unibz

Freie Universität Bozen
Libera Università di Bolzano
Università Liedia de Bulsan

Fakultät für Informatik
Facoltà di Scienze e Tecnologie informatiche
Faculty of Computer Sciences

Bachelor in Computer Science and Engineering

Bachelor Thesis

Mining Data to Model System Usage with Hidden Markov Models

Candidate: Daniele Gadler

Supervisor: Barbara Russo

Co-Supervisor: Andrea Janes

July 21, 2017

Abstract

Process mining is a technique aimed at building process models from event logs of a system. Events and their interactions represent usage sequences of a system. In our case, events represent user interactions with a system. Current state-of-the-art tools can provide process managers with different graphical views of such models. Any of these resulting views can then provide managers with a selection of log events from a log and their interactions. Managers can then use these representations to compare them with an ideal process model or to support process improvement.

However, these views lack any recommendation of real tasks that users want to achieve through interactions with the system (i.e., paths through logs) compounding the process. Thus, although process managers can reduce the view of graphs produced by state-of-the-art tools to a more readable representation, they still have to subjectively recreate the tasks of their process directly from logs at their disposal, based on their experience and knowledge of the system.

We designed an automatic approach to identify the tasks of a process by automatically creating Hidden Markov Models (HMMs) based on usage logs of a system.

The resulting HMMs comprise a set of tasks representing the process that current state-of-the-art algorithms do not create. We can then provide managers with a pool of such models selected by best accuracy.

We applied our approach to one-year and one-month logs collected from users of an industrial mobile ERP system in use at a company located in South Tyrol, Italy. We re-created five versions of the underlying execution process automatically and compared the automaticallly-built models with a manually-built model, constructed by interviewing the manager. We compared the accuracy of the models built automatically and the one built with the process manager and discussed the obtained models with him. We found that the automatically-built HMMs can represent models with high quality and can well compound the manager's process idea, provided the log dataset is accurately chosen. In particular, the HMM produced from the one-month dataset generated a model with higher accuracy than the one built from the one-year dataset and its expressiveness was similar to the one of the model built manually with the company's stakeholder.

Zusammenfassung

Process Mining ist eine Technik zum Aufbau von Prozessmodellen aus Ereignisprotokollen. In unserem Fall meint man damit Interaktionen eines Nutzers mit dem System. Derzeitige aus dem aktuellen Stand der Technik stammende Tools stellen Prozess-Managern verschiedene graphische Darstellungen der obengenannten Modelle zur Verfügung. Diese Darstellungen können dann Prozess-Managern eine Auswahl der Log-Ereignisse und deren Interaktionen anzeigen. Prozess-Manager können dann diese Prozessdarstellungen mit einem idealen Prozess zur Optimierung und Verbesserung des Prozesses vergleichen.

Leider sind in den so generierten Ansichten keine übergeordneten Tasks sichtbar (d.h. Aufgaben die mehrere Ereignisse betreffen, sich also als Pfade durch die Logs darstellen).

Ein Prozess Manager kann zwar die graphische Ansicht eines Prozesses zu einer lesbareren Darstellung reduzieren, aber er muss die Tasks des Prozesses von den zur Verfügung stehenden Ereignisse selbst erstellen. Die Identifikation von Tasks ist daher subjektiv und ausschließlich auf der Erfahrung und Kenntnisse des Prozess-Managers basiert.

In dieser Arbeit haben wir einen automatischen Ansatz zur Identifikation von Prozess-Tasks, d.h., übergeordneten durch die iterative Erstellung von auf Ausführungslogs basierten Hidden Markov Models (HMMs) entworfen. Die resultierenden HMMs identifizieren eine Menge von Tasks, die aktuelle Algorithmen nicht identifizieren. Wir können so Prozess-Managern eine Auswahl von Modellen präsentieren, die aufgrund der besten Genauigkeit ausgewählt worden sind.

Wir haben unseren Ansatz mit einjährigen und einmonatigen Logs angewendet: die Daten wurden von Nutzern eines industriellen mobilen ERP Systems, das in einer Firma in Südtirol verwendet wird, gesammelt. Wir haben fünf Versionen des zugrundliegenden Ausführungsprozesses automatisch neu erstellt und haben die resultierenden Modelle mit einem manuell aufgebauten Modell verglichen, welches durch die Befragung des Prozess-Managers aufgebaut wurde. Wir haben die Genauigkeit der automatisch aufgebauten Modelle mit der Genauigkeit des manuell aufgebauten Modells verglichen und haben die erhaltenen Modelle mit dem Prozess Manager besprochen. Wir haben herausgefunden, dass automatisch aufgebaute HMMs hochwertige Modelle erstellen können und die Prozessidee des Managers gut identifizieren können, vorausgesetzt dass das Log Dataset gut ausgewählt ist. In unserem Fall hatte das vom einmonatigen Dataset generierte HMM eine höhere Genauigkeit als das vom einjährigen Dataset generierte. Außerdem repräsentierte das vom einmonatigen Dataset generierte HMM die Prozessidee des Prozess Managers in einer ähnlicher Weise wie das vom Prozess Manager manuell aufgebaute HMM.

Sommario

Il Process Mining è una tecnica finalizzata alla costruzione di modelli di processo a partire da log di eventi. Nel nostro lavoro gli eventi descrivono le interazioni di un utente con il sistema. Attualmente esistono applicazioni software che permettono di rappresentare graficamente sequenze di tali eventi a diversi livelli di granularità. Un process manager può quindi confrontare le diverse rappresentazioni con un modello ideale di uso del sistema, per ottimizzare o migliorare il processo. Tuttavia, a tal fine, le rappresentazioni in termini dei singoli eventi di log risultano troppo a basso dettaglio per descrivere l'uso di un sistema da parte dell'utente. Il nostro lavoro vuole quindi determinare le sequenze di eventi che possono essere associate ai task compiuti dall'utente nel sistema e che descrivono l'uso e le funzionalità del sistema stesso.

Pertanto, abbiamo progettato e implementato un metodo automatico per l'identificazione di task di un utente. Tale metodo è basato sulla costruzione automatica di Modelli Markoviani a stati latenti (in inglese, Hidden Markov Model, HMM), a partire dai log di uso di un sistema.

A differenza delle applicazioni software sopraccitate, il nostro sistema è quindi in grado di creare modelli rappresentanti i task di un utente in maniera completamente automatica. Dall'intero pool di modelli generati, si possono estrarre i modelli con qualità migliore: questi andranno a supportare il process manager nel miglioramento del processo.

Il nostro metodo è stato applicato ai dati raccolti dall'utilizzo di un sistema mobile ERP attualmente in uso presso un'azienda situata in Alto Adige. I dati sono stati raccolti in un periodo di un anno: nel nostro caso di studio abbiamo considerato sia l'intero arco temporale di un anno, sia un periodo limitato a un singolo mese. Per entrambi i periodi considerati, abbiamo quindi ricreato cinque modelli di uso del sistema applicando il nostro metodo di creazione automatica di HMM. Abbiamo inoltre creato una HMM manualmente durante un colloquio con il process manager. Previo confronto dei modelli generati automaticamente con quello generato manualmente, abbiamo infine scoperto che le HMM generate con il nostro metodo automatico possono ben rappresentare l'idea di processo del manager, a condizione che il dataset sia scelto accuratamente.

I risultati hanno evidenziato che la HMM generata automaticamente dal dataset di un mese possedeva una precisione maggiore rispetto alla HMM generata automaticamente dal dataset di un anno. Inoltre, la HMM generata automaticamente dal dataset di un mese ben rappresentava l'idea di processo del process manager, analogamente alla HMM creata manualmente insieme al process manager.

Acknowledgement

I would really like to thank my supervisor Prof. Barbara Russo for all of her support and opportunities offered in my Bachelor. I am most grateful to her for introducing me to the world of research in software engineering and especially for allowing me to get involved in many other activities alongside my Bachelor, both in teaching and researching. It is also thanks to her that I could carry out an internship in China as part of my Bachelor.

I would also like to thank Dr. Andrea Janes for his patience and availability throughout this one year of thesis' work and Michael Mairegger for the validation of our method in the experimental phase. The present work was mostly carried out during my last year as an Erasmus student at the Technische Universität Kaiserslautern in Germany, while my supervisors were located in Bolzano and this was at times very challenging, both from a communicational and organizational perspective. However, my supervisors' patience certainly helped overcome these issues.

I would also like to thank my girlfriend 鲁清钦 for her constant help and for standing by me in this whole year. She is the special the one that made my Erasmus in Kaiserslautern worth it. My thanks also must go to my friends in Kaiserslautern: particularly Jonas and Elly, who made my courses at the TU Kaiserslautern more enjoyable, as well as Alka and Praveen, for being great group mates. My sincere thanks also go to Prof. Schmitt and Carolina Nogueira for introducing me to the world of network security and the internship opportunity offered to me in Kaiserslautern and finally Tanittha Sutjaritvorakul for introducing me to computer graphics for pedestrian detection.

From my first two years at the Free University of Bozen-Bolzano, firstly I'd like to thank my all of study colleagues, who have made my study period truly enjoyable. In particular the Bangla team: Saifur, Abdullah, Shantunu, Stefan, with whom I shared great moments and could experience some real teamwork. My thanks also go to all members of the Makerspace, especially Julian, Lorenzo, Angelo and to all of friends in Bolzano with whom I shared a great period, both in classes and outside classes: the genius Michele, Filippo and Enxhi, Giulia, Marta and many many others. I am share I am going to miss everyone of you in the future.

I am certainly indebted to my lecturers Prof. Artale and Dr. Razniewski for their very effective recommendation letters. My thanks also go to Prof. Leonardo Ricci for being a very motivating teacher and making maths a very interesting subject, Prof. Widmann for making German worth learning, Dr. Xiaofeng Wang and Pekka for their inspiring lectures in my first year, as well as all other lecturers I had. Finally, I would like to thank all the members of the Faculty of Computer Science for their dedication and commitment to the research and teaching activities of the Faculty, especially the Dean Prof. Francesco Ricci.

Last but not least, I'd like to thank my parents for allowing me to begin and conclude my studies and my sister.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

1.1. Process Mining

The term “process mining” describes a set of techniques with the goal of studying processes by extracting knowledge from process traces, i.e., logs that document events that happened because a process was executed [1].

In a first step, a process model is generated using the event log (this step is called *“process discovery”*). The “discovered” or “mined” process model can then be used to compare it with an ideal process model (this activity is called *“conformance checking”*) and / or to support process improvement, i.e., to study processes to find bottlenecks, unexpected delays, observe the distribution of work, etc.

With process mining, process models are rendered as directed graphs, where nodes represent events and directed edges are transitions between log events. State-of-the-art algorithms (e.g. Fuzzy miner [2]) can provide process managers with different views of such models by clustering or removing rare nodes or edges. At the end, any of these views presents managers only with a selection of events and their interactions, but lack any recommendation on the real tasks (i.e., a path through a sequence of events) compounding the process. Thus, although managers can reduce the view of the graph to a more readable representation, they still have to subjectively recreate process tasks solely from the event logs at their disposal.

Process mining tools can present managers with graphs extracted from event logs at different granularities, as illustrated in the two graphs in Figure 1.2. In particular, the state-of-the-art algorithm Fuzzy Miner [2] used in such tools (e.g. PRoM) enables for different representations of the log graphs by aggregating subsequent transactions between logs (i.e., node pairs) that rarely appear or removing the least frequent nodes or transactions. The different representations are obtained by sliding up and down thresholds for significance (e.g., frequency) or correlation (pairwise occurrence of nodes) of nodes or edges. To motivate our study, we illustrate different graphical representations of the logs of the mobile ERP system provided by the Fuzzy Miner algorithm in DISCO [3]. Different representations of the process are obtained by sliding up and down thresholds for significance (e.g., frequency) or correlation (between nodes) of nodes or edges, as shown in Figure 1.1. With the "Spaghetti" graph on the left-hand side graph of Figure 1.2, the user is presented with all details without distinguishing what is important and what is not. In the more clustered and abstract representation in the right-hand side graph of Figure 1.2, the graph increases in readability, but might miss relevant details or semantics relevant for the process.

1.2. Motivation

When logs record the use of a software system, log graphs represent user's chains of actions to the system needed to complete tasks. Therefore, managers may trace tasks performed by a user as paths in log graphs.

Being tasks strictly connected with a system's functioning, managers can achieve a deeper understanding of the most used functionalities in their system by looking at generated process views.

Unfortunately, the identification of paths in log graphs that represent user's tasks is solely dependent on the managers' knowledge of the underlying system. Namely, process mining tools do not provide support to recommend which path actually corresponds to which user's task. Furthermore, tasks can be represented as a graph by different sets of log events, based on the view's granularity. So, which view would better help managers to recognize tasks in log graphs is also an open issue.

Through the work presented in this thesis, we want to give managers an automated tool that identifies such tasks and profiles application usage of the system. With our approach, managers are provided with a solid baseline from which they can base their decisions.

1.3. Goal

Our goal is to build a model from the logs that best represents the tasks performed by the users to a system. One way is to cluster events logs *manually* (i.e., events logs are split into subsets and for each subset, a process model is created [4]) according to the manager's knowledge of the system. This can be performed with process mining tools mentioned in Section 1.1; however, all these tools require the intervention of the process manager.

A *semi-automatic* approach is the one of Damevski *et. al.* [5], who build a Hidden Markov Model (HMM) iteratively by involving the stakeholder (in their case study case, testers in a software development process) only in the selection of the input for the next iteration. Thus, both approaches require the stakeholder to be involved in the HMM building.

In this study, we propose a method to iteratively build HMMs in an *automatic* way by applying the HMM theory to the logs under study with no manager's intervention. We then conduct an experiment to compare the three approaches (manual, semi-automatic and automatic) on use logs the model constructed iteratively with the manager, the model built automatically and the best DISCO model selected by the manager.

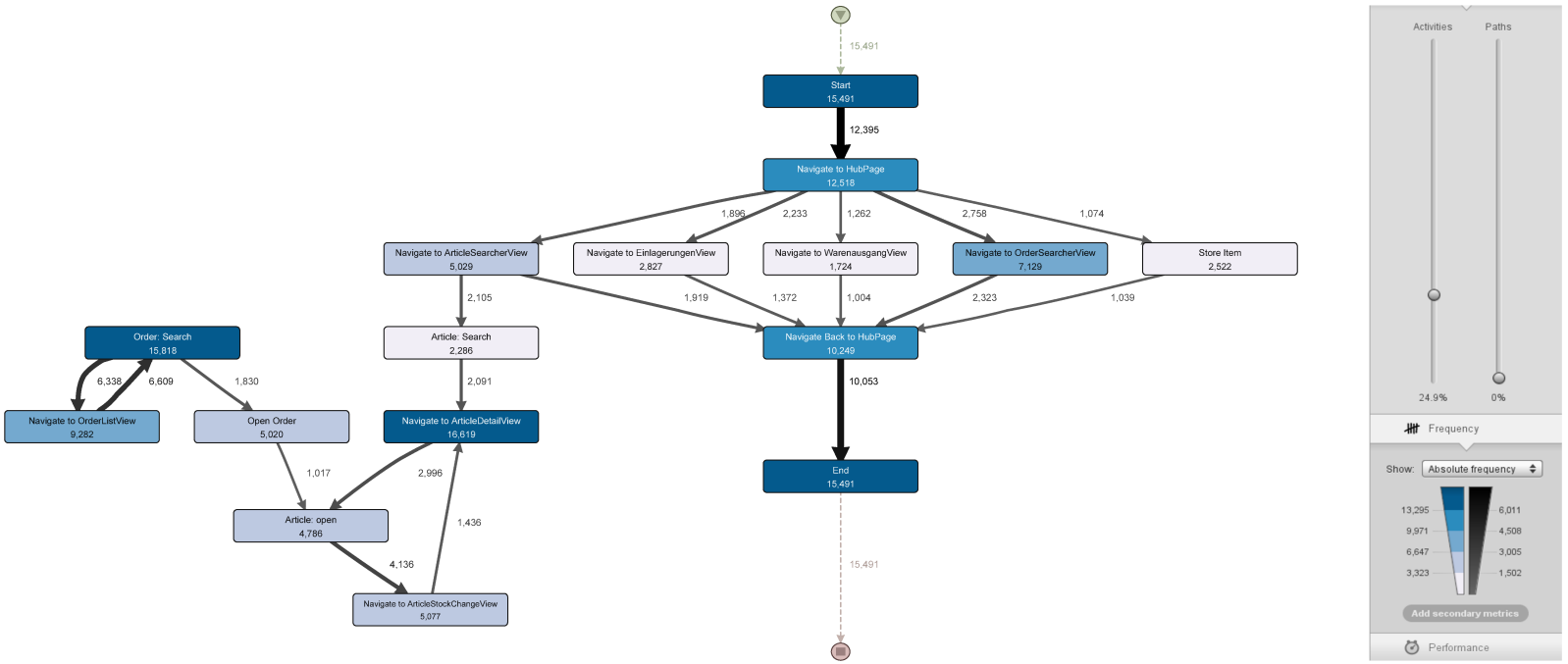


Figure 1.1: Example Graph tuning options in DISCO

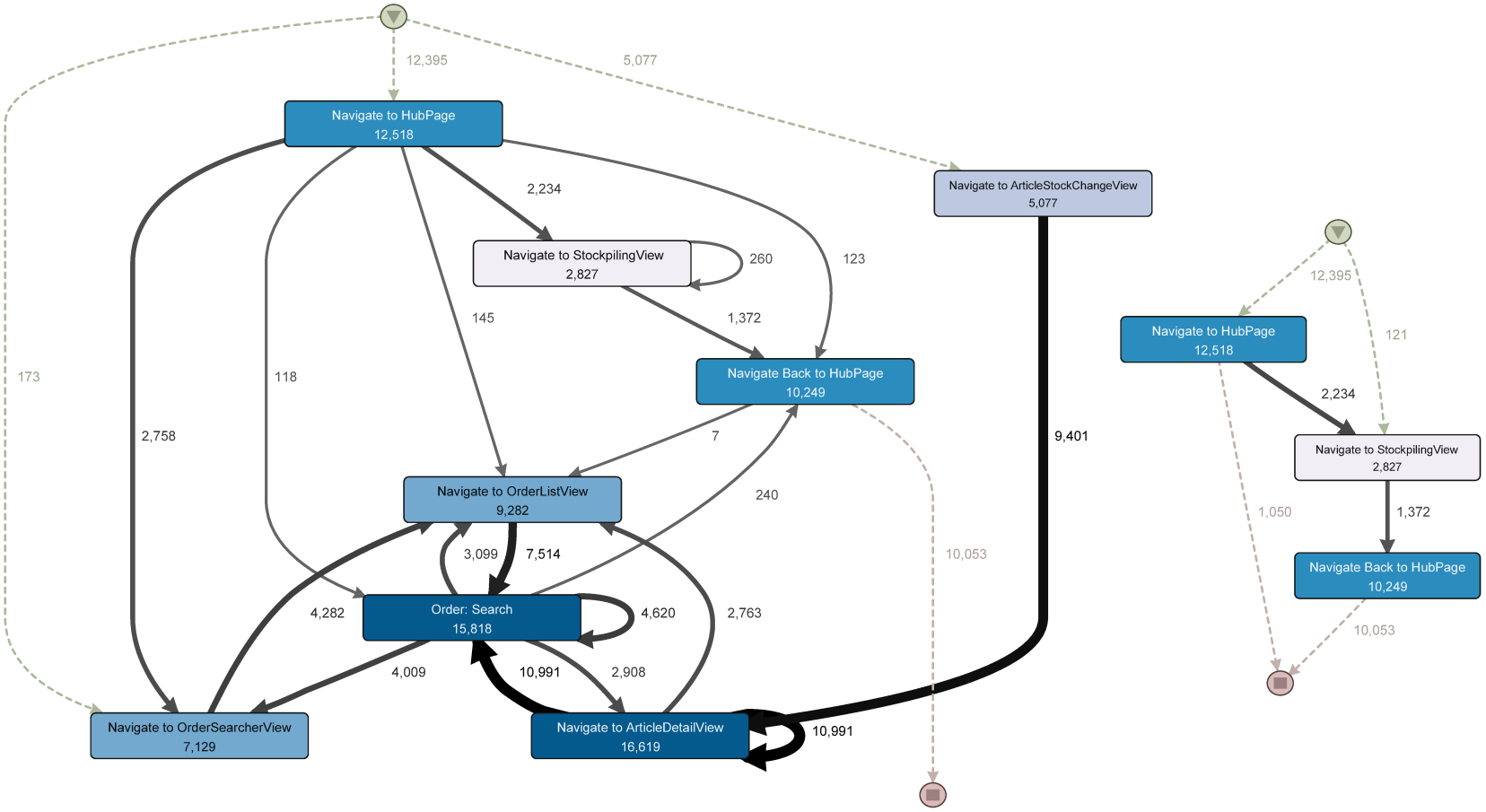


Figure 1.2: Different renderings of the data log in use, obtained by tuning significance and correlation of the Fuzzy miner algorithm in DISCO

1.4. Hidden Markov Models

Among several statistical modeling methods at disposal, a particular technique has found relevant application in computer science to model complex problems and to solve them: *Hidden Markov Model* (from now on, HMM). In the last few decades, HMMs have found application in a wide array of fields, such as speech recognition [6], handwriting recognition [7], text translation [8], gene prediction [9]. HMMs were recently used by [5] *et al.* for reconstructing debugging behavior and capturing system traces.

