Hybrid Intelligent Approaches for Business Process Sequential Analysis

Mai P. Le

A dissertation submitted in partial fulfillment
of the requirements for the degree of
Doctor of Philosophy
of the
University of Bournemouth.

Design, Engineering and Computing

Final version, 2014

Contents

List of Figures

List of Tables

Declaration

No portion of the work contained in this document has been submitted in support of an application for a degree or qualification of this or any other university or other institution of learning. All verbatim extracts have been distinguished by quotation marks, and all sources of information have been specifically acknowledged.

Signed:

Date: Final version, 2014

Abstract

The quality of customer services is an important differentiator for service oriented companies like telecommunication providers. In order to deliver good customer service, the underlying processes within the operations of a company have to run smoothly and must be well controlled. It is of great importance to be able to predict if processes are likely to fail and to be aware of developing problems as early as possible. A failure in a customer service process typically results in a negative experience for a customer and companies are keen to avoid this from happening. Process performance prediction allows companies to pro-actively adapt with process execution in order to prevent process problems from affecting their customers. Process analytics is often compounded by a number of factors. Very often processes are only poorly documented because they have evolved over time together with the legacy IT systems that were used to implement them. The workflow data that is collected during process execution is high dimensional and can contain complex attributes and very diverse values. Since workflow data is sequential in nature, there are a number of data mining methods such as sequential pattern mining and probabilistic models that can be useful for predicting process transitions or process outcomes. None of these techniques alone can adequately cope with workflow data. The purpose of this thesis is to contribute a combination of methods that can analyse data from business process in execution in order to predict severe process incidents. In order to best exploit the sequential nature of the data we have used a number of sequential data mining approaches coupled with sequence alignment and a strategy for dealing with similar sequences. The methods have been applied to real process data from a large telecommunication provider and we have conducted a number of experiments demonstrating how to predict process steps and process outcomes. Finally, we show that the performance of the proposed models can be significantly improved if they are applied to individual clusters of workflow data rather than the complete set of process data.

Acknowledgements

I am writing here to express my thankfulness to all the people who directly and indirectly participated my journey. I am only here with their support. Many thanks to my supervisory team including Prof. Bogdan Gabrys, Prof. Detlef Nauck and Prof. Trevor Martin for their support. Prof. Bogdan Gabrys who spent time reading and helping when I needed. I know him as a serious and genuine scientist and he gave me strict supervision so that I could improve my work. Prof. Detlef Nauck has been my supervisor and mentor. He helped and encouraged me through hard moments that I could not focus on working. I am thankful for the support, opportunities and freedom that he gave me, I therefore had space to direct the project myself. I would like to send my special thanks to my family who always respect my decisions and stand by me in all circumstances. Thanks to them for trusting me and watching out for me during my time in Paris and the UK. All of my friends in Paris, Hanoi and Ipswich, with whom I shared my tears and happiness - you brightened my journey! I would like to thank Naomi Bailey, Prof. Mark Hadfield and Fiona Knight for being so kind to me. I always had great support from Bournemouth University including askBU whenever I needed help whilst I was based at British Telecom. This thesis would not be what it is now without the help of Paul Taylor. Thanks a lot Paul for your patience reading through my thesis! Big thank you to you all! I also want to thank BT for funding the research grant that supported my PhD project.

Chapter 1

Introduction

Large service companies like telecommunication businesses run complex customer service processes in order to provide communication services to their customers. The flawless execution of these processes is essential since customer service is an important differentiator for these companies. They must also be able to predict if processes will complete successfully or run into exceptions in order to intervene at the right time, pre-empt problems and maintain customer service. Very often processes are only poorly documented because they have evolved over time together with changes in the legacy IT systems that were used to implement them. Before companies can address the problem of pro-active process monitoring and process event prediction they have to identify the process model. Process mining can reconstruct process models from workflow data to some extent by searching for a model that explains the observed workflow. Workflow data is sequential in nature. There are a number of data mining methods such as sequential pattern mining (association rules) and probabilistic models that can be useful for predicting process transitions or process outcomes. In this work, sequential analysis is investigated in order to find approaches which fit for predicting process steps, process outcomes, providing process failure warnings and determining the (uncertain) process logic. These approaches are trained and tested on real business processes from a multinational telecommunications company. Methods in data mining alone are not capable of modelling highly complex and diverse business process data. The thesis will therefore demonstrate the results and outline alternative approaches.

1.1 Introduction

Business processes are important in every business organisation. A business process is a sequence of tasks (events) or activities which produce a service or product. To continue to compete companies require continuous improvement in all phases of service delivery to maintain low prices, good service, and high quality (Ruta and Majeed (2011)), (Tsui *et al.* (2005)). For this, effectively designed processes are essential.

Processes are typically designed based on knowledge about how a certain objective can be achieved efficiently. When process execution is not enforced by automatic workflow systems and people have a certain degree of freedom, they do not always follow the designed process. In large companies many departments can be involved in the execution or maintenance of a process and processes can be implemented across a large number of IT systems. In these environments it is easy for a company to lose track of the original process design and process evolves in an uncontrolled fashion. In general, organisations cannot afford to develop and maintain tailored software for business processes (i.e. automate business processes-processes executed by an automatic system). The solution is to specify the software to use and either to adapt the process to fit to the software, or to keep the process as it is and try to support the operations.

In such a scenario rediscovering a business process and preempting problems during execution are necessary in order to improve or to redesign (rebuild) the process and support operational phase in process management system. Hence the relatively young discipline of process mining (van der Aaslt et al (2011)) is getting increasing attention from researchers.

There are three types of work in process mining: discovery, conformance and enhancement (Anne (2010)), (van der Aalst (2011)). They all aim for excellence in process execution by discovering, monitoring and improving processes. Discovery involves building a model from event logs. The second type is used to check whether there is conformance between the process model and data from the logs. The target of the third type, process enhancement, is to change or extend the derived model.

Process mining contributes to many phases in business process management including the diagnosis phase, operational support, etc. (van der Aaslt (2004)), (van der Aaslt et al (2011)), (Weijters and van der Aalst (2001)). The reason is that during the execution phase the process is monitored and can be slightly adjusted without redesigning the process. In the diagnosis phase, the enacted process is analysed and the outcome of this phase can be used to redesign the process. Predictions and recommendations based on models learnt from historical data can be used for online maintenance because being aware of what might happen helps managers to set up suitable strategies for timely intervention (van der Aaslt *et al.* (2011)). For example, it may be of interest to investigate certain loops that lead to process failure; suggesting an optimal way to complete the process starting from the current step, etc.

In the role of historical data analysis, process mining faces typical challenges related to data such as data cleansing and merging data or drift concept etc. This is especially important when dealing with business process data.

