. Ney. LSTM neural networks for language modeling. In INTERSPEECH, 2010.

25. Sutskever, I., Vinyals, O., & Le, Q. V. (2014). Sequence to sequence learning with neural networks. In Advances in neural information processing systems (pp. 3104-3112).

26. Demetriades D, et.al. Trauma deaths in a mature urban trauma system: is "trimodal" distribution a valid concept? J Am Coll Surg. 2005 Sep;201(3):343-8. PubMed PMID: 16125066.

27. Carter EA, et al. Adherence to ATLS primary and secondary surveys during pediatric trauma resuscitation. Resuscitation. 2013 Jan;84(1):66-71. PubMed PMID: 22781213.

28. Clarke JR, et al. An objective analysis of process errors in trauma resuscitations. Acad Emerg Med. 2000 Nov;7(11):1303-10. PubMed PMID: 11073483.

29. Choi, Edward, et al. "Doctor ai: Predicting clinical events via recurrent neural networks." Machine Learning for Healthcare Conference. 2016.

30. Yu, Feng, et al. "A dynamic recurrent model for next basket recommendation." Proceedings of the 39th International ACM SIGIR conference on Research and Development in Information Retrieval. ACM, 2016.

31. Siddiqi, Sajid M., et al "Fast state discovery for HMM model selection and learning." Artificial Intelligence and Statistics. 2007.

32. Chen, S., Yang, S., Zhou, M., Burd, R., & Marsic, I. (2017, November). Process-oriented Iterative Multiple Alignment for Medical Process Mining. In Data Mining Workshops (ICDMW), 2017 IEEE International Conference on (pp. 438-445). IEEE.

33. Yang, S., Zhou, M., Chen, S., Dong, X., Ahmed, O., Burd, R. S., & Marsic, I. (2017, August). Medical Workflow Modeling Using Alignment-Guided State-Splitting HMM. In Healthcare Informatics (ICHI), 2017 IEEE International Conference on (pp. 144-153). IEEE.

34. Yang, S., Ni, W., Dong, X., Chen, S., Farneth, R. A., Sarcevic, A., ... & Burd, R. S. (2018, June). Intention Mining in Medical Process: A Case Study in Trauma Resuscitation. In 2018 IEEE International Conference on Healthcare Informatics (ICHI) (pp. 36-43). IEEE.