In the present thesis, we use HMMs for their capability of capturing hidden states in complex events. In fact, HMM hidden states can be associated to tasks or developer intentions' [5] and the transition probability of switching from a task to another is also given in HMMs in the form of a transition matrix. The Probability of a symbol to be emitted in a certain state is also given in the form of an emission matrix. Furthermore, we make use of a Markov Process to represent represent usage interactions in tasks because of the Markov Property. In our case, we assume that only the present task affects the future task and past tasks have no relevance for the future (i.e: a task is independent of tasks that occurred before it).

One issue of HMMs is the difficulty to interpret HMMs constructed with state-of-the-art algorithms, which are described in Section 2.5. Once such algorithms are applied to HMMs constructed from large datasets, they often produce models that are very complex and therefore difficult to interpret [5].

To tackle this issue, we construct HMMs in two manners and compare the resulting models with a process manager:

- *Manually*: We build HMMs based on a set of symbols hand-picked by a field expert (a process manager), out of a set of interesting sequences in the dataset.
- *Automatically*: We build HMMs automatically, based on the k top-interesting sequences from the dataset.

In-depth details concerning the theory underlying HMMs are given in Sections 2.2 to 2.6

1.5. Contributions

In this thesis, we extend Damevski *et al.*'s work [5] by giving the following contributions:

- *Creation of an automated tool for HMM Construction*: Instead of solely relying on the manual iterative creation of HMMs, we devised a semi-automated approach for HMM construction, where we consider the k top interesting sequences iteratively. This approach is described in Chapter 3.
- *Re-Engineering of Damevski et al.'s Algorithms in R*:

  We implemented the different algorithms described by *Damevski et al.* in R, as it is a programming language apt for data processing and statistical analysis. In the re-engineering phase, we also corrected and adapted several parts of their algorithms. The modified algorithms are provided in Sections 3.2.2 and 3.2.3.

- *Application and Evaluation of Damevski et al.'s method to Process Mining*: While Damevski *et al.* apply their proposed method to the modeling of developer debugging behavior, we make use of their proposed method with appropriate corrections in process mining through an industrial study case. The evaluation of the proposed method through the considered study case is described in Chapter 4. We then validate the model obtained through the automated construction together with the process manager. Validation is described in Chapter 5.

- *Correction of Machine-Learning algorithm 'Baum-Welch' from R package 'HMM'*: While implementing Damevski et al.'s algorithms in R, we also corrected the 'Baum-Welch' algorithm from the package 'HMM', as described in Section 3.2.1.

1.6. Related Work

The present work is mainly based on the study by K. Damevski, H. Chen, D. Shepherd, and L. Pollock [5], who apply an interactive approach for HMM construction for the discovery of developer interaction based on interesting sequences in the study of IDE debugging behavior. In their proposed approach, interesting sequences consist of sequences of symbols that can possibly give rise to a new state. Interpretation of interesting sequences is performed manually by a human expert, who decides whether a new state could better capture a set of interesting sequences. Interesting sequences consist of sequences that well represented by the data, but not by the model (HMM).

Each interesting sequence has a score, which is computed as the frequency of an interesting sequence in the data minus the probability of the interesting sequence to appear in the model.

Hence, very interesting sequences characterized by a high score are sequences well captured by the data, but not well captured by the model.

Should an expert decide to add a new state to the HMM based on the interesting sequences found, symbols present in the interesting sequences picked will be constrained to that new state and a new iteration of the algorithm is triggered, where the HMM model is trained with the Baum-Welch Algorithm.

Originally, interactive HMM construction based on interesting sequences was proposed by S. Jaroszewicz in [10] and [11], who deems this approach as able to "produce accurate but easy to understand models". As application field, Jarowzwewicz applies this method to the analysis of website visitor's behavior and manages to extrapolate a good model for website usage.

In this thesis, instead, we mine usage logs of a system. Logs are one of the major data sources used in building prediction models in empirical software engineering [12]. System logs have been extensively used in diagnosing faults, scheduling applications and services of hardware [13], [14], or in dependable computing, and in computer networks management and monitoring [15], [16]. In software engineering, system logs have been employed to model the workflow design of the activity processes [1], like Petri nets or to predict system misbehavior [17]. There are few papers that mined use logs, as use logs are not automatically collected by modern logging services (Section 4.2). Recently, Astromskis *et al.* [18] mined use logs to understand the usability of the same ERP system by analyzing the frequency of users' chains of interactions. Although the same dataset of their study is being used in the present work, this thesis has a completely different scope and makes use of a

different method.

1.7. Thesis Structure

This thesis is divided into seven main chapters:

1. *Introduction*: We introduce the main concepts underlying this thesis, describe the motivation for tackling the considered problem and the main goal we want to achieve. We also lay out the contributions it provides and give an overview of the related work.

2. *Background Information*: We provide an essential knowledge base needed to understand the content of this thesis. This knowledge base encompasses mainly probability theory, Markov models, their functioning and applications.

3. *Method Design*: We describe the different phases comprising the method designed for achieving the goal of building a log-based model that best represents the tasks performed by the users to a system.

4. *Evaluation*: We validate the proposed method by carrying out an experiment from a datalog obtained from an industrial in-use application. We compared our automatic approach with the iterative one laid out by Damevski *Et al.* [5] and the process mining tool DISCO.

5. *Discussion*: We summarize and discuss the results obtained by comparison of the considered approaches.

6. *Conclusions*: We lay out the conclusions obtained by the present work.

7. *Future Work*: We suggest how the proposed approach could be expanded and improved further, as well as possible future applications of the approach.

Chapter 2

Background Information

The definitions laid out in this section are based on Ross' Book "Introduction to Probability and Statistics for Engineers and Scientists" [19].

2.1. Probability Theory

Probability theory is a fundamental branch of mathematics, which has been developing since the 17th century to analyze events and the likelihood of their outcomes. In the 19th century, Laplace formalized this theory, laying the basis to classical probability theory.

A *Sample Space* is said to be the set of all possible outcomes of an experiment, hence a sample space Ω is defined as:

$$\Omega = \{\omega_1, \omega_2, ..., \omega_n\}, \text{ with } \omega_n \text{ being a single outcome.}$$

An *Outcome* $\omega \in \Omega$ is a single result of an experiment that produces a sample space Ω.

For example, the sample space of a die roll for a die with six faces is:

$$\Omega = \{1, 2, 3, 4, 5, 6\}, \text{ with each 1,2,3,4,5,6 being different outcomes.}$$

An *Event* is a subset of the sample space, with its probability being given by the sum of all of its outcomes.

The *probability* of an event E, $P(E)$ provides information about how likely the considered event is going to happen and must always be between 0 and 1, with 0 representing the chance of an event never occurring and 1 representing the likelihood of an event always occurring.

For example, let's consider a 6-face fair die roll, with each outcome having the same probability of occurring and let's consider the following event "The die has produced an even number". Then, the probability of this event would be:

$P(E) = P(2) + P(4) + P(6)$

$P(E) = 1/6 + 1/6 + 1/6 = 1/2$

2.2. Markov Process

A Markov Process is a stochastic process (i.e., a mathematical object containing a set of random variables [20]) described at any time by a finite amount of states $S1, S2, ..., Sn$

that satisfies the Markov property, as specified in “Efficient Learning Machines: Theories, Concepts, and Applications for Engineers and System Designers" [21].

The *Markov Property* states the following:

“A stochastic process has the Markov property, if the conditional probability distribution of its future states is only dependent on the present state and not on past states. "

Consequently, a Markov process is *memoryless* and needs not to store any past information about past states. Furthermore, such a process is not static: conversely, as it evolves over time depending on its present state, it is said to be a *dynamic system*, which can transition from a state to another one with a certain probability. Such probabilities to pass from a state to another one are represented in a so-called *transition probability matrix*, as shown on the right-hand side of Figure 2.1.

In our case, we make use of a Markov Process to represent usage interactions in tasks, as we assume that only the present task affects the future task and past tasks have no relevance for the future (i.e: a task is independent of tasks that occurred before it). To understand why this property may hold in our setting, we will refer to the graph of Figure 5.5. Let's consider the path formed by tasks S9, S4, S2 and let's suppose the HMM is currently in task S4. It is clear that searching for an article (S2 - future) is not affected by the fact that we checked whether an item is in a Stock (S9 - past) but only if the stock has been changed (S4 - present).

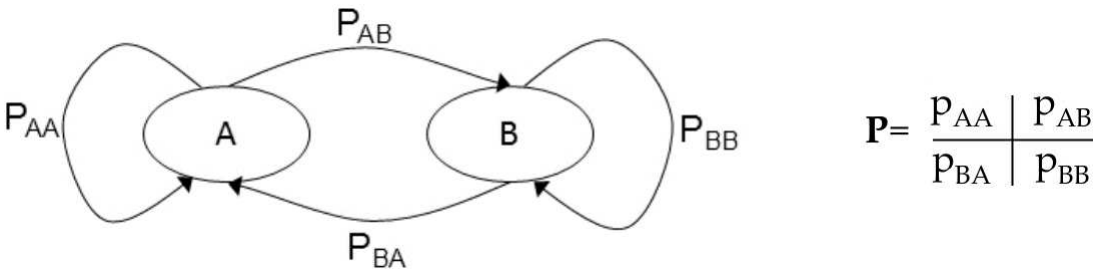


Figure 2.1: Markov Process with 2 States A and B and its corresponding transition probability matrix [22].

2.3. Discrete Markov Chain

A Markov chain is a Markov process discrete in time [23], undergoing transitions from one state to another over discrete time intervals, forming an interlinked chain of states.

Analogously to a Markov Process, it needs to respect the Markov property: future states must not depend on past events, but just on present ones. The term *Markov chain* is also used to indicate a Markov process which has a finite or countable state space.

Given that Markov Chains are probabilistic state machines, the probability of a Sequence or a set of sequences to be observed within a single state must always sum to 1.

Hence, for a state A, given a sequence S of n elements, the total probability is given by the following formula:

$$P(S_A) = \sum_{i=1}^{n} S_i = 1$$

2.4. Hidden Markov Model

A *Markov Model* is a stochastic model used to shape systems undergoing changes over time that respect the Markov property.

As stated in [21], a *Hidden Markov Model* is a probabilistic state machine whose states are not directly observable, but hidden. They model two distinct, but tightly-coupled, stochastic processes: one concerning observable symbols (e.g. users' activities traced by log events) and one the hidden states of the system (e.g., users' tasks). As the states are hidden, the symbols producing observations or sequences of observations cannot be determined precisely. However, the likelihood for a certain state to produce a certain symbol can be computed and is outlined in an *Emission Probability Matrix*. An example of such matrix with respect to the HMM shown in Figure 2.2 is given in Table 2.1. The probability to transition from a state to another one is described in the *State Transition Probability Matrix* and an example of it with respect to the HMM shown in Figure 2.2 is given in Table 2.2.

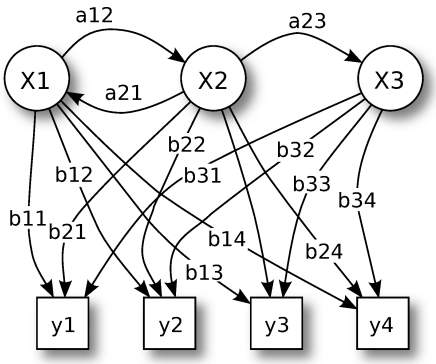


Figure 2.2: Hidden Markov Model with X as states, y as possible observations, b as emission probabilities, a as state transition probabilities [24].

Table 2.1: Table representing the Emission Probability Matrix for Figure 2.2

Emission Probability				
	y1	**y2**	**y3**	**y4**
X1	b11	b12	b13	b14
X2	b21	b22	b23	b24
X3	b31	b32	b33	b44

Table 2.2: Table representing the State Transition Probability Matrix for Figure 2.2

State Transition Probability			
	X1	**X2**	**X3**
X1	-	a12	-
X2	a21	-	a23
X3	-	-	-

2.5. State-of-the-Art Algorithms in HMMs

Three main algorithms are of particular interest in the usage of HMMs [25]:

1. *Evaluation*: Given an HMM model H and a sequence of observations $O = o_1, o_2, .., o_n$, the *Forward* algorithm finds the probability of the sequence of observations O to be generated by the model H. Hence, it finds: $P\{O|H\}$.

2. *Decoding*: Given an HMM model H and a sequence of observations $O = o_1, o_2, .., o_n$, the *Viterbi* algorithm finds the sequence of hidden states with the highest probability to be generated by the model H.

3. *Learning*: Given an HMM model H and a sequence of observations, $O = o_1, o_2, .., o_n$, *the Baum-Welch* algorithm [5] finds optimal parameters T, E, π in $H = (S, V, T, E, \pi)$, where T is the State Transition probability Distribution, E is the Emission (or observation symbols) probability distribution and π describes the initial state probability distribution.

   Optimal parameters for a sequence of observations $O = o_1, o_2, .., o_n$ need to maximize $p\{O|H\}$.

2.6. Applications of HMMs: Unsupervised Machine Learning

HMMs can be used in the process of inferring a function to describe hidden states from unlabeled data. For this reason, they find extensive usage and application in the prediction and modeling of complex events and hence represent an optimal technique for extrapolating *high-level intent* from a set of simple observations [5]. A few examples of their potentiality are hereby presented:

1. *Text translation*: The famous web-based Google Translate tool for translating text sentence-by-sentence from a language to another is an example of the *decoding* problem, as it tries to match a piece of text, (a set of observations) with a sequence of hidden states (its translation). It does so by maximizing the accuracy of the translation and taking into account a huge set of analyzed web pages that have a translation in both languages, combined with expert's feedback on sentences' translations [8].

2. *Human speech recognition*: Different knowledge bases can be used as parameters for *training* an HMM, such as acoustics, language, and syntax to create a unified probabilistic model adept in recognizing human speech [6].

3. *Handwriting recognition*: Similarly to Human Speech Recognition, HMMs can be trained with a wide array of parameters like aspect ratio, number of strokes, tilt, distance of characters, characters size to infer a model that can describe someone's handwriting, as shown by Shu [7].

Chapter 3

Method Design

3.1. Overview

In this section, we present an overview of the method designed for building an HMM from logs and validating the models generated with the process manager. After presenting the main steps comprising the method, non-study-case specific steps are described in detail in this Chapter. Study-case specific steps, i.e: data pre-processing and validation are described respectively in Chapters 4 and 5.

1. *Data pre-processing.* In this step, symbols not relevant for the process are removed and sequences with the same ID are grouped and enumerated.

2. *Iteratively Building HMM* (IHMM). In the approach proposed by Damevski *et al.* [5] for building an HMM from a log dataset iteratively, in every iteration a state is added to the Baum-Welch trained HMM. We re-factored and re-engineered Damevski *et al.*'s method by also building a set of R scripts that we have made available [1].

   We modified the scripts so at so show the top k θ-interesting sequences and build IHMMs in two different ways:

   - *Build Automatically generated, Iterative HMM* (AIHMM). The HMM is built iteratively by automatically picking symbols contained in the top k θ-interesting sequences.
   - *Build the Interactive - Iterative HMM* (IIHMM). Analogously to [5], the HMM is built iteratively by presenting the manager with the top k θ-interesting sequences. The manager hand-picks symbols to be compounded in a task.

   We repeated AIHMM and IIHMM construction for k = 1,2,3,4,5.

3. *Validate AIHMMs with best-likelihood and IIHMM with the manager.* The manager is presented the AIHMMs characterized by best quality (model quality is the log-likelihood of the model, i.e: the natural logarithm of the HMM's likelihood to capture the underlying data) and these are compared with the built IIHMM as well as the DISCO model.

[1] goo.gl/YCV9z6

3.2. Iteratively Building an HMM (IHMM)

Referring to the definition of HMMs, Symbols and States outlined in Section 2.4, in our setting *Symbols* are unique log events triggered by the users (e.g., `Order: Search`), whereas *States* are the tasks users want to accomplish with the system (e.g., `Order a product`), and *Sequences* are sequences of log events as defined in Section 4.4.

Following the approach of [5], we build an HMM iteratively by adding one state at each iteration, starting from the state 'Start', which contains all symbols in our sequences (sink state).

In each iteration, the model is trained with the Baum-Welch algorithm (Section 3.2.1) to find optimal probabilities in the transition and emission probability matrices. Afterwards, a list of θ*-interesting sequences* is generated.

Interesting Sequences: An interesting sequence is the longest sequence of log events that are not yet modelled by HMM, but it does appear in the data. Interesting sequences in the list are ranked by their interestingness score ($I^{\mathrm{DM}}(O|\mathbb{O}), H$) on a Hidden Markov Model H and are computed on the union of θ-frequent sequences ($\mathbb{O}^D(\theta,\mathbb{O})$) and θ-probable sequences ($\mathbb{O}^M(\theta,H)$), through the re-engineered versions of Damevski *et al.* [5]'s algorithms (respectively outlined in sections 3.2.2 and 3.2.3).