The diversity is explicitly visible in the highly complex dimensions of the structure of the asynchronous data sequences (i.e. the workflow of the executed process). These sequences consist of many types of tasks (events) and each task in turn can be described by a certain number of attributes. The complexity and diversity arises from the evolution over time of poorly documented processes. Also, as mentioned above, process execution is not enforced by automatic workflow systems and people have degrees of freedom, they do not always follow the designed process. This adds an additional challenge related to the need to identify the actual process model. The consequence is there are many prototypes (different execution sequences) for one process. Each execution sequence of a process is a process instance. A business process instance (S_j) is a composition of discrete events (or tasks) in a time-ordered sequence, $S_j = \left\{s_1^{(j)}, s_2^{(j)}, \ldots, s_{n_j}^{(j)}\right\}$, s_k takes values from a finite set of event types $E = \{e_1, \ldots, e_L\}$. Apart from its starting time $t_i^{(j)}$ and ending time $T_i^{(j)}$, each of these events has its own attributes. For simplicity, we assume that a process does not contain any overlapping events, that means there are no parallel structures (Ruta and Majeed (2011)). To simulate and use the parallel or mutual exclusive structure in process data, works in Berlingerio *et al.* (2009) from workflow data analysis can be referred to.

Other challenges for process mining are to combine process mining with data mining and visual analytics. It is a very conventional trend that researchers combine visual analytics with data mining. Visual analytics helps users to derive insight from their data easily.

Process mining aims to improve the process performance by eliminating errors in the designed process, and improving the execution of the current process.

i) First, for perfect process (re)design: the role of business process analysis and modelling in process mining is to modify the processes to improve efficiency and to effectively use the information technology. The authors in McKibben and Pacatte (2003) list several options for business process improvement:

1. Business process reengineering: major change in the process and might include the goal and direction of the process as well,
2. Business process redesign: major change in the workflow of the process but the goals of the process are unchanged,
3. Business process improvement: incremental and continuous changes based on measuring and monitoring process performance,
4. Technology transfer: a process must be change into a new technology, environment or information system,
5. Process standardisation: a process is defined to provide predictable and repeatable performance.

ii) Second, for excellent execution: process mining can also be used to monitor an in-life process. The data collected for monitoring purposes can be used as the basis for prediction of process failure. This aspect of process mining has overlaps with decision support and management support.

1.2 Project Aims and Objectives

While process mining focuses on deriving a process model from workflow data, methods to predict process step transitions and process outcomes are not yet regularly applied in the area of process analytics. The objectives of this project are to contribute to the predictive process analytics by designing suitable methods for event sequences. This project attempts to create a model to predict future events of a process based on a collection of historical data about all previous tasks and their task attributes. Realistic data contains information from legacy processes and rebuilding their logic is an important aspect of the project. If the reconstructed logic has uncertainties, this implies that there is missing information or the operators of the process do not behave consistently. Legacy processes can appear highly stochastic and if next step prediction is unreliable, the alternative option is to provide recommendations of how the process can be completed from the current step.

We also aim to mine rules which are related to severe events (failure or delay) within the process. Ideally, every step in this process is driven by a certain number of rules. In theory, if all the rules are known, the next step of the process can be predicted. Realistically, this is not the case, but it is still useful if rules about events related to the success of the process can be found. All these objectives are helpful for improving and operating the process because they contribute to its execution in terms of decision making and management support.

To achieve these aims, specific mathematical models are needed and there are a variety of such models in data mining literature. The challenge is to pick the ones that are most suitable depending on these objectives. In particular, we need approaches that can extract information from sequential data and model its temporal nature. Task attributes are another valuable source of information. They can help in reducing the remaining uncertainties after modelling task sequences. Approaches that can deal with both sequential data and non sequential task attributes are therefore desirable. Alternatively, different approaches can be combined as long as they properly address part of the overall objective.

In summary, the aims of this project are to build a flexible model which can:

- predict and rebuild the process based on the execution history as well as the task attributes,
- discover rules which can be used to warn of impending process failure.

Due to the nature of process data which is highly complex and sequential, there is neither a fixed methodology nor an established process to cope with the mentioned problems.

Based on an extensive literature review we have selected a number of approaches that appear to be promising when building predictive models for sequential data. We have concentrated on sequential data rather than on non-sequential data because it best conserves the nature of a process. The approaches selected from literature have been selected with the objective of predicting the events in the process and also for rebuilding the process logic. We mined historical data using conditional probabilities in order to fit the temporal relation between data. More precisely, higher order Markov models are improved by adding a default prediction improvement mode and used to predict the next steps of the uncompleted process. Another option is to use association rules for sequential pattern mining. The technique seems to be efficient in treating sequential data (Agrawal and Srikant (1995)) and may lend insight into the dependency relationship among the variables describing single events or among the events. We would like to be able to deal with the diversity of data. It is appropriate to enrich our sources of sequential models to find the ones that cope best with data diversity. Non sequential approaches, which are known as potentially good solutions for diverse data, are studied. Of these approaches, we have selected the ones that are changeable into sequential approaches. Last but not least, we investigated sequential clustering to tackle the diversity which might be in the treated data. The techniques we have chosen to investigate further are:

- Markov models,

- K nearest neighbour,
- sequence alignment,
- association rules (GOSPADE based), and
- K means clustering (HMM based and sequence alignment based).

1.3 Published Papers

Journal Papers

Le, M., Gabrys, B., and Nauck, D., *Hybrid Approaches for Predicting Business Process Event and Classifying Process Outcomes* (2013), to appear in the journal Expert System.

Conference Papers

Le, M., Gabrys, B., and Nauck, D., *A Hybrid Model for Business Process Event* (2012), Proc. 32nd BCS-SGAI International Conference on Artificial Intelligence, pages 179-192. Cambridge, UK. Le *et al.* (2012)

Le, M., , Nauck, D. and Gabrys, B., *Sequential Approaches for Predicting Business Process Outcome and Process Failure Warning* (2013), Proc. 3rd SIMPDA International Symposium on Data Driven Process Discovery and Analysis, Workshop of the International Conference Very Large Databases (VLDB), pages 1-15. Riva del Garda, Italy. Le *et al.* (2013b)

Le, M., Gabrys, B., Nauck, D., and Martin, T. *KNNs and Sequence Alignment for Churn Prediction* (2013),to appear in Proc. 33rd BCS-SGAI International Conference on Artificial Intelligence. Cambridge, UK. Le *et al.* (2013a)

Le, M., Gabrys, B., Nauck, D., and Martin, T. *Sequential Clustering for Event Sequences and Its Impact Next Process Step Prediction* (2014),to appear in Proc. 15th IPMU International Conference on Information Processing and Management of Uncertainty in Knowledge-Based Systems. Montpelier, France. Le *et al.* (2014)

1.4 Organisation of the thesis

Chapter 2 provides a literature review of sequential analysis. This discusses the approaches from different domains in sequential analysis, their advantages as well as disadvantages. It also looks at non sequential approaches which can be ameliorated into sequential approaches and potentially suits the objectives of this work.