The interestingness score measures the difference of frequency of the sequence in the data ($F^{\mathrm{D}}(O,\mathbb{O})$) and the probability to be modelled by the HMM ($P\{O|H\}$) and is shown in Fig. 3.1.

$$I^{\mathrm{DM}}(O|\mathbb{O},H) = \begin{cases} (a): F^{\mathrm{D}}(O,\mathbb{O}) - P\{O|H\}, O \in \mathbb{O}^D(\theta,\mathbb{O}) - \mathbb{O}^M(\theta,H) \\ (b): F^{\mathrm{D}}(O,\mathbb{O}) - P\{O|H\} O \in \mathbb{O}^D(\theta,\mathbb{O}) \cap \mathbb{O}^M(\theta,H) \end{cases}$$

Figure 3.1: Formula for computing the interestingness of a sequence [5].

For (a), $P\{O|H\}$ is available, as it is already computed in Algorithm 2. For (b), $P\{O|H\}$ is computed with the *forward* part of the forward-backward procedure for HMMs, as described by [26].

Both frequency and probability need to be greater than a given value θ. In this work, we set the value equal to $\frac{2}{\mathbb{O}}$ (where $\mathbb{O}$ is the total amount of sequences), as we want to have at least two instances of a sequence before considering it interesting.

Hence for every interesting sequence O, we check whether $I^{\mathrm{DM}}(O|\mathbb{O},H) \geq \theta$. The way we compute the frequency of the longest sub-sequences with frequency greater than θ is illustrated in Algorithm 1, which is a re-engineered version of the original algorithm in [5]. In particular, the algorithm computes the longest sequence as longest prefix (i.e., a subsequence that starts at the beginning of the sequence to which it belongs) between sequences.

Symbols Selection: To add a new state, the symbols of the interesting sequences generated are selected. How to select such symbols depends on the two different procedures described in Section 3.4 and Section 3.3 respectively. Then, a new state is added and the selected symbols are moved from the "ALL DATA" state to the new one. To move such symbols, the emission matrix needs to be updated to include the probabilities of the selected symbols in the new state and set to 0 the probabilities of the same symbols removed from the 'ALL DATA" state. The resulting matrix is displayed in Fig. 3.2. The iterative process is repeated until no new θ-interesting sequences are generated.

$$E_{ij}^{n+1} = \begin{cases} \begin{cases} 0 & \text{if } j \in \text{MS} \\ E_{1j}^n / \sum_{j \notin \text{MS}} E_{1j}^n & \text{otherwise} \end{cases} & \text{for } i = 1 \\ E_{ij}^n & \text{for } 1 < i < n+1 \\ \begin{cases} r_j & \text{if } j \in \text{MS} \\ 0 & \text{otherwise} \end{cases} & \text{for } n+1 = 1 \end{cases}$$

Figure 3.2: The emission matrix updated at iteration $n+1$. r_j are uniformly random numbers for which $\sum_{j \in \text{MS}} r_j = 1$

Quality of the generated HMM at each iteration: The iterative process continues and a new iteration starts until the quality of the generated HMM degrades. To evaluate the quality of the generated HMM, we perform two checks at the end of each iteration:

- The log likelihood of HMM at iteration i is greater than or equal to the log likelihood of the HMM at iteration $i-1$.
- The log likelihood of the HMM at iteration $i1$ is greater than or equal to the log likelihood to a baseline HMM with the same amount of states of the HMM at iteration $i-1$. For iteration 1, we consider the log-likelihood of the HMM model before symbols are constrained onto the new state.

As baseline HMM, we augmented the HMM by adding a new state in which symbols have equal probability to be emitted (uniform probability) and transitions to that state have equal probability as well (uniform probability). When the aforementioned checks pass, then a new iteration begins. Otherwise, the process stops and reports the HMM generated so far, hence at iteration i.

Symbols selection: Firstly, we generate them automatically (Section 3.4), then we built IIHMMs interactively with the process manager (Section 3.3). In both cases, we repeated the process for k = 1, 2, 3, 4, 5.

3.2.1. Baum Welch Algorithm

To train an HMM with the 'Baum Welch' algorithm for finding optimal parameters in the transition and emission matrices, we made use of the "baumWelch" function from the package 'HMM' [27]. However, when training an HMM with a single state, we encountered the following error:

"Error in f[i, length(observation)] : subscript out of bounds."

Consequently, we inspected the code for the "baumWelch" function and identified the problem in the auxiliary function "baumWelchRecursion". In particular, we realized that the following loop: "for (i in 2:length(hmm$States))" started at index 2 without prior checking if the input HMM had 2 or more states: when looping through an HMM with a single state, this gives rise to the aforementioned error.

We fixed the problem by implementing a check for the number of states to be at least 2. Then, we contacted the package maintainer and submitted the fixed code. We obtained the following answer:

Mon 10/10/2016, 09:01

Hi Daniele

thank you for using my package!

I thought that a HMM-model with one hidden state doesn't make sense, since the underlying transition matrix is trivial and you can infer the observation parameters directly (here Baum Welch is not efficient). But I see, it might be useful for other reasons...

Thank you very much for your code, I will fix this in the next release. As you mentioned, a simple check on the number of hidden states is sufficient.

3.2.2. Theta-Frequent Sequences - Algorithm 1

Input: Observation Sequences $\mathbb{O}$;frequency threshold θ where $\frac{2}{|\mathbb{O}|} = \frac{2}{K} \leq \theta \ll 1$

Output: longest θ-frequent sequences $\mathbb{O}^D(\theta, \mathbb{O})$

1 sort $\mathbb{O}$ in lexicographical order into list
2 L{$\mathbb{O}$} = $(O^{(1)}, ..., O^{(K)})$
3 // T^p, T^c: previous and current prefix length
4 $T^{(p)} = 0$
5 **for** $i \leftarrow 1$ *to K - 1* **do**
6 $T^{(c)} \leftarrow 0$
7 j $\leftarrow 1$
8 **while** $O_j^{(i)} = O_j^{(i+1)} \wedge j \leq min(Len(O^{(i)}), Len(O^{(i+1)}))$ **do**
9 $T^{(c)} \leftarrow T^{(c)} + 1$
10 $j \leftarrow j + 1$
11 **end**
12 **if** $T^{(p)} \neq T^{(c)}$ **then**
13 $\hat{O} \leftarrow$ Prefix $(O^{(i)}, T^{(c)})$
14 f $\leftarrow$ ComputePrefixFrequency $(\mathbf{L}(\mathbb{O}), i, \hat{O}, T^{(c)})$
15 SaveSeqAndFreq$(\mathbb{O}^D(\theta, \mathbb{O}), \hat{O}, f)$
16 $T^{(p)} \leftarrow T^{(c)}$
17 **end**
18 **end**
19 **Function** *ComputePrefixFrequency (**L**($\mathbb{O}$), i, $\hat{O}$, T)*
20 $n \leftarrow 1, j \leftarrow$ i -1
21 **while** $j > 0 \wedge$ *Prefix*$(O^j, T) = \hat{O}$ **do**
22 $n \leftarrow n + 1, j \leftarrow j - 1$
23 **end**
24 $j \leftarrow i + 1$
25 **while** $j \leq K \wedge$ *Prefix*$(O^j, T) = \hat{O}$ **do**
26 $n \leftarrow n + 1, j \leftarrow j + 1$
27 **end**
28 **return** $\frac{n}{K}$

Algorithm 1: Reviewed algorithm to generate the longest θ-frequent sequences $\mathbb{O}^D(\theta, \mathbb{O})$ where $\theta \geq \frac{2}{K}$

3.2.3. Theta-Probable Sequences - Algorithm 2

Input: HMM where $\mathbb{S}$ are states, $\mathbb{V}$ are the symbols, **T** is the Transition Matrix, **E** is the Emission Matrix, π is the initial states probability.

HMM $\lambda = (\mathbb{S}, \mathbb{V}, \mathbf{T}, \mathbf{E}, \pi)$; $|\mathbb{S}| = N$; $|\mathbb{V}| = M$; $\pi^0 = \frac{1}{N}\left\{\pi_1^{(0)}\pi_2^{(0)}...\pi_N^{(0)}\right\}$

Output: θ-probable sequences $\mathbb{O}^M(\theta, \lambda)$

1 //ϵ: $\mathbb{V}^0$, the empty sequence
2 **Function** *GenerateHMMSequences (O, $\alpha(O)$)*
3 p $\leftarrow \sum_{i=1}^{N} \alpha(O, i)$
4 **if** $p \geq \theta$ **then**
5 SaveSeqAndProb($\mathbb{O}^M(\theta, \lambda), O, p$)
6 **foreach** $o \in \mathbb{V}$ **do**
7 **for** $j \leftarrow 1$ *to* M **do**
8 $O' \leftarrow O_o$
9 $\alpha(O', j) \leftarrow \left[\sum_{i=i}^{N} \alpha(O, i)\, T_{ij}\right] E_j(o)$
10 **end**
11 GenerateHMMSequences(O', $\alpha(O')$)
12 **end**
13 **end**

Algorithm 2: Reviewed algorithm to generate the longest θ-probable sequences $\mathbb{O}^M(\theta, \lambda)$ from HMM $\lambda = (\mathbb{S}, \mathbb{V}, \mathbf{T}, \mathbf{E}, \pi)$, where $\theta \geq \frac{2}{K}$

3.3. Build an Interactively generated Iterative HMM (IIHMM)

At each iteration, we presented a set of top k θ-interesting sequences to the manager, who identified the symbols to add in the new state of the model and gives the label to the states generated by the model at each iteration.

The flowchart in Figure 3.3 displays the full IIHMM construction process.

3.4. Build an Automatically generated Iterative HMM (AIHMM)

In each iteration, to automatically select the symbols for the next iteration, we consider the top k θ-interesting sequences generated by the HMM at iteration i. All the symbols of the interesting sequences that are not contained in existing states of the current HMM have been selected to be included in the next iteration. We iterated this procedure until the quality of the generated model degrades as described in Section 3.2.

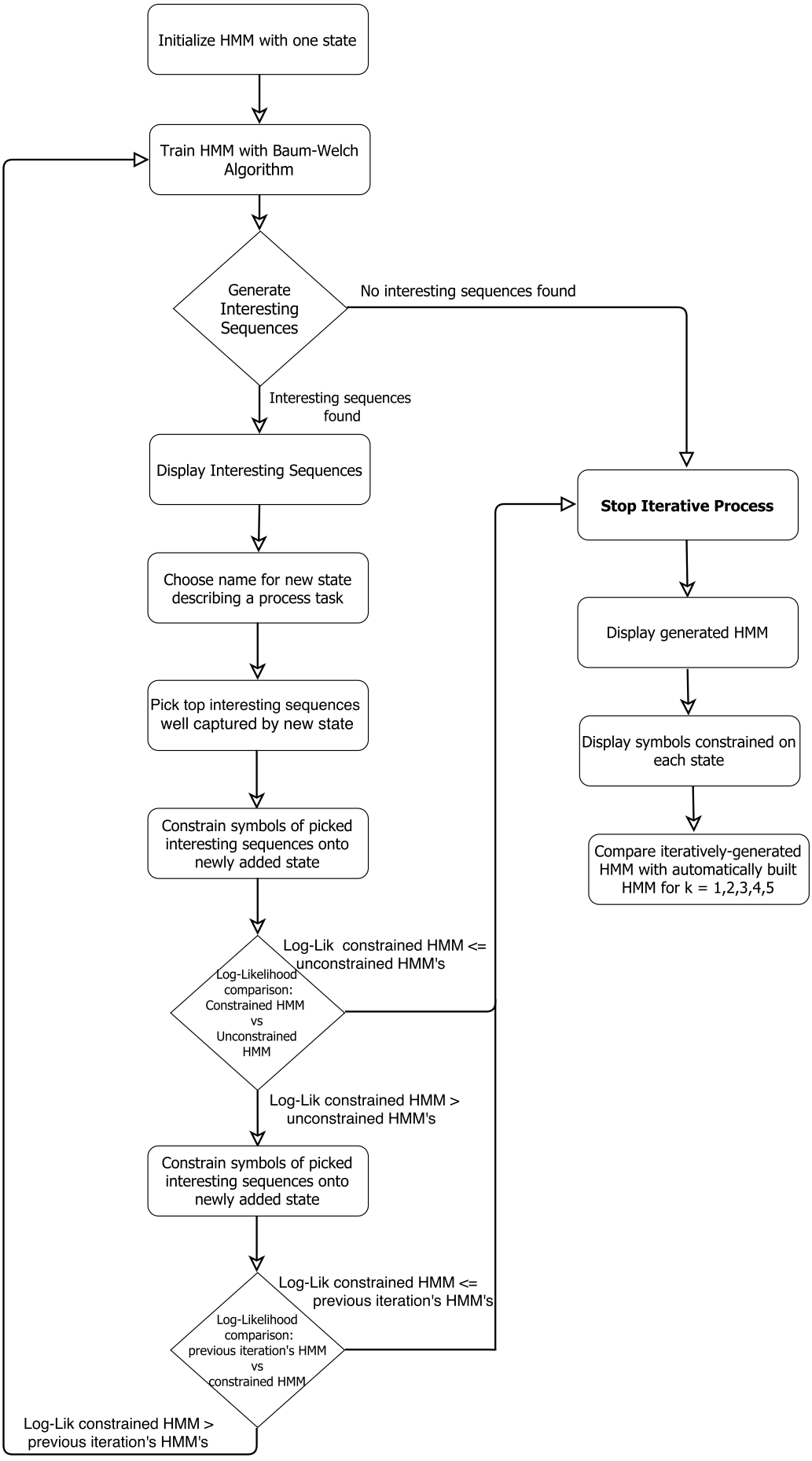


Figure 3.3: Flowchart describing the IIHMM construction process.

Chapter 4

Evaluation

In the present chapter, we describe the experiment conducted on an industrial case study to validate the method described in Chapter 3.

4.1. Case Study

To collect the data used in this paper, we instrumented a software system in use at an industrial company located in South Tyrol, Italy. The company has 11 collaborators and manufactures complex precision turned parts, milled parts, and technical products. The software we mined is an ERP system developed in-house and is available for the users as a standalone or mobile version. In this work, we will apply our approach to the mobile version as its functionalities result simpler and easier to interpret.

To be competitive on the market, the small company continuously invests in quality (e.g., 100% of the produced parts are quality checked) and allows customers to produce custom-designed parts in free machining steel, plastic, aluminum, copper, and others. The company uses CNC short- and long turning machines for rotary parts made of various materials in complete machining. All CNC machines have driven tools, C-axes, Y-axes and counter-spindles. It produces parts that vary from diameters from 1mm up to 40mm off the shelf. All post-machining operations such as rolling, slitting, honing, grinding, brushing, thread cutting and molding are also carried out by the company under study.

From a bird's eye view, the monitored software is an in-house programmed Enterprise Resource Planning (ERP) system, currently developed and maintained by two programmers. The ERP system is released as stand alone or mobile. The mobile version has fewer features so that users perform simpler tasks. To develop the mobile application, the company uses Xamarin[1], a set of tools that allow for the usage of C# to write Android, iOS, and Windows apps with native user interfaces and share code across multiple platforms. The mobile ERP system is available both at Store[2] (for the Windows Phone application) and Google Play Store[3] (for the Android application). In this work, we will focus on the logs generated by the mobile ERP system as they are potentially simpler to interpret.

[1] https://www.xamarin.com/

[2] https://www.microsoft.com/en-US/store/apps/windows-phone

[3] https://play.google.com/store?hl=en

4.2. Use Logs Collection

To create an event log, we extended the ERP software so that it collects the user behavior and generates an event log. With “user behavior”, we mean that the user enters or leaves certain parts of the software. To trace this form of behavior, various approaches are possible. One approach is to add the logging code manually wherever it is interesting for the analysis to trace the user entering that section [18]. Approaches which have smaller impact on the overall source code of the application but are less flexible in selecting which events to trace include: to use inheritance or aspect oriented programming to add the logging code to classes of objects that need to be traced; monitor user interface changes on the operating system level; or use language-specific features to monitor the user interface.

In this work, we follow a different approach that exploits few Microsoft-based technologies and uses language-specific features to monitor the user interface. The software was written in C#[4], based on the Microsoft .NET Framework 4.5[5] and developed with Microsoft Visual Studio 2013[6]. The user interface was developed using Windows Presentation Foundation[7] (WPF), which is a framework to build user interfaces on Microsoft Windows Platforms. Within WPF, to collect the needed data, we are able to attach an event handler to the base class of the control we want to monitor, e.g., to the classes “Button” or “Window”, which is then called whenever an event occurs within one of the child classes. This approach allowed us to develop the logging behavior at one point of the application, without affecting the code of the application itself.

4.3. Collected Data

The data was collected in a time-frame ranging from 18th February 2015 until 4th April 2016. In Table 4.1, we describe a typical interaction with the program to illustrate the data collected. The monitored program is an internally developed Enterprise Resource Planning (ERP) application in which customers, orders, production plans, and production steps can be stored and executed.

In our example, the user starts the program and wants to create a new order entry with an associated contact. The user clicks on “Add new order", which opens the form to create a new order.

The logger will now generate a Globally Unique Identifier (GUID) for this event, (e.g., “fec1ce05-a197-456a-9d8e-d303ac2a0319") and stores the timestamp, the currently logged user, and the description of the event in the database [18].

As the user continues interacting with the software, the logging component traces his or her activities. An example of the collected data is shown in Table 4.1.

The Timestamp is the time in nano seconds as collected by the system clock elapsed since 12:00:00 midnight January 1, 0001 (e.g., 1420070400 is January 1st, 2015 at midnight) at which the log event has been triggered. The EventID is the GUID of the log event. The Resource reports the user that triggered the event and the Description illustrates the user’s action to the system.

[4] https://docs.microsoft.com/en-us/dotnet/csharp/programming-guide/index
[5] https://www.microsoft.com/it-it/download/details.aspx?id=30653
[6] https://msdn.microsoft.com/en-us/library/dd831853(v=vs.120).aspx
[7] https://docs.microsoft.com/en-us/dotnet/framework/wpf/

Table 4.1: Example entries of the generated event log. [18]

Timestamp	EventID	Resource	Description
1420070400	fec1ce05...	User 1	Start Order
1420071000	2d4126d0...	User 1	Search Customer
1420071600	b51dcc60...	User 1	Select Customer
1420072200	75273fad...	User 1	Start select order date
1420072800	965ef980...	User 1	Select Date
1420073400	d5d681c6...	User 1	End Select order date
1420074000	8606bfcd...	User 1	Refresh Order
1420074600	251389cd...	User 1	End Order

Table 4.2: Details about the data log in use

# logs	users	timeframe	size
148107	11	18/2/2015 - 4/4/2016	20,1MB

4.4. Data Pre-processing

In this step, logs not relevant to the process have been removed and sequences have been formed and enumerated.

We collected 148107 event logs, which are comprised of 60 symbols. The histogram in Figure 4.1 draws the symbols' frequencies, whereas the boxplot in Figure 4.2 draws the symbols' distribution, highlighting outliers.

To define the sequences of logs to be used in the HMM, we analysed whether all symbols are relevant to our analysis.

Figure 4.1: Bar chart describing symbols' frequency.