In Chapter 3, according to the research from the preceding chapter, Markov models and their extensions are used to predict process next steps. We show a novel model that is the combination of Markov model and sequence alignment. This new model addresses one weak point of the higher Markov models: lack of coverage.

Chapter 4 presents an extension of a simple but powerful approach in data mining: KNN and a rule based loop detection. The topic of this chapter is to predict process outcomes and to provide failure warnings if possible. Our proposed approach is more precise than the work in Jagadeesh and van der Aalst (2010) in terms of objective and further by studying the different impacts caused by a common segment and a total similarity degree between two given sequences on the outcomes of process instances.

The work in Chapter 5 touches another common area in data mining: sequential clustering. In order to deal with the diversity of the available data, distributing sequences into different groups then treating them separately using suitable approaches is appropriate. Each cluster is then expected to contain similar sequences which are expected to provide input for the performance of the predictive models. Predictive models from the preceding chapters are then applied to clustered data.

Chapter 6 concludes with a brief discussion on advantages and disadvantages of the considered approaches. The possible research directions resulting from the completed work of the PhD are also discussed in the last section.

Chapter 2

Sequential Analysis

2.1 Introduction

Sequential data analysis is not new in data analysis and there is a large corpus of literature devoted to describing and comparing the relative merits of different approaches for different data types as well as problems. Sequential analysis is getting increasing attention because classical analysis techniques can be inefficient or fail whilst dealing with some specific problems. Such problems occur when the entity (data) becomes more complex over time and due to the development in technologies and science as well as society. It is also possible that existing problems which could not be dealt with in the past are re-investigated because new theories and technologies might be applicable. Applications requiring sequential analysis can be from different fields and each application has its particular characteristics as well as its distinctive data. Hence, it is common that approaches can be very powerful whilst solving a problem with a specific type of data but cannot cope with other problems associated with data of different characteristics. Some approaches can only deal with sequences of the same length, others can only deal with numeric variables (feature vectors) etc. For example, to measure the distance between sequences, different functions are suitable based on the data types. This chapter provides a literature review of a range of known sequential approaches from probabilistic models, sequential pattern mining to sequential clustering in data mining. Some of these discussed approaches are potential candidates for solving our problem. We also present some non sequential approaches which are powerful and can supplement the disadvantages of the existing sequential approaches in dealing with our problem. These non sequential approaches are studied and based on such study, some of the non sequential approaches are chosen to turn into sequential approaches later on.

2.2 Probabilistic Models

2.2.1 Markov Models

Markov models are a kind of stochastic model. They are based on the mathematical theory of Markov chains. Markov models (Papoulis and Pillai (1991)) have been used in studying stochastic processes in terms of modelling and predicting customer behaviour. The idea of the model is to use $k > 0$ preceding steps to predict the following $(k+1)^{\text{th}}$ step. Given a sequence of random variables $\{X_n\}$, a first order Markov model uses the current step x_{i-1} to predict the next step x_i and to generalise, a k^{th} order Markov model uses the last k steps to predict the next one:

$$p(x_i|x_{i-1},\ldots,x_{i-n}) = p(x_i|x_{i-1},\ldots,x_{i-k}). \tag{2.1}$$

To construct a Markov model a matrix M, $M = (m_{ij})$ is created containing the probabilities for moving from the previous k steps to the next. The columns of the matrix correspond to the unique tasks, $i = (1,\ldots,I)$, and the rows of the matrix correspond to the unique patterns of length k which are present in the training data set that is used to construct the model. These unique patterns are called states $j = (1,\ldots,J)$. Elements m_{ij} of matrix M are the probabilities p_{ij}:

$$p_{ij} = \frac{n_{ij}}{n_i}. \tag{2.2}$$

where n_i is the number of times pattern i occurs in the data set, and n_{ij} is the number of times pattern i is followed by step j in the data set.

Higher order Markov models can be expected to be more accurate. However, we have to take into account their weak coverage property. For example, consider using a Markov model to predict web-page access and consider a sequence of web pages $\{P_1,P_2,P_1,P_3,P_2,P_1\}$ that is used to create the model. In a second order Markov model, there would be no sample $\{P_2,P_3\}$. Consequently, the prediction for this pattern would have to be the default prediction of the model, i.e. the most frequent step (P_1 in this example). One approach to overcome the problem is to merge the transition states of different order Markov models, after pruning the 'redundant' states. A particular method is the selective Markov model, an extension of All K^{th} order Markov models (Deshpande and Karypis (2004)).

Markov models are often not dynamic and once the model is trained, the obtained matrix is fixed and used for predictions. It is important that the training dataset is large enough to be representative for the encountered patterns. In order to accommodate changes in data the matrix can be rebuilt from scratch after a certain period of time. It is straightforward to create a dynamic Markov model that adapts to new data by storing the counts n_{ij} and n_i and updating M with additional rows and/or columns if new patterns and/or steps are encountered in new data. We can also apply a discounting factor to the current counts to give more weight to the pattern frequencies in new data.

2.2.2 Hidden Markov Models

In the field of sequential data, hidden Markov models (HMMs) are a very powerful tool. Similarly to Markov models, HMMs are a type of stochastic model. They are also based on the mathematical theory of Markov processes which was further developed into the theory of HMMs by Baum in the 1960s (Baum *et al.* (1970)). HMMs have received much attention due to their application in speech recognition. Moreover, there is a large variety of applications ranging from telecommunications, recognition (gesture, speech etc.) to finance, biology etc.

Many approaches in temporal data series only deal with observed variables. We can consider how the performance of the model changes after adding an unobservable or hidden variable which we assume to have an influence on the observed variables. HMMs assume that there is a latent variable that influences the values of the observed variable. The benefits of this approach are shown, for example, in Cox Jr and Popken (2002).

As an example for an application domain consider modelling customer behaviour based on customer event sequences (e.g. historic purchases, contacts with customer service etc.). A latent variable that influences customer behaviour could be the level of customer satisfaction.