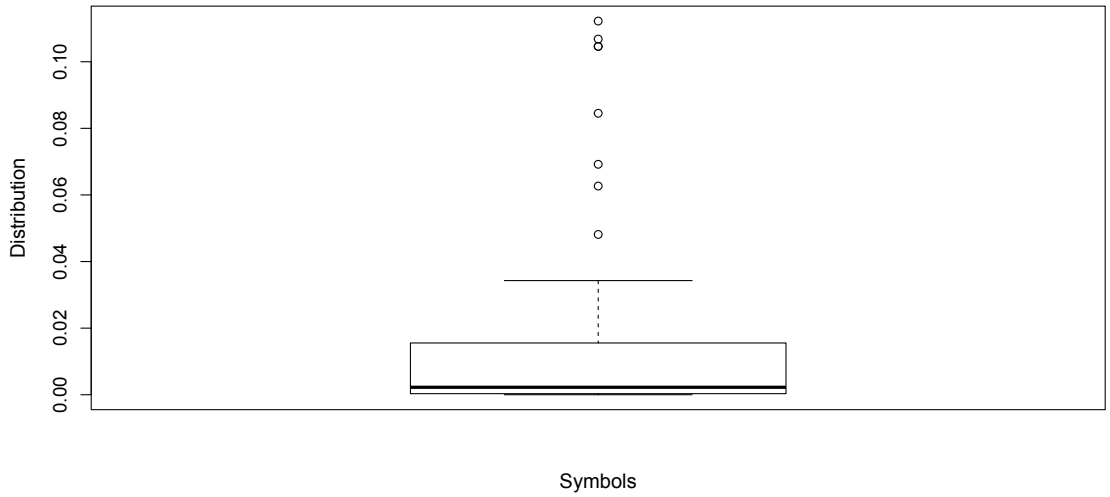


Figure 4.2: Boxplot describing symbols' distribution.

1) Filter out extremely rare/frequent symbols that are not relevant to the process or redundant. We first extracted the list of outliers from the distribution of symbols frequencies. Then we augmented such list with symbols triggered by single users. The thesis' author, his supervisor and the co-supervisor independently reviewed the final set of symbols with the goal of excluding those with little relevance for the process. We proceeded in three steps:

- A list of candidate symbols has been identified, shown in Table 4.3. Firstly, we extracted the list of outliers from the distribution of symbols frequencies. Then we augmented such list with symbols triggered by single users.
- The thesis' author, his supervisor and the co-supervisor individually reviewed all symbols to exclude those with little relevance for the process and cast a vote to include or exclude the symbol from the analysis Votes cast are shown in Table 4.3.
- All symbols that received a negative agreement among the authors were removed. The symbols "Start" and "End" are *built-it*, as they are added automatically by the logging system respectively upon beginning and conclusion of an activity. Hence, they provide no information about the activity being carried out and were removed. The symbols "Navigate to hubpage" and "Navigate back to hubpage" correspond to the user's intention of returning to the ERP system's main menu and, analogously to symbols "Start" and "End", also provide no information about the task being performed.

2) Define sequences. Sequences have been formed as ordered sets of logs starting with the "Start" and ending with the "End" symbol. Sequences have then been enumerated. After pre-processing the data, the dataset consisted of 14343 sequences and 56 unique symbols. On average, sequences contain 6.55 symbols, of which 2.87 are unique.

4.5. Experimental Validation

In order to validate the approach laid out in Chapter 3, we conduct an experiment to compare the *AIHMM*, *IIHMM* and the *DISCO model* together with the process manager (from here onwards, company stakeholder: CS). Hence, We apply the method described in

Table 4.3: Rare and most frequent symbols outlined by the boxplot or symbols triggered by single users and the decision to filter

Symbol	vote	note
`End`	remove	built-in
`Navigate to articledetailview`	keep	
`Order: Search`	keep	
`Start`	remove	built-in
`Navigate back to hubpage`	remove	always before `End`
`Navigate to hubpage`	remove	default to perform another operation
`Navigate to orderlistview`	keep	
`Navigate to ordersearcherview`	keep	

Section 3.2 for generation of AIHMM (as outlined in Section 3.4) and IIHMM (Section 3.3). In both cases, we make use of the following $k = 1, 2, 3, 4, 5$.

Building Interactively generated IHMM (IIHMM): To perform activities described in Section 3.3, we defined a protocol of interaction with the process manager (from now onwards, company's stakeholder) as shown in Table 4.4. Overall, interaction with the manager lasted two hours.

4.5.1. View Selection

After being presented with the Small View (Figure 4.3), the Medium View (Figure 4.4), the Large View (Figure 4.5) and the Largest View (Figure 4.6) as defined in Step 1 of the protocol in Table 4.4 , the CS selects the *Large View* from Figure 4.5.

DE to CS: "Why did you choose this process model? Which elements are inside that give you the impression that this is a good match? Or which tasks are not well captured in the other models?"

CS: "The selected model shows more or less the steps a "normal" user has to perform on a daily basis. Instead, the largest view shows the steps for some of the users that perform advanced tasks". A possible user target for Largest View is an intern. The small and medium views are too generic - there are essential actions missing like "Warenausgang". Also, in the medium and small view, you cannot see how the user navigates around the application, and hence it does not show the process used in the company".

Table 4.4: Interaction protocol with the Company stakeholder (CS)

Attendees:	Company's stakeholder (CS), Moderator (M), Domain Expert (DE), Operator (O)
System:	Mobile System
Output:	IIHMM
Step 0:	M explains CS about goal of the activity
Step 1:	The CS is presented with the following views: Small View (Figure 4.3), Medium View (Figure 4.4), Large View (Figure 4.5) and Largest View (Figure 4.6) generated with DISCO with parameter "Activities" respectively at 5%, 15%, 25%, 35% and parameter "paths" at 0%, based on the experiment conducted in [18].
Step 2:	The O runs the IIHMM process (as in Section 3.2) with k=1;
Step 3:	The CS is presented with k θ-interesting sequences and is asked to select the symbols that can be best clustered into a task;
Step 3.1:	The CS decides a task name for the symbols selected;
Step 3.2:	The CS discuss the resulting process in comparison with the DISCO model the stakeholder selected in Step 1;
Step 4:	Steps 3.1 and 3.2 are replicated for each iteration and different values for k = 2,3,4,5.
Step 5:	Final discussion and selection of the most representative IHMMs.

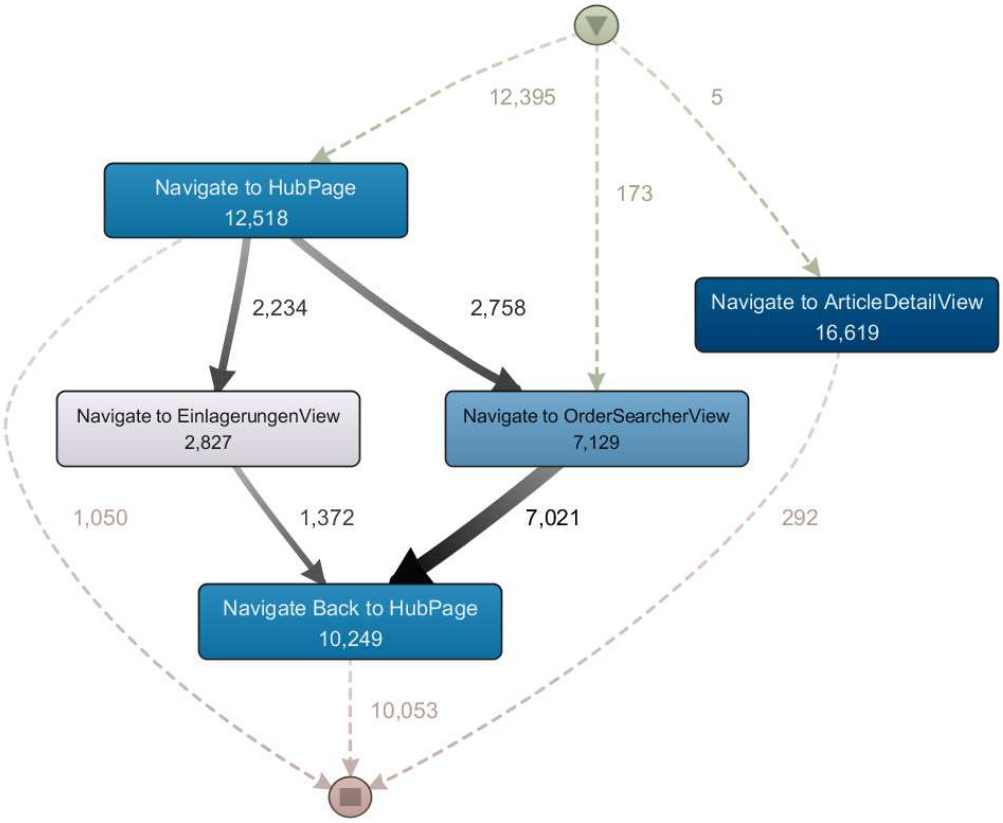


Figure 4.3: Small view of the BP showing 5% of all activities.

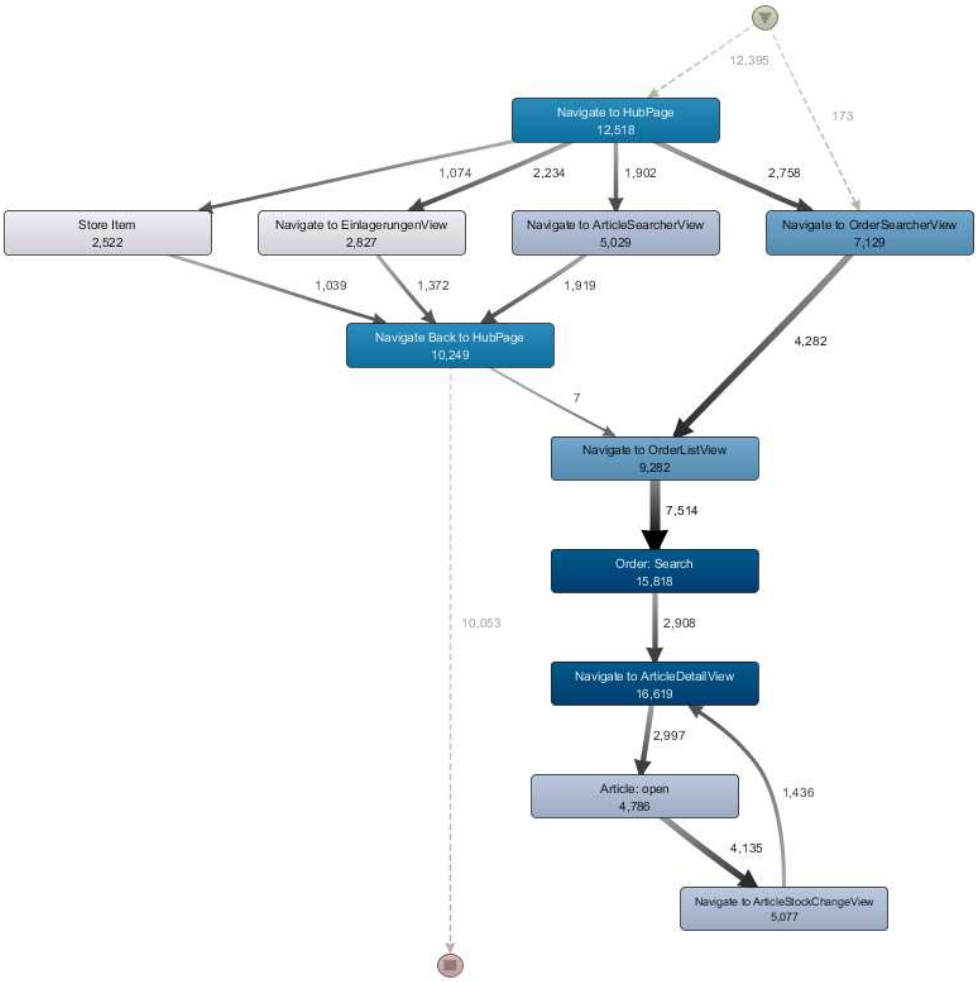


Figure 4.4: Medium view of the BP showing 15% of all activities.

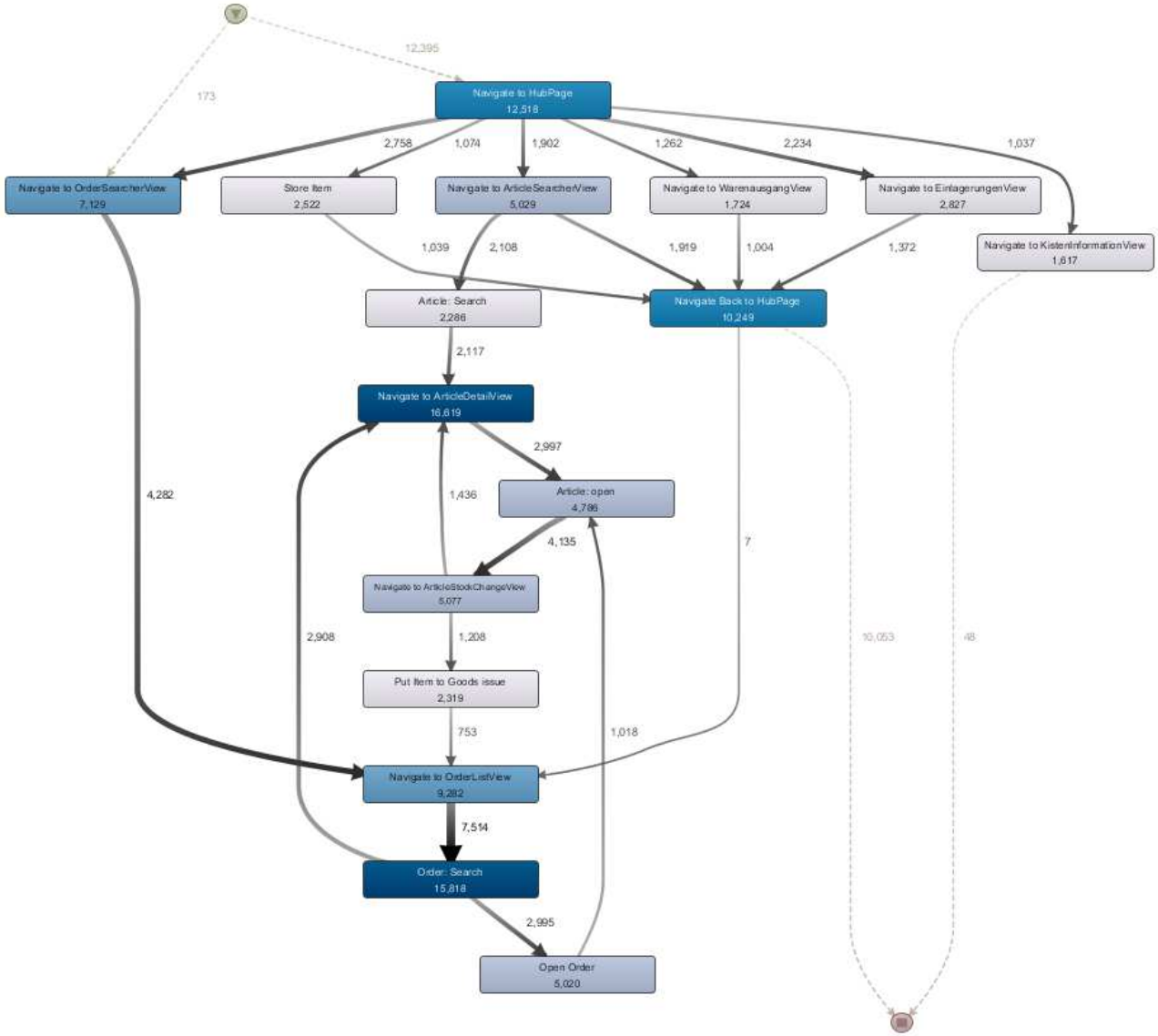


Figure 4.5: Large view of the BP at 25% of all activities.

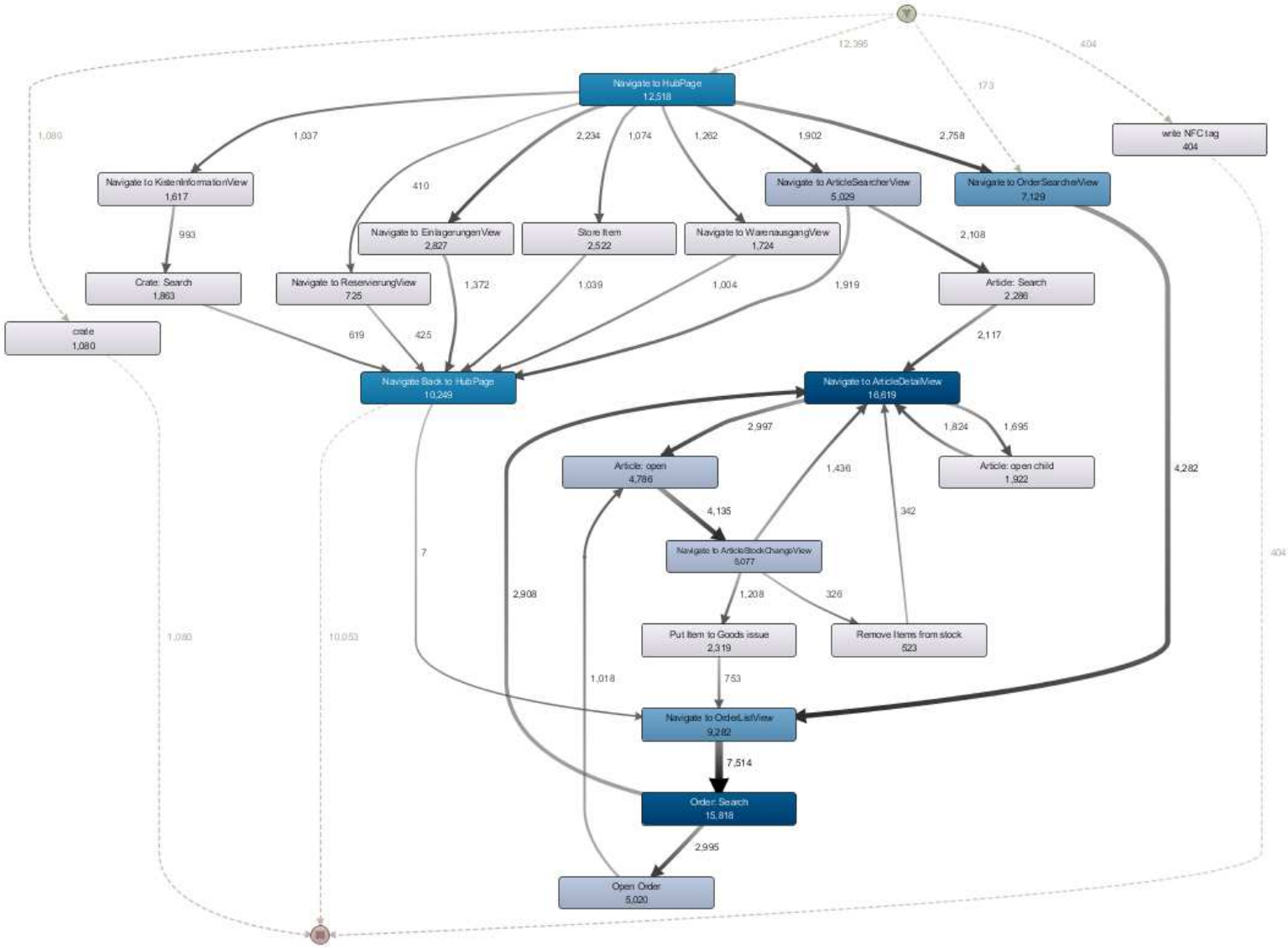


Figure 4.6: Largest view of the BP showing 35% of all activities.