Let $O = \{o_1, \ldots, o_N\}$ be the sample of observed variables, and Q a sequence of hidden states $\{q_1, \ldots, q_N\}$. The observed variable can be continuous and follow certain (usually Gaussian) distribution, or it can be discrete, and take values from a finite set $V = \{v_1, \ldots, v_K\}$. The hidden states are discrete, and their value space is Ω, $\Omega = \{s_1, \ldots, s_M\}$. The expression of the joint probability of observed sequence is as follows:

$$p(o_1, o_2, \ldots, o_n | q_1, \ldots, q_n) = p(q_1) \prod_{i=2}^{n} p(q_i | q_{i-1}) \prod_{i=1}^{n} p(o_i | q_i). \tag{2.3}$$

Due to the complexity of the computation, lower order hidden Markov models are more popular. The most popular one is the first order hidden Markov model which assumes that the current state depends only on the preceding state and is independent from the earlier states.

$$p(q_n | q_{n-1}, o_{n-1}, \ldots, q_1, o_1) = p(q_n | q_{n-1}), \tag{2.4}$$

Intuitively, higher order HMMs can be expected to be more accurate (Thede and Happer (1999)). However, the lack of coverage problem needs to be considered, as mentioned in the section on Markov models. It is obvious that with increasing order of the model we are facing more complex computation. A HMM is defined by a set of three parameters $\lambda = (A, B, \pi)$:

$$A = (a_{ij}) = (p(q_i | q_j)), \tag{2.5}$$

where A is the transition matrix and an element a_{ij} is the probability to move from state j to state i. It is necessary to point out that only homogeneous HMMs are considered here, implying the time independence of the transition matrix. The non-homogeneous type is discussed in Netzer *et al.* (2007). Furthermore,

$$B = (b_{ij}) = (p(o_i | q_j)), \tag{2.6}$$

where B is the matrix containing probabilities of having $o_t = v_i$, denoted as o_i, if the hidden state is q_j at time t. Finally,

$$\pi_j = p(q_1 = s_j), \tag{2.7}$$

is the probability that s_j is the initial state at time $t = 1$.

There are three fundamental problems in HMMs: estimation, evaluation and decoding.

The estimation problem is about finding the optimal set of parameters $\lambda = (A, B, \pi)$ which maximize $p(O|\lambda')$ given a sequence O of observed states and some initial model λ'. The Baum-Welch algorithm is used for this. The evaluation problem is to find how the given sequence of observations O fit to the given model λ. The forward (backward) algorithm is appropriate for solving such problem. To solve the decoding problem the Viterbi algorithm is applied for finding the most realisable, suitable sequence of hidden states $q_1, \ldots, q_k$ for the associated sequence O and model λ.

Parameter Estimation

Let D be the set of samples $x = \{x_1, \ldots, x_n\}$. Considering the supervised case, D can be partitioned into c classes $D_1, D_2, \ldots, D_c$. The probability of drawing D_i from D is $p(w_i)$, therefore, $p(x|w_i)$ is the probability of drawing x from D_i. We also make an assumption of the independence between classes which means that parameters from class i are not influenced by class j with $i \neq j$.

- **Maximum-likelihood estimation**

  When we have a set of samples x and its prior distribution, the goal is to find parameters for the distribution that fit the most samples and the prior distribution. Let $p(x,\theta)$ be the probability distribution, x is known, θ is the set or a vector of parameters $\theta = (\theta_1, \ldots, \theta_M)$, we treat $p(x,\theta)$ as a function of θ:

  $$L(\theta) = p(x,\theta). \tag{2.8}$$

  In order to maximise $L(x,\theta)$, its $\nabla_\theta L(x,\theta)$ has to be set equal to zero:

  $$\nabla_\theta L(x,\theta) = 0, \tag{2.9}$$

  Equation 2.9 can be solved to obtain the optimal θ. In general, the log-likelihood function is used instead of likelihood function, which leads to an advantage in the calculation, therefore we let:

  $$L(\theta) = \log p(x,\theta). \tag{2.10}$$

  This method is simple and nearly always converges as the support of the observed samples increases. Next, we consider how maximum-likelihood estimation works by calculating in detail two cases which assume the prior distribution to follow a Gaussian distribution.

  Case 1: Unknown μ.
  In this case we suppose that the samples have normal distribution with the mean value μ and variance σ^2 which is assumed to be known thus $\theta = \mu$. From the assumption it follows that the log-likelihood function has the following form:

  $$\ln p(x_k|\mu) = -\frac{1}{2}\ln(2\pi\Sigma) - \frac{1}{2}(x_k - \mu)^2(\sigma^2)^{-1}, \tag{2.11}$$

because σ^2 is a constant, the partial derivative of the log-likelihood function is:

$$\nabla_\mu \ln p(x_k|\mu) = (\sigma^2)^{-1}(x_k - \mu). \tag{2.12}$$

Obviously, μ^* will be obtained by multiplying by σ^2 the following equation:

$$\sum_{k=1}^{n} (\sigma^2)^{-1}(x_k - \mu^*) = 0, \tag{2.13}$$

$$\mu^* = \frac{1}{n}\sum_{k=1}^{n} x_k. \tag{2.14}$$

Case 2: Unknown μ and σ^2.
In this case θ is a vector $\theta = (\theta_1, \theta_2)$ where $\theta_1 = \mu$, $\theta_2 = \sigma^2$ thus the log-likelihood function has the following form:

$$L(\theta) = \ln p(x_k|\theta) = -\frac{1}{2}\ln 2\pi\theta_2 - \frac{1}{2}\theta_2(x_k - \theta_1)^2. \tag{2.15}$$

By applying the Nabla operator on log-likelihood function we obtain:

$$\nabla_\theta L(x_k, \theta) = \left(\frac{1}{\theta_2}(x_k - \theta_1), -\frac{1}{2\theta_2} + \frac{(x_k - \theta_1)^2}{2\theta_2^2}\right). \tag{2.16}$$

Once again, using the condition $\nabla_\theta L(x, \theta) = 0$ to obtain the optimal θ, we have:

$$-\sum_{k=1}^{n} \frac{1}{\theta_2^*} + \sum_{k=1}^{n} \frac{(x_k - \theta_1^*)^2}{(\theta_2^*)^2} = 0, \tag{2.17}$$

$$-\sum_{k=1}^{n} \frac{1}{\theta_2^*}(x_k - \theta_1^*) = 0, \tag{2.18}$$

where θ_1^* and θ_2^* in equations 2.17 and 2.18 are the searched values of vector θ. Finally, by solving these equations we obtain:

$$\theta_1^* = \mu = \frac{1}{n}\sum_{k=1}^{n} x_k, \tag{2.19}$$

$$\theta_2^* = \sigma^2 = \frac{1}{n}\sum_{k=1}^{n} (x_k - \mu)^2. \tag{2.20}$$

- **Bayesian parameter estimation**

  The results of the Bayesian parameter estimation are similar to those obtained using the Maximum-likelihood (ME), however the computation of the latter is more complex. Whilst θ in ME is considered as an unknown, fixed parameter, it is considered as a random variable in Bayes parameters estimation. $P(w_i|x)$ is the posterior probability and lies at the heart of Bayesian estimation. It can be calculated from the prior probability $P(w_i)$ and $P(x|w_i)$ by the Bayes formula:

$$P(w_i|x,D) = \frac{p(x|w_i,D)P(w_i,D)}{\sum_{j=1}^{c} p(x|w_j,D)P(w_j,D)}. \tag{2.21}$$

For simplicity, $P(w_i|D)$ is replaced by $P(w_i)$ implying that whilst the support of D increases the $p(w_i)$, probability of having D_i remains unchanged. Moreover, the independent hypothesis mentioned above leads to $p(x|w_i,D) = p(x|w_i,D_i)$. Thus equation 2.21 could be rewritten as:

$$P(w_i|x,D) = \frac{p(x|w_i,D_i)P(w_i)}{\sum_{j=1}^{c} p(x|w_j,D_j)P(w_j)}. \tag{2.22}$$

The problem is then expressed as follows:

* *Hypotheses: i)* given a set of samples D, which is drawn independently from the fixed but unknown probability distribution $p(x)$. As in the maximum-likelihood estimation, $p(x|\theta)$ has the form of probability distribution with unknown θ. We can treat $p(x|\theta)$ as a function of variable θ and constant x.
ii) $p(\theta)$ is assumed to be known and $p(x|\theta) \sim p(x|D)$.