4.5.2. IIHMM Construction

In this section, we outline the IIHMM construction process described in Section 3.3. We carry out IIHMM construction for k = 1,2,3,4,5. Tables 4.5, 4.6, 4.7, 4.8, 4.9 and 4.10 show the result of applying the IIHMM construction process on one-year long dataset. Following, the content of the aforementioned tables is explained.

Firstly, the CS is presented with top k θ-*interesting sequences* generated from an HMM trained with the Baum-Welch Algorithm and with amount of states equal to the Iteration index (I). The CS selects symbols to group into a task (*Selected Symbols*) and assigns a *Task Name* to them.

The log-likelihood of the newly constrained HMM is compared with the log-likelihood of the HMM from the previous iteration ($HMM_i > HMM_{i-1}$). In case of I = 1, the log-likelihood of the HMM from the previous iteration is the one extracted from the HMM after it is trained with the Baum-Welch and no symbols have been constrained yet.

The log-likelihood of the newly constrained HMM (*HMMC*) is compared with an unconstrained HMM (*HMMU*) with the same amount of states (*log-lik HMMC > HMMU)*. The *HMMU* is built iteratively by adding one state per iteration and training the resulting HMM in every iteration with the Baum-Welch algorithm, without constraining symbols on the newly added state. If both mentioned log-likelihood checks for increase are satisfied, a new iteration is triggered, else the IIHMM generation process stops for the current k and the current HMM is displayed. Alternatively, different symbols can be selected and the iteration repeated.

IIHMM with k = 1

Table 4.5: Table describing the unwinding of IIHMM construction for k = 1

IIHMM with k = 1					
I	**Interesting Sequences**	**Selected Symbols**	**Task Name**	**log-lik HMM_i > HMM_{i-1}**	**log-lik HMMC > HMMU**
1	(1)"Store Item" "Navigate to OrderSearcher-View" "Order: Search" "Order: Search" "Navigate to OrderListView" Score =0.000133858392852624	"Store Item" "Order: Search"	Add Article to Stock	-270327.9 > -15636971	-270327.9 $\not>$ -265354.4

For k = 1, IIHMM construction is halted in Iteration 1, as the log-likelihood of the constrained HMM does not surpass the log-likelihood of the unconstrained HMM.

IIHMM with k = 2

Table 4.6: Table describing the unwinding of IIHMM construction for k = 2

IIHMM with k = 2					
Iteration	**Interesting Sequences**	**Selected Symbols**	**Task Name**	**log-lik HMM_i > HMM_{i-1}**	**log-lik HMMC > HMMU**
1 (G 1)	(1)"Store Item" "Navigate to OrderSearcherView" "Order: Search" "Order: Search" "Navigate to OrderListView" Score =0.000133858392852624 (2)"Article: Search" "Article: Search" "Navigate to ArticleDetailView" "Navigate to ArticleSearcherView" Score =0.000133931191838847	"Order: Search" "Article: Search" "Navigate to ArticleDetailView"	Display Article Details of an Order	-267127.2 > -15636971	-267127.2 $\not>$ -265354.4
1 (G2)	(1)"Store Item" "Navigate to OrderSearcherView" "Order: Search" "Order: Search" "Navigate to OrderListView" Score =0.000133858392852624 (2)"Article: Search""Article: Search" "Navigate to ArticleDetailView" "Navigate to ArticleSearcherView" Score =0.000133931191838847	"Article: Search" "Store Item" "Order: Search"	Add Item to Stock	-269581.1 > -15636971	-269581.1 $\not>$ -265354.4

For k = 2 and in both selected symbols groups, IIHMM construction is halted in Iteration 1, as the log-likelihood of the constrained HMM does not surpass the log-likelihood of the unconstrained HMM.

IIHMM with k = 3

Table 4.7: Table describing the first two iterations in the unwinding of IIHMM construction for k = 3

IIHMM with k = 3					
I	**Interesting Sequences**	**Selected Symbols**	**Task Name**	**log-lik HMM_i > HMM_{i-1}**	**log-lik HMMC > HMMU**
1 (G 1)	(1)"Store Item" "Navigate to OrderSearcher View" "Order: Search" "Order: Search" "Navigate to OrderListView" Score =0.000133858392852624 (2)"Article: Search" "Article: Search" "Navigate to ArticleDetail-View" "Navigate to ArticleSearcherView" Score =0.000133931191838847 (3) "Crate: Search" "Navigate to OrderSearcherView" "Order: Search" "Navigate to OrderListView" "Order: Search" Score =0.000133934757208659	"Article: Search" "Store Item" "Order: Search"	Add Item to Stock	-269581.1 > -15636971	-269581.1 $\not>$ -265354.4
1 (G 2)	Same as cell above	"Order: Search" "Store Item" "Clear goods issue" "Navigate to Order ListView"	Deliver Items	-262484.1 > -15636971	-262484.1 > -265354.4
2	(1) "RawMaterial: Search" "Navigate to Warenausgang-View" Score =,0.00012996775391172 (2) "Select Kolli from List" "Select Kolli from List" Score = 0.000130316720191263 (3)"Crate: Search" "Navigate to OrderSearcherView" "Order: Search" "Navigate to OrderListView" "Order: Search" Score =0.000131721230683609	"Select Kolli from List" "Navigate to Warenaus-gang View" "Crate: Search"	Modify kolli for delivery	-261060.6 > -262484.1	-261060.6 > -265354.4

Table 4.8: Table describing the third and fourth iterations in the unwinding of IIHMM construction for k = 3

IIHMM with k = 3					
I	**Interesting Sequences**	**Selected Symbols**	**Task Name**	**log-lik HMM_i > HMM_{i-1}**	**log-lik HMMC > HMMU**
3	(1)"Navigate to OrderListView" "Order: Search" "Order: Search" "Show" Score =0.000129448551943325 (2) "Article: Search" "Navigate to ArticleDetailView" "Navigate to ArticleStock-ChangeView" "Article: open" Score =0.000129757717948898 (3)"Store Item" "Navigate to NFCWriterView" Score =0.000130299759265652	"Article: Search" "Article: open" "Navigate to ArticleStock ChangeView" "Store Item"	Change location of article in stock	-254081.1 > -261060.6	-254081.1 > -265354.4
4	Same as in iteration 3	-	-	-	-

In Iteration 4, the very same interesting sequences as in iteration 3 are generated. Hence, because no new interesting sequences are generated, the IIHMM construction stops for k = 3. The IIHMM construction process continues with the HMM constructed so far for k = 4.

IIHMM with k = 4

Table 4.9: Table describing the unwinding of IIHMM construction for k = 4, starting from iteration 4

IIHMM with k = 4					
I	**Interesting Sequences**	**Selected Symbols**	**Task Name**	**log-lik HMM $_i$ > HMM$_{i-1}$**	**log-lik HMMC > HMMU**
4	(1) "Navigate to OrderListView" "Order: Search" "Order: Search" "Show" Score =0.000129448551943325 (2)"Article: Search" "Navigate to ArticleDetailView" "Navigate to ArticleStockChangeView" "Article: open" Score =0.000129757717948898 (3)"Store Item" "Navigate to NFCWriterView" Score =0.000130299759265652 (4) "Store Item" "Navigate to OrderSearcherView" "Order: Search" "Order: Search" "Navigate to OrderListView" Score =0.000130444318250473"	None	-	-	-

The HMM construction resumed at the last iteration where the process stopped with k = 3 (hence, being the process in iteration 4, the HMM currently has 4 states). For k = 4, the CS did not identify any additional tasks that can be captured by the symbols present in the interesting sequences generated.

IIHMM with k = 5

Table 4.10: Table describing the unwinding of IIHMM construction for k = 5, starting from iteration 4

IIHMM with k = 5					
I	**Interesting Sequences**	**Selected Symbols**	**Task Name**	**log-lik HMM > HMM**	**log-lik HMMC > HMMU**
4	(1)"Navigate to OrderListView" "Order: Search" "Order: Search" "Show" Score =0.000129448551943325 (2) "Article: Search" "Navigate to ArticleDetailView" "Navigate to ArticleStock ChangeView" "Article: open" Score =0.000129757717948898 (3)"Store Item" "Navigate to NFCWriterView" Score =0.000130299759265652 (4)"Store Item" "Navigate to OrderSearcherView" "Order: Search" "Order: Search" "Navigate to OrderListView" Score =0.000130444318250473 (5)"Navigate to Kisten InformationView" "Crate: Search" "Navigate to OrderSearcherView"	"Crate: Search"	List Content of crate	-255234.8 $\not>$ -254081.1	-255234.8 > -265354.4

The log-likelihood of the HMM constrained with symbols detected at iteration 4 and trained with the Baum-Welch algorithm did not surpass the quality of the HMM buit in Iteration I_{i-1}. This signifies that the model quality has degraded and consequently, the HMM construction process stops.

Resulting IIHMM

The resulting IIHMM has four states and, before the model quality degrades, the log-likelihood is -254051. The states in the final HMM are the following ones, each one of them corresponding to an iteration.

1. All Data
2. Deliver Items
3. Modify kolli for delivery
4. Change location of article in stock

The "All Data" state is the initial state present in the HMM upon initialization and initially contains all symbols.

The following symbols were constrained onto the different states of the HMM through the iterations corresponding to the state number.

- **State 1**
    1. Navigate to ArticleSearcherView
    2. Navigate to ArticleDetailView
    3. Navigate to EinlagerungenView
    4. Navigate to OrderSearcherView
    5. Open Order
    6. Put Item to Goods issue
    7. Navigate to KistenInformationView
    8. crate
    9. Article: Open drawing
    10. Article: open child
    11. Reserve Item for mounting
    12. Article: mount
    13. Article: check in
    14. Navigate to ReservierungView
    15. open montagereservierung
    16. Add Items to stock
    17. Remove Items from stock
    18. Swap crate
    19. location
    20. Navigate to NfcWriterView
    21. Write NFC
    22. write NFC tag
    23. open order list

24. Add secondary tile
25. Hide completed items
26. Navigate to CncFileTransmissionView
27. Article: Show CNC-files
28. Show
29. Hide past items
30. Swap Items in stock
31. Article: open stock information
32. Navigate to RohmaterialSearcherView
33. open einlagerungen
34. Article: unmount
35. open goods issue
36. Navigate to SettingsView
37. Navigate to AboutView
38. Navigate to RohmaterialListView
39. RawMaterial: Search
40. Navigate to RohmaterialDetailView
41. RawMaterial open
42. Navigate to RohmaterialChangeView
43. Navigate to NFCWriterView
44. RawMaterial Remove
45. RawMaterial Clear
46. Search
47. Crate: Search

- **State 2**

  1. Navigate to OrderListView
  2. Order: Search
  3. Store Item
  4. Clear goods issue

- **State 3**

  1. Navigate to WarenausgangView
  2. Crate: Search
  3. Select Kolli from List

- **State 4**

  1. Article: Search
  2. Article: open

3. Navigate to ArticleStockChangeView
4. Store Item

Figure 4.8 plots the log-likelihoods resulting from IIHMM construction for different values of k. The best resulting model is the one with k = 3 and k = 4 as they share the same best log-likelihood. From now on, we will just consider the HMM resulting with $k = 4$ as best HMM, as it is equal to the one produced with $k = 3$. Figure 4.7 shows the directed graph of the transition matrix resulting from the IIHMM construction process. Nodes represent states and contain symbols constrained onto them. Directed edges represent the transitions from the current state (node where the edge starts from) to the next state (node to that the edge points . Numbers labeling edges represent the probability for the transition to occur.

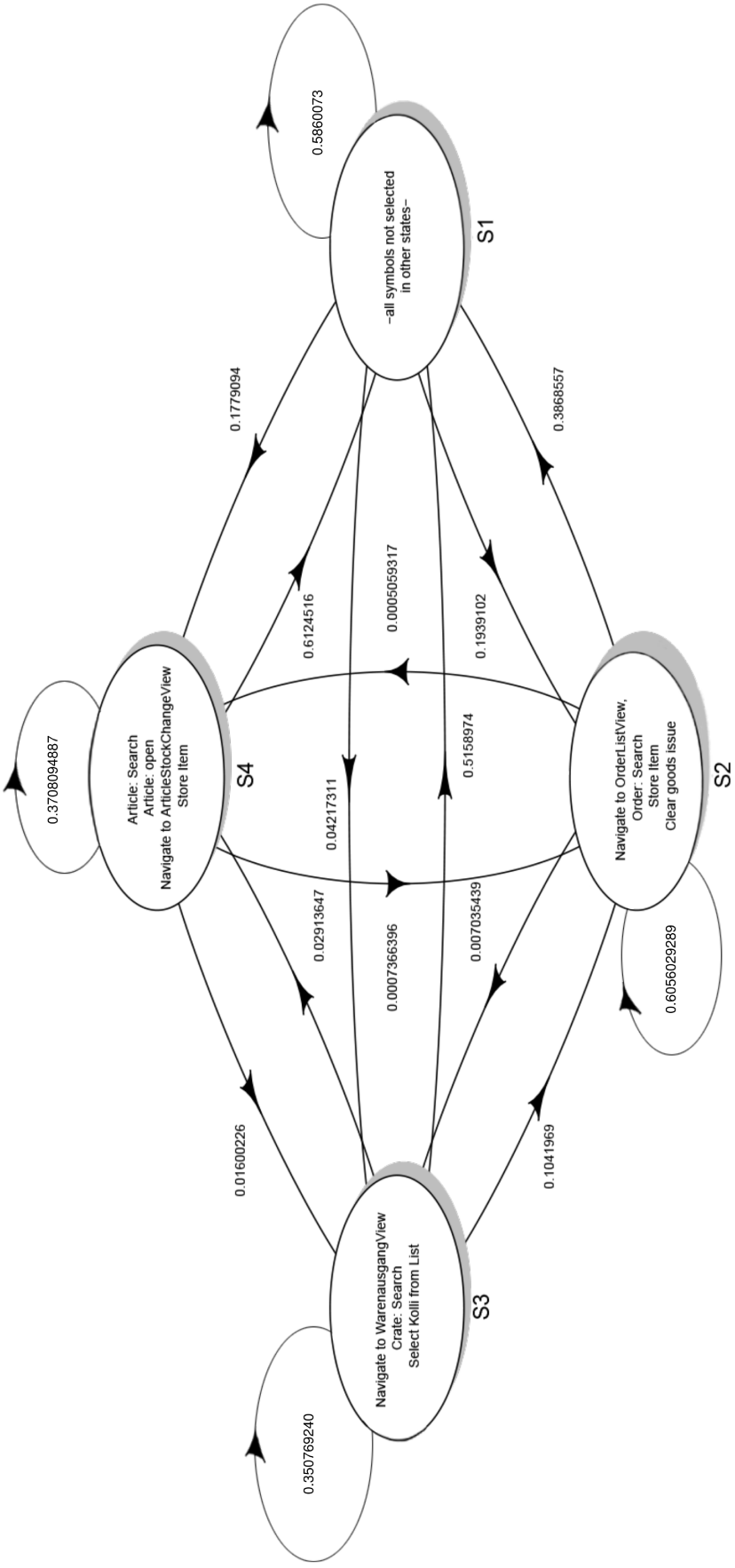


Figure 4.7: Transition Matrix for resulting constructed IIHMM

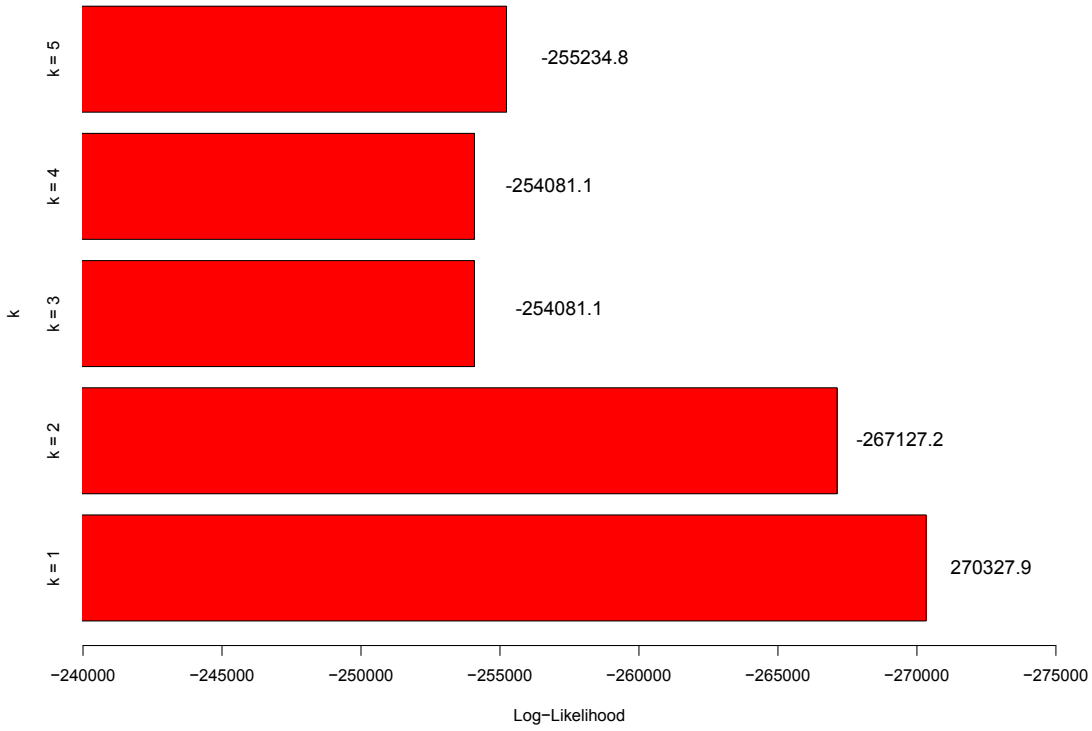


Figure 4.8: Plot of log-likelihoods resulting from the IIHMM construction process for different values of k. With k = 5, the IIHMM model quality degraded, whereas with k = 4, no new task was identified by the manager.

4.5.3. AIHMM Construction

We made use of the algorithm described in Section 3.4 to build AIHMMs with k = 1,2,3,4,5 by iteratively considering the top k θ-interesting sequences. All the symbols of the found interesting sequences that are not contained in the existing states of such HMM have been selected to be included in the next state. We iterated this procedure until the quality of the generated model degrades, as described in Section 3.4.

Best Model: The plot in Figure 4.9 shows the log-likelihood resulting from AIHMM construction for different values of k on the one-year dataset. The best log-likelihood of the AIHMMs was achieved for k = 1 (Log-likelihood = -247595.46). Its resulting transition matrix graph is shown in Figure 4.10.