* *Find* $p(\theta|x)$

$$p(x|D) = \int p(x,\theta|D)d\theta. \tag{2.23}$$

Using equation 2.23 and *ii)* with respect to Bayes formula we have:

$$p(\theta|x) = p(x|\theta)p(\theta). \tag{2.24}$$

To understand the process of Bayes parameter estimation let's consider the Gaussian case with the only unknown parameter μ. We have $p(x|\mu) \sim \mathcal{N}(\mu,\sigma^2)$ and $p(\mu) \sim \mathcal{N}(\mu_0,\sigma_0^2)$ as the hypothesis, where μ_0 is the guessed value of the associated mean and σ_0^2 is its variance. As we mentioned above, μ is considered as a random variable, and its values form a Gaussian distribution. Finding μ^* that optimizes the probability $p(\mu|D)$ is our goal.

$$p(\mu|D) = \frac{p(D|\mu)p(\mu)}{\int p(D|\mu)p(\mu)d\mu} = \alpha \prod_{k=1}^{n} p(x_k|\mu)p(\mu), \tag{2.25}$$

$$p(\mu|D) = \alpha \exp -\frac{1}{2}[(\frac{n}{\sigma^2} + \frac{1}{\sigma_0^2})\mu^2 - 2(\frac{1}{\sigma^2}\sum_{k=1}^{n} x_k + \frac{\mu_0}{\sigma_0^2})\mu], \tag{2.26}$$

where α is a normalization factor that depends on D but not on μ. This function has the form of normal distribution hence $p(\mu|D) \sim \mathcal{N}(\mu_n,\sigma_n^2)$, $p(\mu|D)$ is a reproducing density and $p(\mu)$ is a conjugate prior. It can be rewritten under the generic Gaussian form with corresponding coefficient as follows:

$$p(\mu|D) = \frac{1}{\sqrt{2\pi}\sigma_n} \exp\left[-\frac{1}{2}\left(\frac{\mu-\mu_n}{\sigma_n}\right)^2\right]. \tag{2.27}$$

Equaling the corresponding coefficients from equation 2.26 and equation 2.27 we obtain μ_n and σ_n^2.

$$\mu_n = \left(\frac{n\sigma_0^2}{n\sigma_0^2+\sigma^2}\right)\mu_n^* + \frac{\sigma^2}{n\sigma_0^2+\sigma^2}\mu_0, \tag{2.28}$$

where

$$\mu_n^* = \frac{1}{n}\sum_1^n x_k, \tag{2.29}$$

and

$$\sigma_n^2 = \frac{\sigma_0^2\sigma^2}{n\sigma_0^2+\sigma^2}. \tag{2.30}$$

There are some criteria that one has to take into consideration when choosing one of the two estimations. First of all is the computational complexity, then the interpretation ability and finally the confidence in the prior information in form of the prior probability distribution $p(x|\theta)$. For those reasons maximum-likelihood estimation is to be preferred, see Eastwood and Gabrys (2009). In contrast to this, Bayes parameters estimation is used in Ascarza and Hardie (2009), Motoi *et al.* (2008).

Algorithms

- Forward: This algorithm and the backward algorithm, which will be presented in the next section, are used in the evaluation problem, i.e. to calculate $p(O|\lambda)$ given $O = \{o_1,\dots,o_T\}$ and λ. Both of those methods are recursive. Let $\alpha_i(t)$ be the probability of seeing the subsequence $\{o_1,\dots,o_t\}$ ending up in state i at time t, the recursive formula is as follows:

$$\alpha_i(1) = \pi_i b_i(o_1), \tag{2.31}$$

$$\alpha_j(t+1) = \sum_{i=1}^{M}\alpha_i(t)a_{i,j}b_j(o_{t+1}), \tag{2.32}$$

$$p(O|\lambda) = \sum_{i=1}^{M}\alpha_i(T). \tag{2.33}$$

- Backward: Similar to the former algorithm, let $\beta_i(t)$ be the probability of the ending subsequence $\{o_{t+1},\dots,o_T\}$ then backward algorithm based on the following equations:

$$\beta_i(T) = 1, \tag{2.34}$$

$$\beta_i(t) = \sum_{j=1}^{N}a_{i,j}b_j(o_{t+1})\beta_j(t+1), \tag{2.35}$$

$$p(O|\lambda) = \sum_{i=1}^{N}\beta_i(1)\pi_i(o_1). \tag{2.36}$$

- Viterbi: Viterbi algorithm was proposed in 1967, since then it has become famous and is used in various fields such as cryptology and telecommunications etc. (Forney (1973)). Known as a method of decoding convolutional codes, Viterbi is very useful in the decoding problem of HMMs. This algorithm is used to find the most reasonable hidden state sequence for a given λ and observed sequence O. It includes the solution for the problem of finding the optimal number of hidden states. Viterbi is also known as an extension of the forward algorithm. The basic idea is to find the state that maximises the transition probability at each step. Even though it is a local optimum it provides a good sequence of states for many purposes. Let $\delta_i(t)$ be the most probable path ending in state i at time t, i.e, given the sequence of observations $O = \{o_1, \dots, o_t\}$ and $\lambda = (A, B, \pi)$:

$$\delta_i(t) = \max_{q_1,\dots,q_t} P(q_1, \dots, q_{t-1}, q_t = i, o_1, \dots, o_t|\lambda), \tag{2.37}$$

We get the following recursive formula:

$$\delta_j(t) = \max[\delta_i(t-1)a_{ij}]b_j(o_t). \tag{2.38}$$

$$\delta_i(1) = \pi_i b_i(o_1). \tag{2.39}$$

- Baum-Welch: As previously mentioned, this algorithm is used for estimation problems. It belongs to the Expectation Maximization family of algorithms. It seeks the likelihood of the observed sequence using an initial set of parameters λ.