Unfortunately, this model included only 2 states, which were not very meaningful. Therefore, we selected the second-best log-likelihood of the AIHMMs, which was achieved for k = 4 (Log-likelihood = -251416.6). Following, we show the symbols constrained on the different states of the AIHMM generated with k = 4. Resulting AIHMMs with states and symbols constrained on each state for k = 1,2,3,5 are shown in Appendix A. Figure 4.13 shows the transition matrix graph.

One-month comparison: To understand the reason of the low number of states for k =

1, we repeated the AIHMM process for one month of logs. The plot in Figure 4.12 shows the log-likelihood increase for different values of k on one-month dataset. In this case, the best model was for $k = 1$ and the graph had many more states, as it can be noted from Figure 4.14. Thus, we decided to include also this graph and submit three graphs to the stakeholder (i.e., one year $k = 1, 4$ and one month $k = 1$).

AIHMM with k = 4

Final Amount of states: five
Resulting log-likelihood: -251416.590533526

- **State 1**:

  1. Put Item to Goods issue
  2. crate
  3. Article: Open drawing
  4. Article: open child
  5. Reserve Item for mounting
  6. Article: mount
  7. Article: check in
  8. open montagereservierung
  9. Select Kolli from List
  10. Add Items to stock
  11. Remove Items from stock
  12. Swap crate
  13. location
  14. write NFC tag
  15. open order list
  16. Hide completed items
  17. Navigate to CncFileTransmissionView
  18. Article: Show CNC-files
  19. Hide past items
  20. Swap Items in stock
  21. open einlagerungen
  22. Article: unmount
  23. open goods issue
  24. Navigate to SettingsView
  25. Navigate to AboutView
  26. Navigate to RohmaterialListView
  27. RawMaterial: Search
  28. Navigate to RohmaterialDetailView

29. RawMaterial open
30. Navigate to RohmaterialChangeView
31. RawMaterial Remove
32. RawMaterial Clear
33. Search

- **State 2**:

1. Navigate to ArticleSearcherView
2. Article: Search
3. Navigate to ArticleDetailView
4. Navigate to OrderSearcherView
5. Navigate to OrderListView
6. Order: Search
7. Store Item
8. Clear goods issue
9. Navigate to ReservierungView
10. Add secondary tile

- **State 3**:

1. Navigate to KistenInformationView
2. Crate: Search
3. Show
4. Navigate to NFCWriterView

- **State 4**:

1. Open Order
2. Article: open
3. Navigate to ArticleStockChangeView
4. Navigate to NfcWriterView
5. Write NFC
6. Article: open stock information

- **State 5**:

1. Navigate to EinlagerungenView
2. Navigate to WarenausgangView
3. Navigate to RohmaterialSearcherView

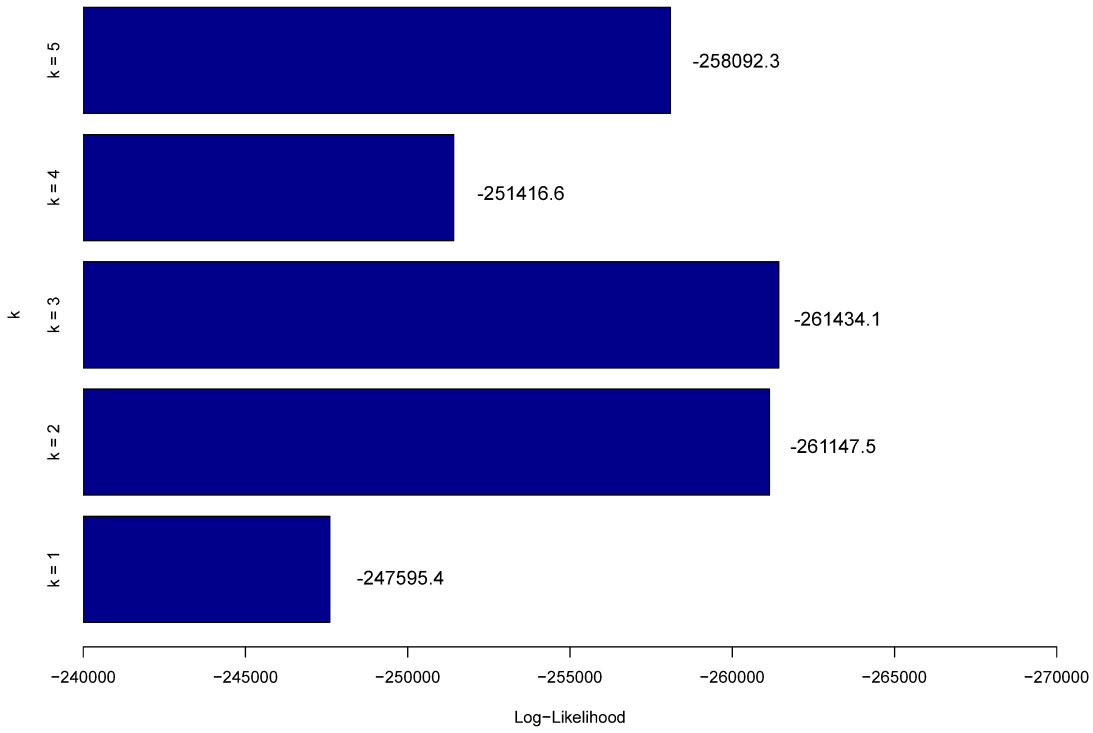


Figure 4.9: Log-Likelihood resulting from AIHMM construction for different values of *k* on one-year dataset.

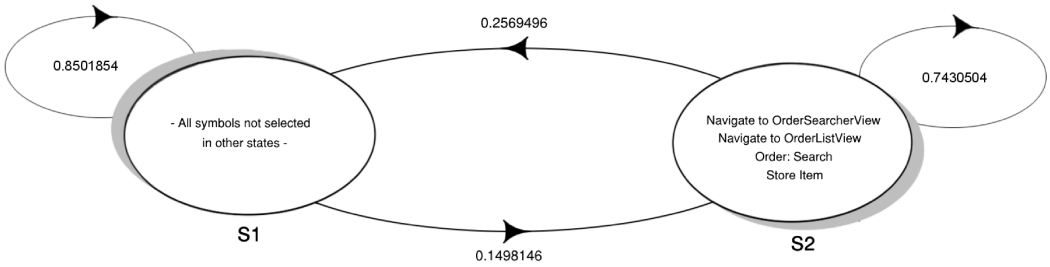
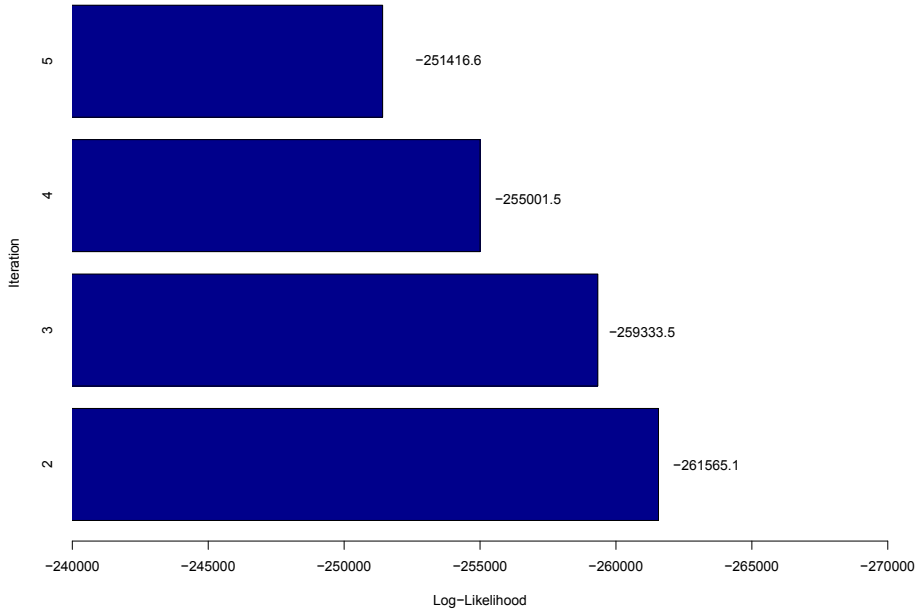


Figure 4.10: Transition Matrix graph for AIHMM (first-best log-likelihood) with k = 1 on one-year dataset



1

Figure 4.11: Log-Likelihood resulting from AIHMM construction for $k = 4$ on one-year dataset.

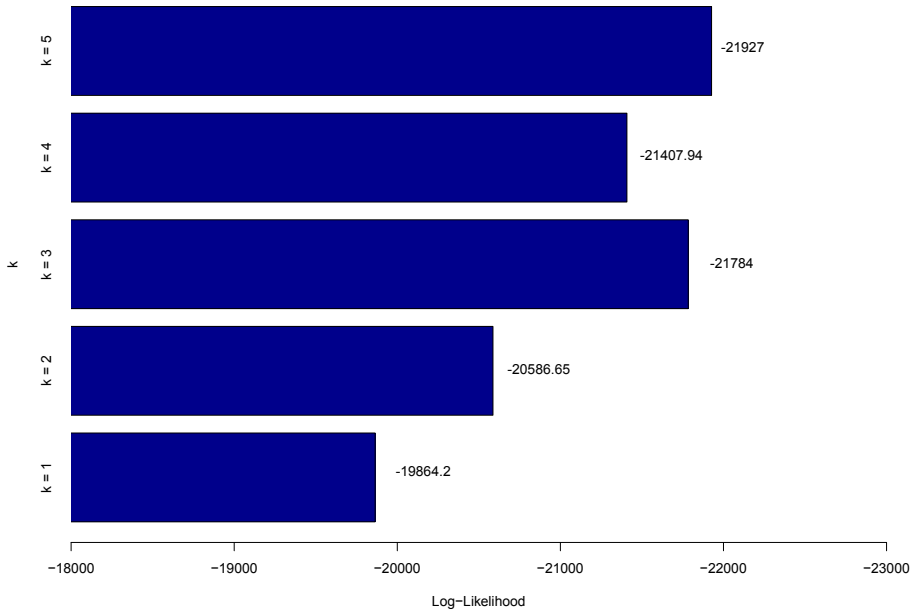


Figure 4.12: Log-Likeihood resulting from AIHMM construction for different values of k on one-month dataset

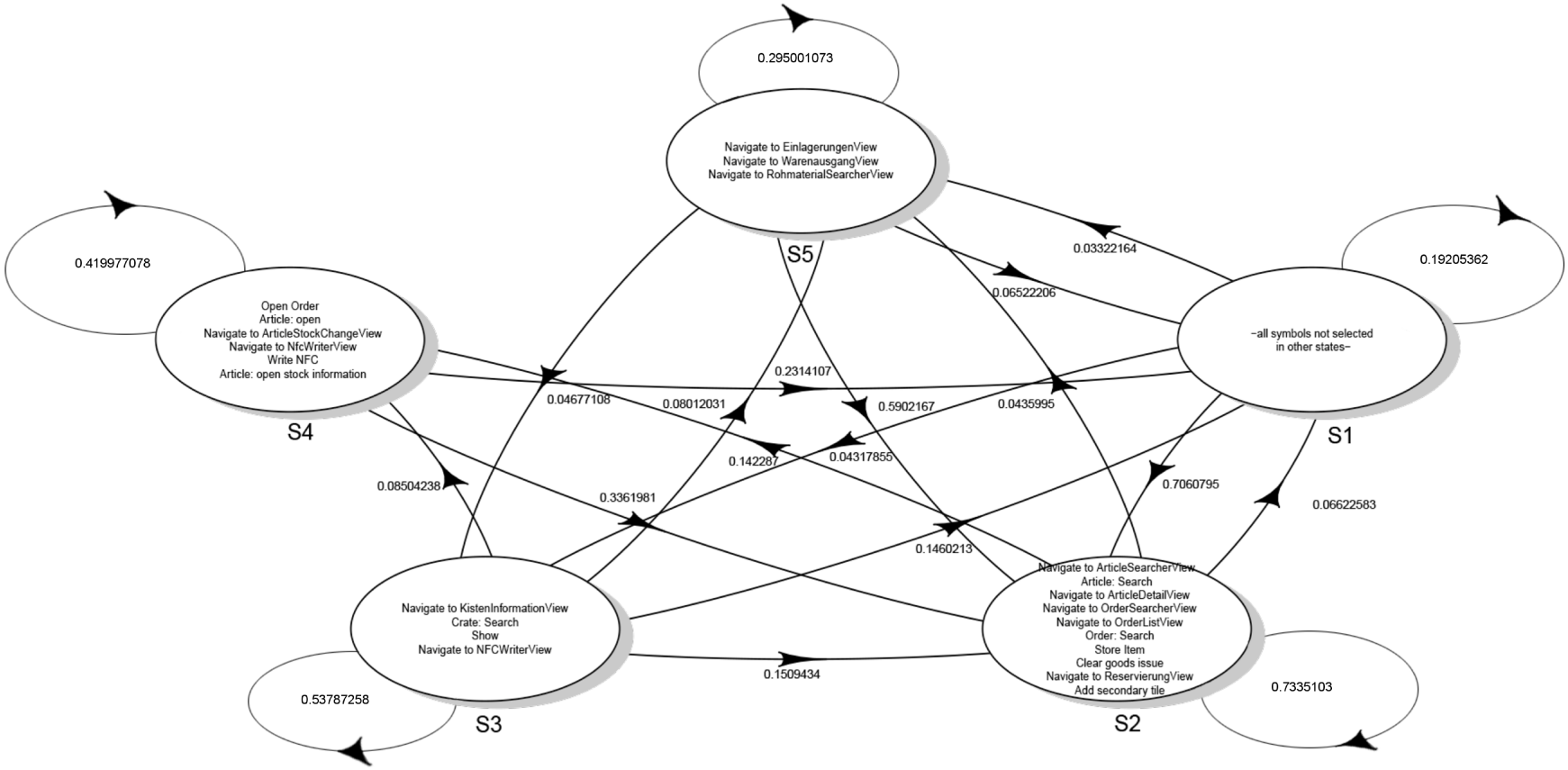


Figure 4.13: Transition Matrix graph for AIHMM (second-best log-likelihood) with $k = 4$ on one-year dataset

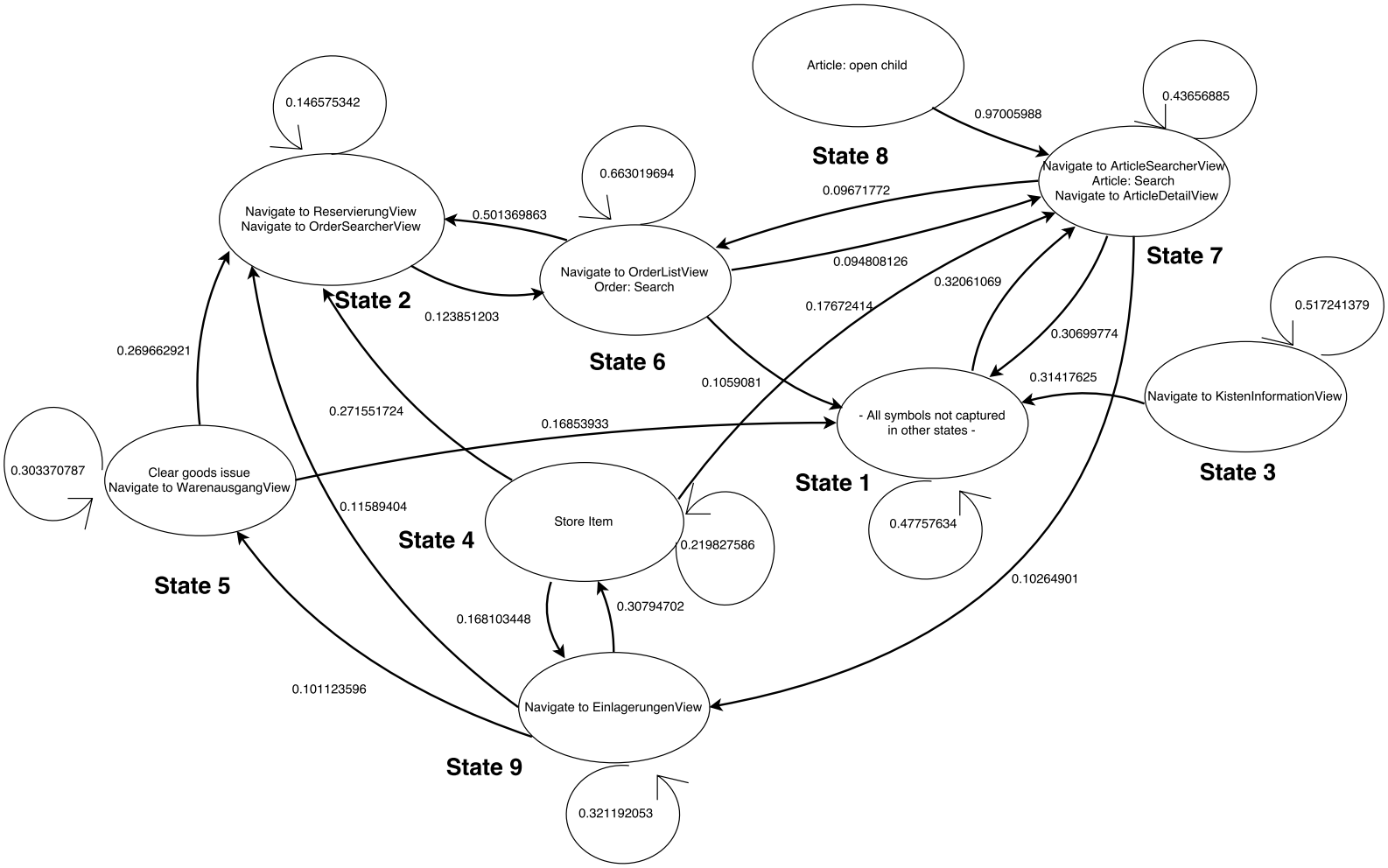


Figure 4.14: Transition Matrix graph for AIHMM (first-best log-likelihood) with $k = 1$ on one-month dataset

4.5.4. Missing tasks in comparison with the DISCO model

For every created task T in the generated IIHMM, the company stakeholder was asked the following questions. Afterwards, we display the answers to the questions we obtained from the company stakeholder.

1. M to CS: “Can you find symbols constrained on Task T in the Large DISCO view selected (Fig. 4.5) ?"
2. If a symbol did not appear to be in the DISCO model, but the CS still selected it, M asked the following follow-up question to the CS:
   M to CS:“Why did you assign symbol S to task T"?
3. M to CS: “Would you like to add any symbol from the DISCO view to task T? (i.e: symbols missing in task T but present in the DISCO view)"

Deliver Items

CS: “All symbols selected in the task ’Deliver Items’ are in the DISCO view, apart from "Clear Goods Issue. I selected the symbol "Clear Good Issue" because it is part of a typical interaction with the system, even if it does not occur very frequently. Namely, it should show up just after "Put item to Goods issue". The symbol "Select kolli from list" can be added to the task and, optionally, "Deliver item’. "

Modify kolli for delivery

CS: “The only symbol present in the DISCO view selected is "Navigate to WarenausgangView", but "Crate: Search" and "Select Kolli from List" do not show up. Thus, even if this task should occur very frequently, the DISCO model does not capture it. No other symbols from the DISCO view could be added to this task.

Change location of article in stock

CS: “All symbols selected in the task "Change location of article in stock" are present in the DISCO view. Hence, the task is well captured by the DISCO view. No other symbols from the DISCO view could be added to this task."

List Content of Crate

CS: “The symbol "Crate: Search" from this task does not show up in the DISCO view, hence the task is not captured by the DISCO model.

Missing Tasks

M: “Additionally, the following tasks are not present in the IIHMM, but are present in the DISCO view. These could compound the following symbols as well:

- *Search for Open Orders*: "Navigate to OrderSearcherView", Order: Search", "Navigate to OrderListView"
- *Show Stock Content of Article*: "Navigate to ArticleSearcherView", "Article: Search", "Navigate to ArticleDetailView" "

Chapter 5

Discussion

5.1. Accuracy of AIHMMs and IIHMMs

Before submitting the IHMMs to the company's stakeholder, we compared the accuracy of the 5+5 models generated with the interactive and automatic approach as pointed out in the Experimental validation (Section 4.5, setting k = 1,2,3,4,5). Figure 5.1 compares the log-likelihood of the obtained IIHMMs with the log-likelihood of the obtained AIHMMs. The higher the value of the log likelihood, the better the accuracy. Hence, three out of five AIHMMs outperform the corresponding IIHMMs.

In particular, the two best log-likelihood values (k=1,4) correspond to AIHMMs.

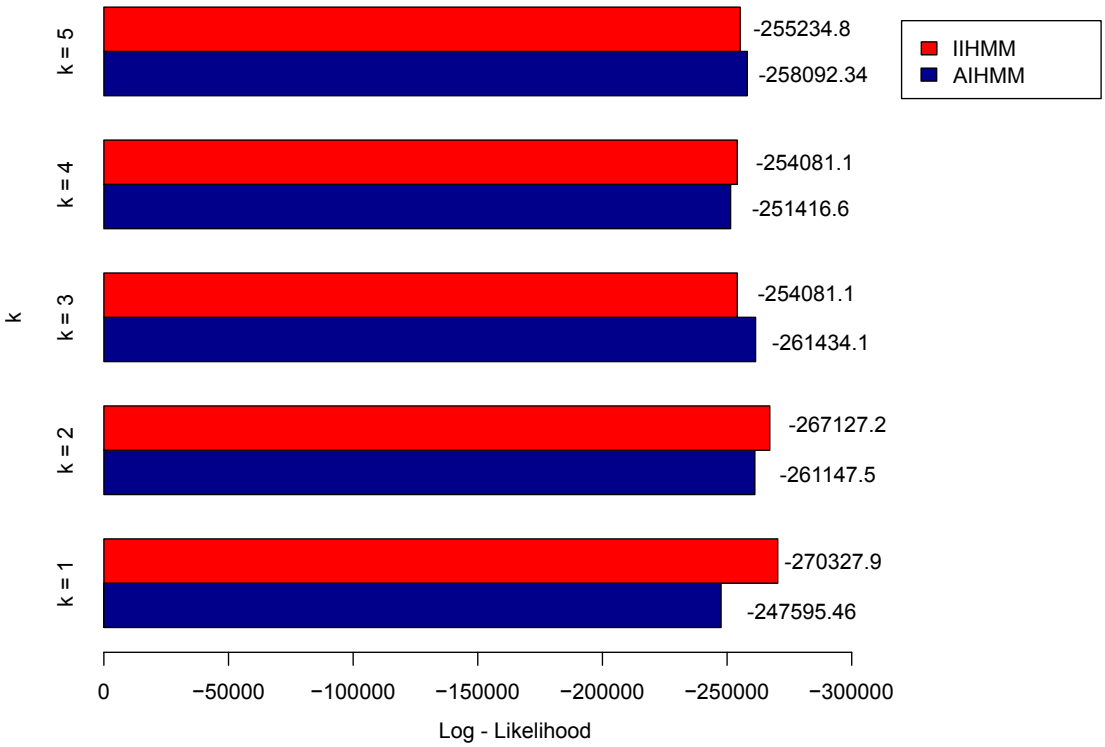


Figure 5.1: Comparison of IIHMM and AIHMM's final log-likelihood for different values of k

We replicated the analysis of the accuracy of AIHMMs for one-month dataset, compared the likelihood from Figure 4.5.3. It is worth noticing that the accuracy of any one-month resulting model is superior of any one-year model and the best model is very rich in states, as Figure 4.14 shows.

5.2. Models' Discussion with the Manager

We presented to the company stakeholder the models we have generated with DISCO (in Section 4.5.1), AIHMM (in Section 4.5.3), and IIHMM (in Section 4.5.2). We then asked the company stakeholder to label the generated AIHMMs' states and compare the log graphs in terms of the accuracy he perceives in representing users' tasks in the mobile ERP system.

Figure 5.2, Figure 5.3, Figure 5.4, Figure 5.5 illustrate respectively the figures with labels assigned to the AIHMM with $k = 1$ on one-year dataset, IIHMM, AIHMM with $k = 4$ on one-year dataset and AIHMM with $k = 1$ on one-month dataset as defined and refined by the company stakeholder. The caption underneath represents his evaluation on the expressiveness of the graph.

The Company Stakeholder evaluated the model he built iteratively positively (IIHMM on the one-year-dataset IHMM with $k = 4$, Fig. 5.3) and, interestingly, the AIHMM with $k = 1$ (Figure 4.14) built from the one-month-dataset, as well as the AIHMM with $k = 1$ built from the one-year dataset (Fig 5.2) . In particular, he found the AIHMM with $k = 1$ built on the one-month dataset very similar to the IIHMM he built interactively.

From the obtained graphs, the paths describing actions encompassing different states' change can be inferred, based on the transition probability between states.

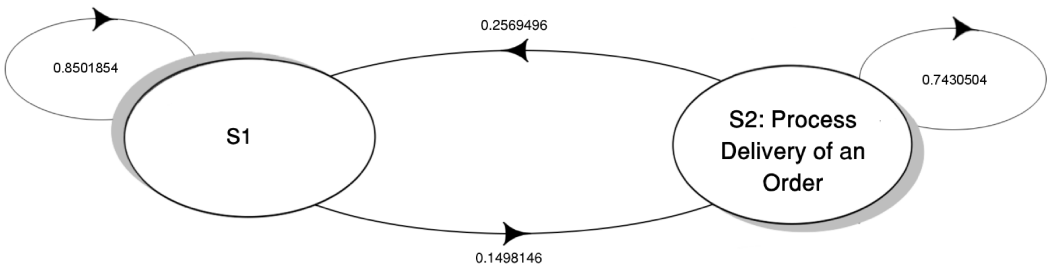


Figure 5.2: AIHMM with $k = 1$ built on one-year dataset.
Comment:Good overview of what the user is mostly doing: search for orders and modify the stock-amount of an article (e.g. put the article to the goods-issue).

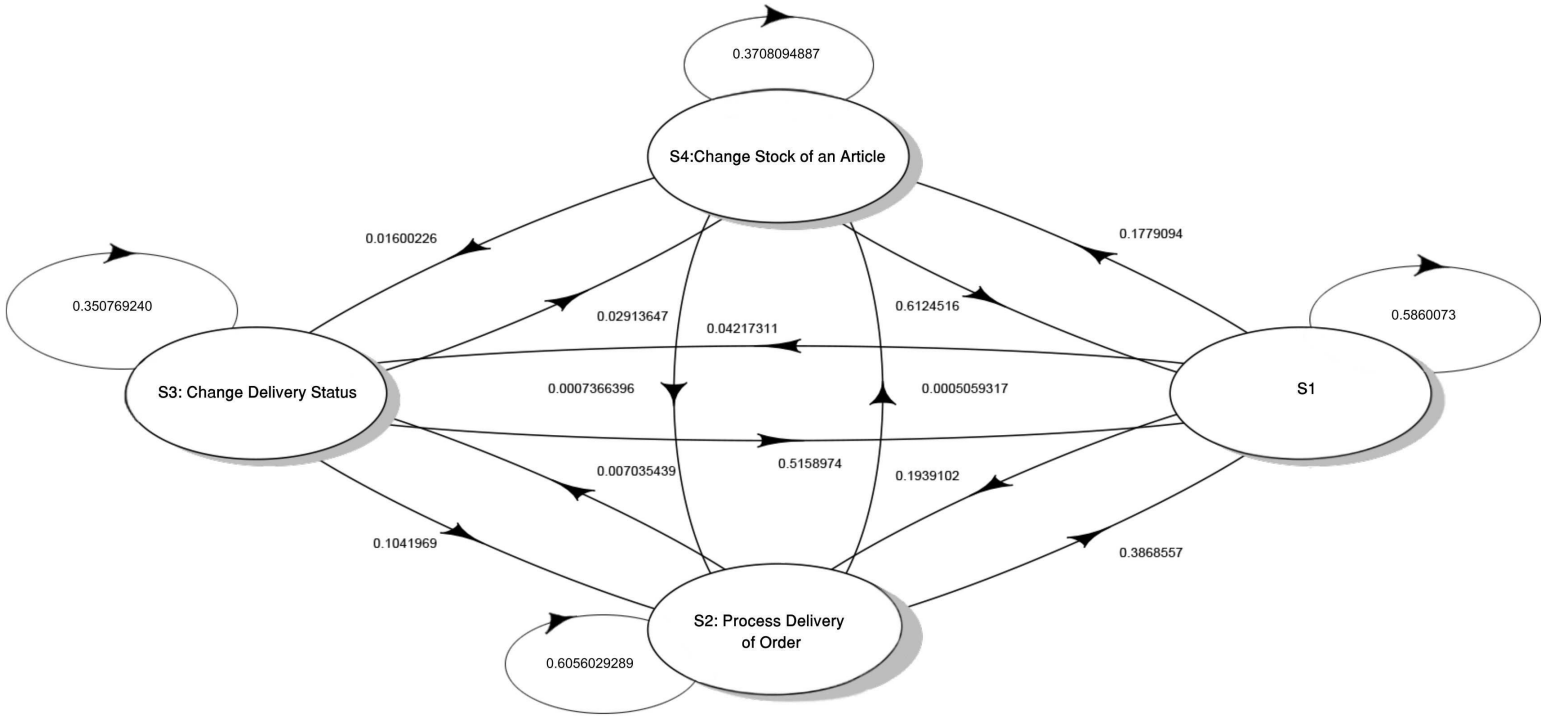


Figure 5.3: IIHMM with $k = 4$ built on one-year dataset.
Comment: It represents a perfect overview about how the user behaves if he wants to deliver articles (e.g., search for open orders, change article stock, delivery).

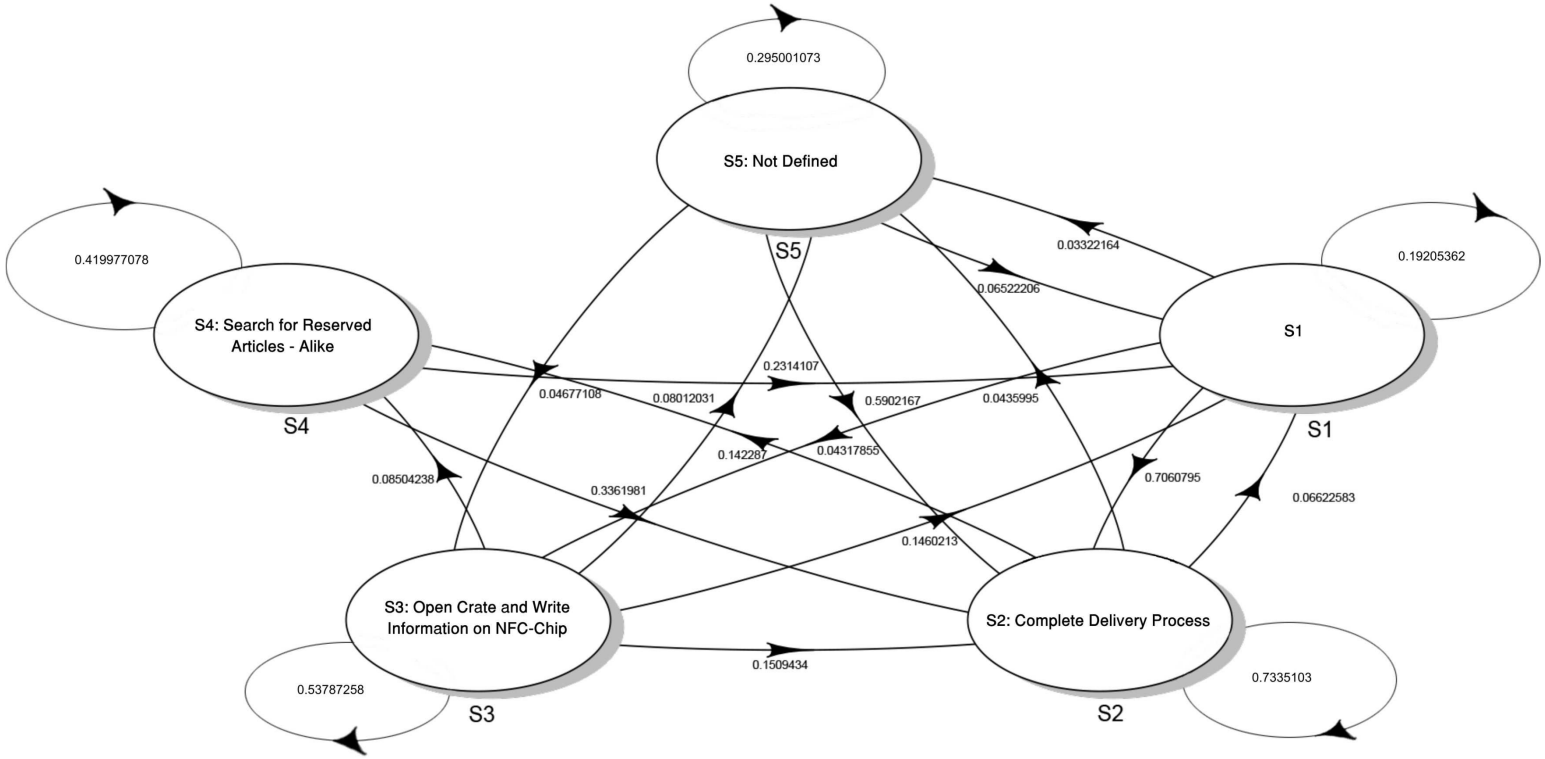


Figure 5.4: AIHMM with $k = 4$ built on one-year dataset.
Comment: Weak model. There are items in the states that do not fit together.

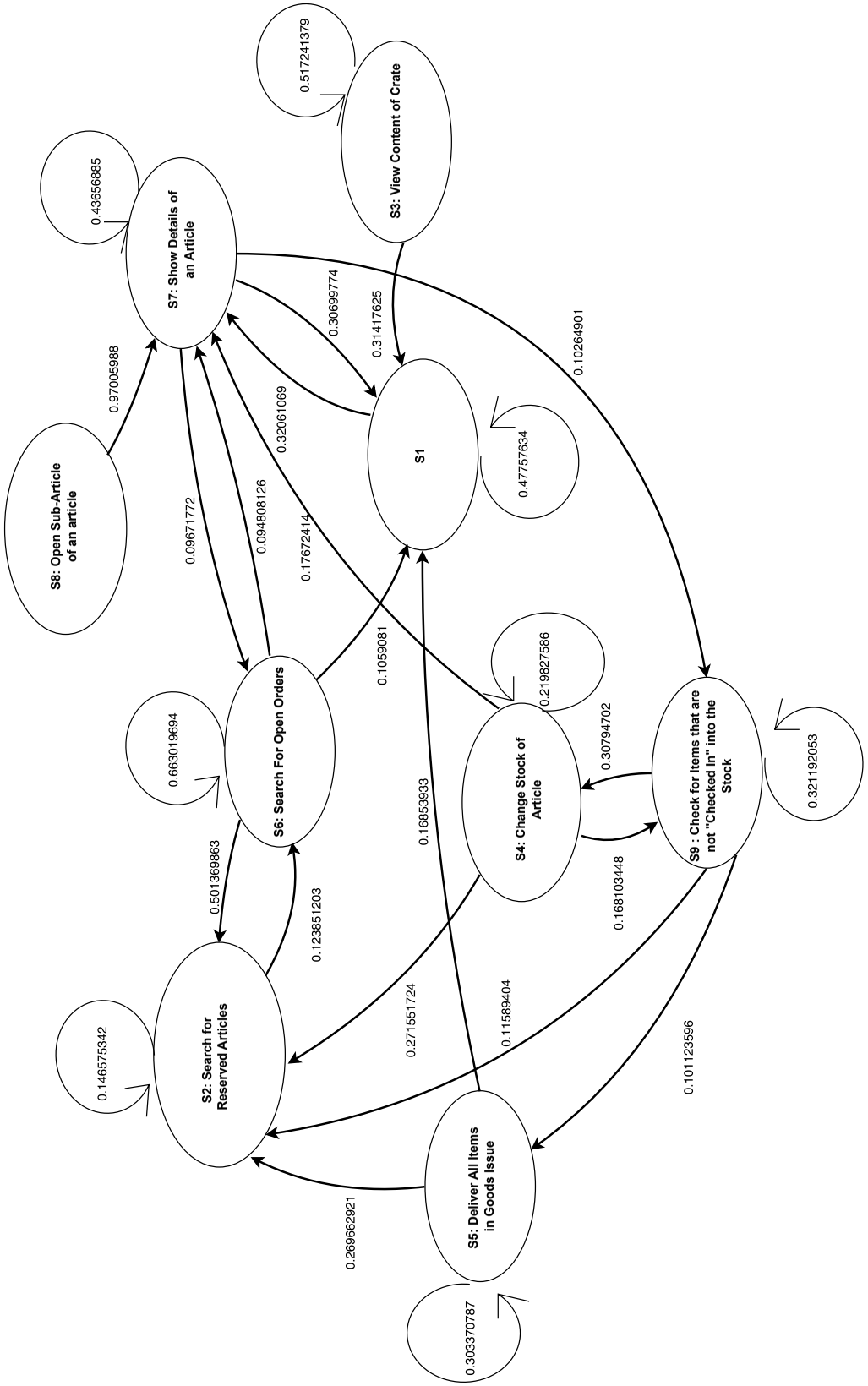


Figure 5.5: AIHMM with $k = 1$ built on one-month dataset.
Comment: It is a more detailed view of the IIHMM, but it represents slightly the same tasks.

Chapter 6

Conclusions

6.1. Summary

In the first chapter, we described the *Method Design* (Chapter 3), then we laid out the *Evaluation* of the proposed method by conducting an experiment on a concrete industrial study case (Chapter 4). Results obtained from this experiment underwent *Discussion* and were validated together with the process manager (Chapter 5). More specifically, the work was sub-divided into the following phases:

1. Understanding Damevski *Et al.*'s approach [5] and their algorithms for IIHMM construction.
2. Re-factoring and re-engineering Damevski *Et al.*'s approach and fixing issues in the algorithms' pseudocode.
3. Implementing Damevski *Et al.*'s algorithms in R.
4. Modifying the re-factored and re-engineered algorithms for AIHMM construction.
5. Validate algorithms experimentally through an industrial study case.