  The problem is defined as follows: given $\lambda = (A, B, \pi)$ and a sequence of observed values O, suppose that there is a set of M hidden states, find λ that optimizes the probability $p(O|\lambda)$. Let $Q(\lambda, \lambda^{'})$ be the expected value of $p(O, q|\lambda)$:

$$Q(\lambda, \lambda^{'}) = \sum_{q\in\Omega} \log p(O, q|\lambda) p(O, q|\lambda^{'}), \tag{2.40}$$

$$p(O, q|\lambda) = \pi_{q_0} \prod_{t=1}^{T} a_{q_{t-1}q_t} b_{q_t}(o_t), \tag{2.41}$$

$$Q(p(O, q|\lambda), p(O, q|\lambda^{'})) = \sum_{q\in\Omega} \log \pi_{q_0} \sum_{q} p(O, q|\lambda^{'}) \tag{2.42}$$

$$+ \sum_{q\in\Omega} (\sum_{t=1}^{T} \log a_{q_{t-1}q_t}) p(O, q|\lambda^{'}) + \sum_{q\in\Omega} (\sum_{t=1}^{T} \log b_{q_t}(o_t))(p(O, q|\lambda^{'})$$

  To maximize function Q we use the Lagrange multiplier for each of the three terms on the right hand side of the equation 2.44, and we obtain:

$$\pi_i = \frac{p(O, q_o = i|\lambda')}{p(O|\lambda')}, \tag{2.43}$$

$$a_{ij} = \frac{\sum_{t=1}^{T} p(O, q_{t-1} = i, q_t = j|\lambda')}{\sum_{t=1}^{T} p(O, q_{t-1} = i|\lambda')}, \tag{2.44}$$

$$b_i(k) = \frac{\sum_{t=1}^{T} p(O, q_t = i|\lambda')\delta_{o_t,v_k}}{\sum_{t=1}^{T} p(O, q_t = i|\lambda')}. \tag{2.45}$$

The next step is the calculation of π_i, a_{ij}, $b_i(k)$. Baum-Welch algorithm uses both forward and backward procedures to archive this goal. The algorithms count the expected number of transitions. Let $\gamma_i(t)$ be the probability of being in state i at time t for the sequence O, then we obtain the following expressions:

$$\gamma_i(t) = p(q_t = i|O, \lambda) = \frac{p(O, q_t = i|\lambda)}{p(O|\lambda)} = \frac{p(O, q_t = i|\lambda)}{\sum_{j=1}^{n} p(O, q_t = j|\lambda)}. \tag{2.46}$$

Using the results of forward and backward algorithms from the former sections with the remind of Markovian conditional independence, $\gamma_i(t)$ becomes:

$$\gamma_i(t) = \frac{\alpha_i(t)\beta_i(t)}{\sum_{j=1}^{n} \alpha_j(t)\beta_j(t)}. \tag{2.47}$$

Let ξ_{ij} be the probability of being in state i at time t and in state j at time $t+1$, i.e:

$$\xi_{ij}(t) = p(q_t = i, q_{t+1} = j|O, \lambda), \tag{2.48}$$

ξ_{ij} can be presented by $\gamma_i(t)$ and $\beta_i(t)$ as follows:

$$\xi_{ij}(t) = \frac{\gamma_i(t) a_{ij} b_j(o_{t+1})\beta_j(t+1)}{\beta_i(t)}. \tag{2.49}$$

If we sum the quantities across the time, we obtain: $\sum_{t=1}^{T} \gamma_i(t)$ the number of times being in state i and $\sum_{t=1}^{T-1} \xi_{ij}(t)$ the number of moves from state i to state j. Consequently, the update rules become:

$$\tilde{\pi}_i = \gamma_i(1), \tag{2.50}$$

$$\tilde{a}_{ij} = \frac{\sum_{t}^{T-1} \xi_{ij}(t)}{\sum_{t}^{T} \gamma_i(t)}, \tag{2.51}$$

$$\tilde{b}_i(k) = \frac{\sum_{t}^{T} \delta_{o_t,v_k} \gamma_i(t)}{\sum_{t}^{T} \gamma_i(t)}. \tag{2.52}$$

Generally, Baum-Welch algorithm proceeds in by three steps:

1. E step: lead to the equation 2.40

1. M step: Maximization the expectation, the equation 2.42, consequently the right hand side of it have to hold 2.43, 2.44, 2.45.

1. Update step: Using FW, BW algorithms to calculate the right hand size of 2.43 - 2.45 by calculating equations 2.47 and 2.49.

Summary

As discussed HMMs have some properties which are beneficial for the treatment of sequential data. For example, there are a number of efficient algorithms to deal with the fundamental problems of HMMs; there is a rich source of tools to train HMMs. However, knowing their weakness is important for their application to real problems.

A particular problem is the independence assumption, which requires the current state to be independent from all other states except the previous one. Another issue can be the time dependence of the transition matrix, which is usually assumed to be static, in the other word it does not change over time. Nevertheless, some scientists attempt to take into account the changing with time property (Netzer *et al.* (2007)). In the Bayes hierarchy case, the non-heterogeneous transition probabilities are dependent on time by using an ordered logit model; in ME case the transition matrix can be re-estimated after a certain time period, however the performance seems not to be verified. Another weak point of the Bayesian estimation method is the lack of popular initial distributions and the integral in Bayesian estimation is difficult to extract and approaches, like Monte Carlo, are required to approximate the posterior distribution.

If an improvement in accuracy is needed, higher order HMMs can be considered. However, higher order HMMs lead to the lack of coverage and higher complexity. Researchers have been working on improvements of the performance of HMMs. Similar to Markov models, there are approaches to deal with the trade-off between coverage and accuracy. However, merging the transition states of higher-order hidden Markov models is much more complicated than for Markov models.

2.2.3 Conditional Random Fields

A generative model searches for the joint distribution of observed and target variables, which concerns the enumeration over all sequences of observation. Obviously, the tractable ability is very limited, especially for a large set of data. To address this problem, the independence assumption is used in HMMs (Lafferty *et al.* (2001)). This assumption seems to be too strict because in some cases a new event depends not only on the current state but also on preceding ones.

Thus having another approach with all the advantages from HMMs and which can avoid the independence assumption is desired. As a conditional probabilistic model, a conditional random field (CRF) which belongs to discriminative model family, has all the advantages of HMMs without having the independence assumption problem. In contrast to the generative models like HMMs, discriminative models do not generate observation variables. Therefore, the joint distribution is ignored and only the conditional probabilities are considered (Wallach (2004)).

DEFINITION 1. *Let $G = (V, E)$ be a graph, where V is the set of vertices, $V = \{y_1, \ldots, y_n\}$, and E is the set of edges with $e_i = (y_{i-1}, y_i)$. (X, Y) is a CRF. When conditioned on X, the random variables Y obey the Markov property with respect to the graph: $p(Y_i|X, Y_j, j \neq i) = p(Y_i|X, Y_j, j\ i)$, where $j\ i$ means that j and i are neighbours in G.*

The following figure illustrates graphical structure of a CRF.