The whole process started in July 2016 and finished in July 2017, spanning over a time-frame of about one year.

6.2. Results

6.2.1. Automatic Use Model construction

The result of the present work is an automated tool for providing managers a use model of their processes and their tasks. Models automatically produced through this tool provide a representation of users' tasks that existing process mining tools (e.g: DISCO) do not provide. We conducted an experimental comparison of Damevski *Et al.*'s manual approach with our semi-automated tool on an industrial study case. Our experimental results indicate that the approach of Damevski *Et al.* can be helpful and effective in manually reconstructing users' tasks iteratively. However, the present work also shows that the process of reconstructing users' tasks can be automated and the manager's knowledge of the system can be replaced by the automated tool. Particular attention needs to be paid to the

dataset used: in fact, we found out that the quality of the model automatically iteratively built on the one-month dataset surpassed the quality of the model built on the one-year dataset and could better represent the manager's idea of the process. The best resulting model also had very similar expressiveness to the model built manually with the company stakeholder. To explain why the one-month model quality surpassed the one-year model quality, we hypothesized that the usage of the ERP system being studied is dependent on the commissioned work and in the one-year time frame, the work commissioned may vary vastly, whereas in the one-month time frame, the work may be more revolving around more specific activities (e.g: commissions of a few customers). The stakeholder confirmed our hypothesis as far as work diversification in the one-month and one-year period is concerned. We also hypothesized that fewer system changes might have occurred in the one-month time frame, in comparison with the one-year time frame. In fact, during the one-year time frame, the system may have undergone more frequent and substantial changes. This may have led to a more varied usage, which was not well captured by the HMM. Consequently, we interviewed the stakeholder and asked him which changes the ERP system underwent in the one-month period and in the one-year period and whether the system underwent substantial changes. The stakeholder reported that in the one-month period, a single GUI change occurred, whereas in the one-year period, the ERP system underwent very substantial structural changes.

Comparison of the proposed method on different log datasets (on sliding time windows) is out of the scope of the present work, but it will be matter of future research.

6.2.2. DISCO Limitations

Only after the stakeholder saw and discussed the IHMMs, he understood the limitation of the visualization provided in DISCO. He realized that as manager, the DISCO model that he selected as best representing the use process was still representing single actions of the user and not the tasks that those actions aimed to achieve.

While implementing our method, we realized that the original algorithm of Damevski *et al.* [5] that finds the longest common subsequences, has a limitation that can have some negative effects in the analysis of long sequences of logs. In the context of developers' debugging behaviour in IDE, as in [5], developers perform actions based on the goal they want to achieve through the IDE (their task). However, in our context of users using an ERP system, users may simply roam around the system before performing the very task they want to achieve.

Consequently, we have extended the algorithm that counts the frequency of sub-sequences (LCS) to match sub-sequences in any part of a sequence. Thus, using the experience we gathered re-engineering the algorithms of Damevski *et al.* and our R implementations, we have been able to define a new algorithm and a script that finds the longest sub-sequence within sequences that are not only composed of prefixes. In other words, our new algorithm is able to identify common tasks that might not start soon but rather after some initial activities of the user to the system that might just be explorative and not related to the specific task the user wants to achieve. The new algorithm will be used in future work.

Chapter 7

Future Work

7.1. Algorithms' Optimization

The algorithms proposed by Damevski et al. were implemented in R, a programming language widely used in the academia with excellent capabilities in data processing and statistical analysis. R is a fairly high level language, hence it compromises computational efficiency for higher-level intelligibility and development efficiency. Our experimental setting for AIHMM construction consisted of a laptop equipped with an Intel Core (^{TM}) i7-4710 HQ with 2.5 GHz clock speed. The algorithm ran in the following timeframe:

- Start: Monday June 05 10:38:58 PM 2017
- End: Monday Jun 05 11:36:51 PM 2017

Hence, the overall duration of the AIHMM construction process was 0 hours, 57 minutes and 53 seconds.

For bigger datasets, given the high computational demand (especially in terms of CPU) required to train an HMM with the Baum Welch Algorithm, the algorithm may perform poorly. Applying the algorithm on bigger datasets will be subject of future work.

In particular, R appears not to be efficient at running the Baum Welch algorithm. Tables 7.1, 7.2, 7.3 show different versions of HMMs trained with the Baum-Welch algorithm by using different programming languages: the time is given in seconds. In this context, R runs very slow in comparison with other programming languages.

We propose the following solutions:

- *Parallelization of the Baum Welch Algorithm*: If the execution of the Baum Welch algorithm could be parallelized on multiple cores, a clear performance boost in training the HMM could take place.

Table 7.1: Small" comparison scenario 1: The Hidden Markov chain has two states, and is trained on a four-letter observations alphabet [28].

	python	**python + C**	**R**	**R + C**	**matlab**	**matlab + C**	**octave + C**	**C++ (armadillo)**	**pure C**
creation of the sample	0.076	0.076	0.25	0.25	0.031	0.031	0.59	0.00082	0.00053
filtering/ smoothing	0.29	0.00072	0.33	0.0028	0.076	0.00072	0.00072	0.0017	0.00054
10 EM	296	2.14	333	9.20	76.5	1.60	3.38	2.5	0.94

Table 7.2: "Medium" comparison scenario 2: The Hidden Markov Chain has 7 hidden states and is trained on 7 possible observations [28].

	python	**python + C**	**R**	**R + C**	**matlab**	**matlab + C**	**octave + C**	**C++ (armadillo)**	**pure C**
creation of the sample	0.14	0.14	0.27	0.27	0.032	0.032	0.62	0.0012	0.00058
filtering/ smoothing	0.32	0.0035	0.38	0.0067	0.081	0.0030	0.0027	0.0038	0.0022
10 EM	331	15.2	395	19.4	87.1	10.9	13.0	9.55	2.81

Table 7.3: "Large" comparison scenario 3: the Hidden Markov Chain has 50 hidden states and is trained on 50 possible observations [28].

	python	**python + C**	**R**	**R + C**	**matlab**	**matlab + C**	**octave + C**	**C++ (armadillo)**	**pure C**
creation of the sample	0.28	0.28	0.30	0.30	0.042	0.042	0.63	0.0058	0.0015
filtering/ smoothing	0.57	0.14	0.68	0.14	0.15	0.094	0.083	0.064	0.073
10 EM	688	355	858	384	282	315	331	246	43

- *Usage of a lower- level language for the Baum Welch Algorithm part*: if the Baum Welch algorithm was executed on a lower-level language (such as C), a performance boost could be obtained.

Appendix A

Resulting AIIHMMs for k = 1,2,3,5

In the present Appendix, we outline all the AIHMMs resulting from the AIHMM construction process described in Section 3.4, specifying their amount of states, symbols constrained on each state and log-likelihood.

A.1. AIHMM with k = 1

Resulting log-likelihood = -247595.459589602
Final Amount of states: two

- State 1

    1. Navigate to ArticleSearcherView
    2. Article: Search
    3. Navigate to ArticleDetailView
    4. Navigate to EinlagerungenView
    5. Navigate to WarenausgangView
    6. Open Order
    7. Article: open
    8. Navigate to ArticleStockChangeView
    9. Put Item to Goods issue
    10. Navigate to KistenInformationView
    11. Crate: Search
    12. crate
    13. Clear goods issue
    14. Article: Open drawing
    15. Article: open child
    16. Reserve Item for mounting
    17. Article: mount
    18. Article: check in

19. Navigate to ReservierungView
20. open montagereservierung
21. Select Kolli from List
22. Add Items to stock
23. Remove Items from stock
24. Swap crate
25. location
26. Navigate to NfcWriterView
27. Write NFC
28. write NFC tag
29. open order list
30. Add secondary tile
31. Hide completed items
32. Navigate to CncFileTransmissionView
33. Article: Show CNC-files
34. Show
35. Hide past items
36. Swap Items in stock
37. Article: open stock information
38. Navigate to RohmaterialSearcherView
39. open einlagerungen
40. Article: unmount
41. open goods issue
42. Navigate to SettingsView
43. Navigate to AboutView
44. Navigate to RohmaterialListView
45. RawMaterial: Search
46. Navigate to RohmaterialDetailView
47. RawMaterial open
48. Navigate to RohmaterialChangeView
49. Navigate to NFCWriterView
50. RawMaterial Remove
51. RawMaterial Clear
52. Search

- State 2

1. Navigate to OrderSearcherView
2. Navigate to OrderListView
3. Order: Search
4. Store Item

A.2. AIHMM with k = 2

Resulting log-likelihood = -261147.503367496
Final Amount of states: two

- State 1:

    1. Navigate to EinlagerungenView
    2. Navigate to WarenausgangView
    3. Open Order
    4. Article: open
    5. Navigate to ArticleStockChangeView
    6. Put Item to Goods issue
    7. Navigate to KistenInformationView
    8. Crate: Search
    9. crate
    10. Clear goods issue
    11. Article: Open drawing
    12. Article: open child
    13. Reserve Item for mounting
    14. Article: mount
    15. Article: check in
    16. Navigate to ReservierungView
    17. open montagereservierung
    18. Select Kolli from List
    19. Add Items to stock
    20. Remove Items from stock
    21. Swap crate
    22. location
    23. Navigate to NfcWriterView
    24. Write NFC
    25. write NFC tag
    26. open order list
    27. Add secondary tile
    28. Hide completed items
    29. Navigate to CncFileTransmissionView
    30. Article: Show CNC-files
    31. Show
    32. Hide past items

33. Swap Items in stock
34. Article: open stock information
35. Navigate to RohmaterialSearcherView
36. open einlagerungen
37. Article: unmount
38. open goods issue
39. Navigate to SettingsView
40. Navigate to AboutView
41. Navigate to RohmaterialListView
42. RawMaterial: Search
43. Navigate to RohmaterialDetailView
44. RawMaterial open
45. Navigate to RohmaterialChangeView
46. Navigate to NFCWriterView
47. RawMaterial Remove
48. RawMaterial Clear
49. Search

- State 2

1. Navigate to ArticleSearcherView
2. Article: Search
3. Navigate to ArticleDetailView
4. Navigate to OrderSearcherView
5. Navigate to OrderListView
6. Order: Search
7. Store Item

A.3. AIHMM with k = 3

Resulting log-likelihood = -261434.092065734
Final Amount of states = 2

- State 1

1. Navigate to EinlagerungenView
2. Navigate to WarenausgangView
3. Open Order
4. Article: open
5. Navigate to ArticleStockChangeView
6. Put Item to Goods issue

7. Navigate to KistenInformationView
8. Crate: Search
9. crate
10. Article: Open drawing
11. Article: open child
12. Reserve Item for mounting
13. Article: mount
14. Article: check in
15. Navigate to ReservierungView
16. open montagereservierung
17. Select Kolli from List
18. Add Items to stock
19. Remove Items from stock
20. Swap crate
21. location
22. Navigate to NfcWriterView
23. Write NFC
24. write NFC tag
25. open order list
26. Add secondary tile
27. Hide completed items
28. Navigate to CncFileTransmissionView
29. Article: Show CNC-files
30. Show
31. Hide past items
32. Swap Items in stock
33. Article: open stock information
34. Navigate to RohmaterialSearcherView
35. open einlagerungen
36. Article: unmount
37. open goods issue
38. Navigate to SettingsView
39. Navigate to AboutView
40. Navigate to RohmaterialListView
41. RawMaterial: Search
42. Navigate to RohmaterialDetailView
43. RawMaterial open

44. Navigate to RohmaterialChangeView
45. Navigate to NFCWriterView
46. RawMaterial Remove
47. RawMaterial Clear
48. Search

State 2

1. Navigate to ArticleSearcherView
2. Article: Search
3. Navigate to ArticleDetailView
4. Navigate to OrderSearcherView
5. Navigate to OrderListView
6. Order: Search
7. Store Item
8. Clear goods issue

A.4. AIHMM with k = 5

Resulting log-likelihood = -258092.33893727
Final Amount of states = four

- State 1

1. Put Item to Goods issue
2. crate
3. Article: Open drawing
4. Reserve Item for mounting
5. Article: mount
6. Article: check in
7. open montagereservierung
8. Add Items to stock
9. Remove Items from stock
10. Swap crate
11. location
12. Navigate to NfcWriterView
13. Write NFC
14. write NFC tag
15. open order list
16. Hide completed items
17. Navigate to CncFileTransmissionView

18. Article: Show CNC-files
19. Hide past items
20. Swap Items in stock
21. Article: open stock information
22. open einlagerungen
23. Article: unmount
24. open goods issue
25. Navigate to SettingsView
26. Navigate to AboutView
27. Navigate to RohmaterialListView
28. RawMaterial: Search
29. Navigate to RohmaterialDetailView
30. RawMaterial open
31. Navigate to RohmaterialChangeView
32. RawMaterial Remove
33. RawMaterial Clear
34. Search

- State 2

1. Navigate to ArticleSearcherView
2. Article: Search
3. Navigate to ArticleDetailView
4. Navigate to OrderSearcherView
5. Navigate to OrderListView
6. Order: Search
7. Store Item
8. Crate: Search
9. Clear goods issue
10. Navigate to ReservierungView
11. Add secondary tile

- State 3

1. Navigate to EinlagerungenView
2. Open Order
3. Navigate to KistenInformationView
4. Show
5. Navigate to NFCWriterView

- State 4

1. Navigate to WarenausgangView
2. Article: open
3. Navigate to ArticleStockChangeView
4. Article: open child
5. Select Kolli from List
6. Navigate to RohmaterialSearcherView

Bibliography

[1] Wil van der Aalst. Process mining: Overview and opportunities. *ACM Trans. Manage. Inf. Syst.*, 3(2):7:1–7:17, July 2012.

[2] Christian W. Günther and Wil M. P. Van Der Aalst. Fuzzy mining: Adaptive process simplification based on multi-perspective metrics. In *Proceedings of the 5th International Conference on Business Process Management*, BPM'07, pages 328–343, Berlin, Heidelberg, 2007. Springer-Verlag.

[3] Wil MP van der Aalst. *Process mining: data science in action. Chapter 11.4*. Springer, 2016.

[4] Minseok Song, Christian W. Günther, and Wil M. P. van der Aalst. *Trace Clustering in Process Mining*, pages 109–120. Springer Berlin Heidelberg, Berlin, Heidelberg, 2009.

[5] Kostadin Damevski, Hui Chen, David Shepherd, and Lori Pollock. Interactive exploration of developer interaction traces using a hidden markov model. In *Proceedings of the 13th International Conference on Mining Software Repositories*, MSR '16, pages 126–136, New York, NY, USA, 2016. ACM.

[6] Mark Gales and Steve Young. The application of hidden markov models in speech recognition. *Found. Trends Signal Process.*, 1(3):195–304, January 2007.

[7] Han Shu. On-line handwriting recognition using hidden markov models. Master's thesis, Department of Electrical Engineering and Computer Science, Massachusetts Institute of Technology, February 1997.

[8] Franz Josef Och, Google Inc. Statistical machine translation: Foundations and recent advances tutorial at mt summit phuket, thailand. `http://www.mt-archive.info/MTS-2005-Och.pdf`, 2005. [Online; accessed August 12, 2016].

[9] Alexander V Lukashin and Mark Borodovsky. Genemark. hmm: new solutions for gene finding. *Nucleic acids research*, 26(4):1107–1115, 1998.

[10] Szymon Jaroszewicz. Interactive hmm construction based on interesting sequences. In *Local Patterns to Global Models (LeGo'08) Workshop at the the European Conference on Principles and Practice of Knowledge Discovery in Databases*, 2008.

[11] Szymon Jaroszewicz. Using interesting sequences to interactively build hidden markov models. *Data Mining and Knowledge Discovery*, 21(1):186–220, 2010.

[12] Adam Oliner, Archana Ganapathi, and Wei Xu. Advances and challenges in log analysis. *Commun. ACM*, 55(2):55–61, February 2012.

[13] Errin W. Fulp, Glenn A. Fink, and Jereme N. Haack. Predicting computer system failures using support vector machines. In *Proceedings of the First USENIX Conference on Analysis of System Logs*, WASL'08, pages 5–5, Berkeley, CA, USA, 2008. USENIX Association.

[14] S. Fu and C. Z. Xu. Exploring event correlation for failure prediction in coalitions of clusters. In *Supercomputing, 2007. SC'07. Proceedings of the 2007 ACM/IEEE Conference on*, pages 1–12, Nov 2007.

[15] Kenji Yamanishi and Yuko Maruyama. Dynamic syslog mining for network failure monitoring. In *Proceedings of the Eleventh ACM SIGKDD International Conference on Knowledge Discovery in Data Mining*, KDD '05, pages 499–508, New York, NY, USA, 2005. ACM.

[16] M. Steinder and A. S. Sethi. Probabilistic fault localization in communication systems using belief networks. *IEEE/ACM Transactions on Networking*, 12(5):809–822, Oct 2004.

[17] Barbara Russo, Giancarlo Succi, and Witold Pedrycz. Mining system logs to learn error predictors: a case study of a telemetry system. *Empirical Software Engineering*, 20(4):879–927, 2015.

[18] Saulius Astromskis, Andrea Janes, and Michael Mairegger. A process mining approach to measure how users interact with software: An industrial case study. In *Proceedings of the 2015 International Conference on Software and System Process*, ICSSP 2015, pages 137–141, New York, NY, USA, 2015. ACM.

[19] Sheldon M. Ross. *Introduction to Probability and Statistics for Engineers and Scientists (Fourth Edition)*. Academic Press, 2009.

[20] Michael Kozdron. University of Regina Canada. Definition of a Stochastic Process.

[21] Mariette Awad Rahul Khanna. *Efficient Learning Machines: Theories, Concepts, and Applications for Engineers and System Designers*. First edition, April 2015.

[22] Diana Alejandra Sanchez-Salas, Jose Luis Cuevas-Ruiz, and Miguel Gonzalez-Mendoza. Wireless channel model with markov chains using matlab. In Vasilios N. Katsikis, editor, *MATLAB - A Fundamental Tool for Scientific Computing and Engineering Applications - Volume 2*, chapter 11. InTech, Rijeka, 2012.

[23] Michael Baron. *Probability and statistics for computer scientists*. CRC Press, 2013.

[24] Wikipedia. Hidden Markov Model, Figure 1: "Probabilistic parameters of a hidden Markov model (example)". `https://en.wikipedia.org/wiki/File:HiddenMarkovModel.svg`, 2016. [Online; accessed August 10, 2016].

[25] Three basic problems of HMMs. Definition of markov model. `http://jedlik.phy.bme.hu/~gerjanos/HMM/node6.html`, 2016. [Online; accessed August 12, 2016].

[26] Lawrence R. Rabiner. Readings in speech recognition. chapter A Tutorial on Hidden Markov Models and Selected Applications in Speech Recognition, pages 267–296. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 1990.

[27] Scientific Software Development Dr. Lin Himmelmann and www.linhi.com. *HMM: HMM - Hidden Markov Models*, 2010. R package version 1.0.

[28] Aurélien Garivier. The Baum-Welch algorithm for hidden Markov Models: speed comparison between octave / python / R / scilab / matlab / C / C++. `https://www.math.univ-toulouse.fr/~agarivie/Telecom/code/index.php`. [Online; accessed April 21, 2017].