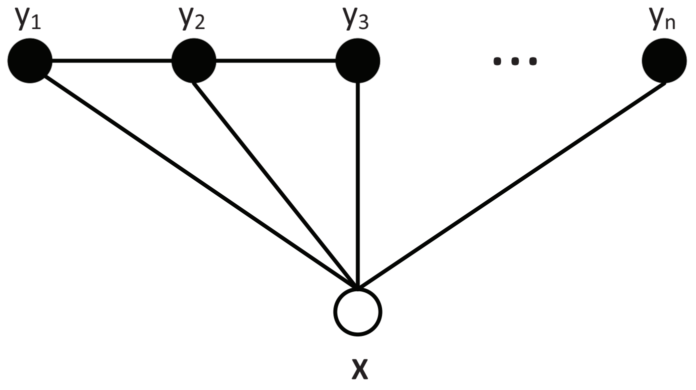


Figure 2.1: Conditional random field.

As can be seen, in Figure 2.1 the white nodes (vertices) of X imply that observations are not generated by CRFs, edges connect related nodes show that the independence assumption is not required for CRFs. Such a graphical structure is helpful for understanding and comparing HMMs and CRFs.

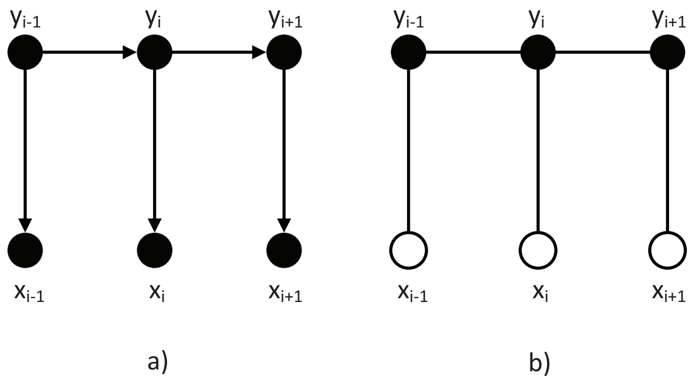


Figure 2.2: a) HMM; b) Conditional random field.

Lafferty *et al.* (2001) define CRFs as a normalization product of the potential functions under the following form:

$$exp(\Sigma_j \lambda_j t_j(y_{i-1}, y_i, \mathbf{x}, i)) + \Sigma_k \mu_k s_k(y_i, \mathbf{x}, i) \tag{2.53}$$

where t is the transition feature function from label $i-1$ to label i of all the observed sequences and s is the state feature function of the label at the position i of all observed sequences. These two types of feature functions can be denoted as $f_j(y_{i-1}, y_i, \mathbf{x}, i)$. These expressions show that, different from HMMs which use one observed variable from the previous position to condition the current state transition between labels y_i and y_j, CRFs can take into account the whole sequence $\mathbf{x}$ if they are related or connected in graph theory terminology/ terms. There is no connection between vertices without a common edge. Avoiding to have independent vertices operated by one feature function, feature functions are built for maximal cliques.

DEFINITION 2. *A clique is a subset of vertices that are fully connected*

DEFINITION 3. *A maximal clique is a clique that is not a subset of any other clique.*

An example of feature functions is given in the following:

$$f_1(y_{i-1}, y_i, \mathbf{x}, i) = \begin{cases} 1 & \textit{if} \quad y_{i-1} = \textit{OTHER and } y_i = \textit{PERSON} \\ 0 & \textit{otherwise} \end{cases} \tag{2.54}$$

$$f_2(y_{i-1}, y_i, \mathbf{x}, i) = \begin{cases} 1 & \textit{if} \quad y_i = \textit{PERSON and } x_{i+1} = \textit{said} \\ 0 & \textit{otherwise} \end{cases} \tag{2.55}$$

CRFs can be converted into HMMs by turning vertices Y into hidden states. HMMs can be converted into CRFs as well by letting the feature functions be $p(y_{t-1} = i, y_t = j)$. The training of CRFs is similar to the one of HMMs and the known algorithms are applicable i.e. forward, backward and maximum likelihood.

2.2.4 Gaussian Processes

The Gaussian or normal distribution is very popular in probability and statistics. The distribution was discovered in 1809 by Gauss while he was researching error theory (Rasmussen and Williams (2006)), (Pimentel *et al.* (2013)). A majority of random variables in economic, agriculture, population, biology have normal distribution (e.g. the height of humans, the length of people's arms, IQ index, the surplus of a company etc.). Due to the fact that a lot of events in the real world tend to follow a normal distribution, researchers investigated the question: Can the Gaussian distribution be used to solve non linear problems in machine learning (data mining)? In trying to answer the above-mentioned question, many non-parametric and parametric problems were proved to converge to Gaussian processes. This includes the work in Neal (1996) which proved that problems in neural networks converge to GPs, or the work in MacKay (2003) which shown that the spline problem can also be reproduced by GPs. Gaussian processes are extensions of the Gaussian distribution. The differentiator between them is the working space. Instead of working on vector space as Gaussian distribution, Gaussian processes work on function space.

Let us remind ourselves of the Gaussian distribution. Given a sample of $\{x_n, y_n\}$, $n = \{1, \ldots, N\}$, the probability distribution has the following form:

$$p(x|\mu, \Sigma) = \mathcal{N}(\mu, \Sigma) = (2\pi\Sigma)^{-\frac{1}{2}} \exp\left(-\frac{1}{2}(x-\mu)^T)\Sigma^{-1}(x-\mu)\right), \tag{2.56}$$

where μ is the mean and Σ is the variance of the distribution.

A Gaussian process is a collection of random function variables $y = (y(x_1), y(x_2), \ldots)$, which have a joint distribution:

$$p(y|K,x_n) = \frac{1}{Z}exp(-\frac{1}{2}(y-\mu)K^{-1}(y-\mu)), \tag{2.57}$$

where μ is the mean function set to be the zero function in GPs and $K(x,x',\Theta)$ is the covariance function. Briefly, a Gaussian process is a stochastic process such that any set of function variables $f_n, n = \{1,\ldots,N\}$ has joint (zero mean) Gaussian distribution $p(f|X) = \mathcal{N}(0,K)$.

The covariance function plays an important role in Gaussian processes as it formulates the correlation between the data points. Intuitively, the larger the distance between any given x_i and x_j, the less correlation between them exists. The length scale λ decides the sensitivity of the changes of the covariance function between x_i and x_j in the case of the squared exponential covariance function. The vertical scale and noise are also taken into account in some covariance functions. Below are some common covariance functions taken from Snelson and Ghahramani (2006).

i) Stationary covariance function:

Squared exponential (SE):

$$K(x,x') = \sigma_0^2 \exp\left[-\frac{1}{2}\left(\frac{x-x'}{\lambda}\right)^2\right]. \tag{2.58}$$

ii) Non-stationary covariance function:

Linear:

$$K(x,x') = \sigma_0^2 + xx'. \tag{2.59}$$

Brownian:

$$K(x,x') = \min(x,x'). \tag{2.60}$$

Periodic covariance:

$$K(x,x') = \exp\left[-\frac{2sin^2\left(\frac{x-x'}{2}\right)}{\lambda^2}\right]. \tag{2.61}$$

The output of a GP has the form of a probability distribution. It is easy to manage and to verify the influence of the parameters on the approximated function. The output can be unstable and in this case an ensemble forecast is a good way to obtain better predictions. Another problem can be that the Gaussian assumption for initial and posterior probabilities is not always reasonable.

2.3 Sequential Pattern Mining

2.3.1 Association Rules

Starting in basket analysis, association rules gained attention due to their capability in decision support, marketing and business management. If from historical data, the association rule (pattern) $e_i \Rightarrow e_j$ was found, and if the event e_i has been observed, then e_j will follow with a confidence of $c\%$. Association rules are a common method for finding relationships and getting insight knowledge from sequential data. Many researchers have used them to discover sequential patterns, frequent sequences or subsequences including: Agrawal and Srikant (1995), Srikant and Agrawal (1996), Agrawal and Srikant (1994).

Given e_i and e_j which are transactions or events, $e_i \Rightarrow e_j$ means if a sequence S_k in S contains e_i then e_j is contained in that sequence with a certain probability. In addition, the number of sequences in which $e_i \Rightarrow e_j$ appears is required to be larger than a given minimum support.

DEFINITION 4. *A data sequence in association rules is a list of transactions.*

DEFINITION 5. *A transaction (also called item set) in association rules is a set of literals, called items.*

DEFINITION 6. *A sequential pattern (rules) is a list of sets of items whose support and confidence exceed user specified thresholds, called minimum support and minimum confidence.*

DEFINITION 7. *The support of an event or an association rule, ms, is the ratio of the number of sequences n in the data sample which contain the considered event/rule to the number of all sequences N.*

$$ms = \frac{n}{N} \tag{2.62}$$

DEFINITION 8. *The confidence of an association rule $e_i \Rightarrow e_j$, mc, is the ratio of its support ms to the support of e_i.*

$$mc = \frac{ms(e_i \Rightarrow e_j)}{ms(e_i)} \tag{2.63}$$

Association rules have significant applications in various fields. Due to their important role, quite a number of studies have been performed. These studies cover a large range of problems, from developing different algorithms which are suitable for different data types to selecting useful rules and eliminating rules which are not interesting. The basic algorithm from early stage is a priori which was invented by Agrawal et al. It is then improved into an algorithm called generalizations of sequential patterns. The newly improved algorithm can profit from the taxonomy structure and the temporal characteristics of the data, by adding three constraints. These constraints are about:

1. Sequential patterns are considered only if adjacent elements occur in a specific time interval. For example, a customer complains three times, each time is one year later then the previous one can have different reaction toward the company than a customer who complains three times in a row within 1 month.

2. Setting a time window for transactions, items from different transactions which are made within a specific time window (chosen by users) then they can be considered elements of a sequential pattern as if they come from one transaction.

3. Including and taking into account the taxonomy structure of the data. Adjacent elements parented from the same higher level can be considered as similar in sequential patterns.

For example, two customers buying two books each. Customer 1 bought *A* and *B*, customer 2 bought *A* and *C*. If *B* and *C* are books from the same categories (directed parent), let's say science fiction, sequential patterns *AB* and *AC* are considered as one sequential pattern and each contributes to the support. A study which focuses on taxonomy structure is found in multiple levels algorithm. In this algorithm, information about types (food, toy, etc.), categories (% fat for food), brands are encoded into transaction tables. For example, President unsalted butter milk is encoded as 1(butter-type(food)) 1 (unsalted-category) 3 (President-brand). Similarly, information about consecutive repeated items (tasks) is contained and stored in IdList (a variant of transaction table) in SPADE and GO-SPADE algorithms.

We list several alorithms which are common in association rules:

- A-priori algorithm and some of its extensions Agrawal and Srikant (1995) (basic).
- Generalizations of sequential patterns algorithm Srikant and Agrawal (1996) (a-priori extension).
- ML (multiple levels) algorithm and some variations Han and Fu (1995) (taxonomy).
- SPADE and GO-SPADE algorithms Leleu *et al.* (2003) (repetition).

DEFINITION 9. *A rule of length k (composed of k elements) is called k-frequent sequence (pattern).*

Association rule mining often returns a large number of irrelevant or uninteresting rules. For example, in basket analysis we may find "milk and bread" as an expected association rule and "toy and milk" as an uninteresting rule. Ways for obtaining useful rules have been studied in Mika *et al.* (1994). In the paper, the concept of a template is introduced. All rules which match one of these templates will be considered useful and kept in the rule base.

Generally, association rule algorithms follow the same procedure which can be summerised as follows. First, scanning through the data to find candidate subsequences of different lengths. This step is called candidate generation. Candidate sequences of shorter length are merged to form candidate subsequences of length with one unit longer compared to the former ones. The resulting candidate subsequences are then put under a pruning process. This pruning process is necessary because it helps to eliminate candidates which do not hold the minimum support and confidence conditions. Second, counting the candidate subsequences left after pruning process which are contained in the data sequences. This step is called counting candidates.

2.3.2 Sequence Alignment

To determine sequence similarity both distance and similarity measures can be used. For numeric variables well-known distance measures exist and can be easily applied. However, in sequence analysis sometimes we have to work with sequences which are constructed from symbols, e.g. categories (churn prediction), phonemes (speech recognition) or characters (hand-writing recognition), etc. We need specialised functions which have the ability to measure the similarity of symbolic sequences.

Sequence alignment is very common in bio-informatics and has a relatively long history in this domain. The target entities of sequence alignment in bio-informatics are amino acid sequences of proteins, DNA sequences, etc. Sequence alignment is used for a number of purposes Needleman and Wunsch (1970). Algorithms used in sequence alignment are mainly divided into global alignment and local alignment. Global alignment provides a global optimisation solution, which spans the entire length of all query sequences. In contrast, local alignment aims to find the most similar segments from two query sequences. In this work, both types of alignment are investigated to verify which one is effective in determining the similarity between the two sequences, the overall comparison between two given sequences or the most similar (consecutive) segments. The similarity between process sequences is used to predict the process outcomes.

2.3.3 Global Alignment Algorithm

In this kind of algorithm, sequences are aligned from the first event to the last one. One such algorithm was introduced by Needleman and Wunsch (1970). There are three characteristic matrices associated with this algorithm: substitution matrix, score matrix and traceback matrix. The role of the substitution matrix is to generate the degree of matching between any two events from the set of event types, or in other words matching subsequences of length 1. This degree, which is irrespective of the position of the events then contributes to the matching score in the score matrix that consider the complete sequences, i.e. all events in the order they occur. Given two sequences, we then have to consider the order of the events and compute the score of matching the i^{th} event in one sequence with the j^{th} event in the other sequence, $i = \{1,\dots,len_1\}$, $j = \{1,\dots,len_2\}$ and len_1, len_2 are the lengths of the two given sequences. These scores define the score matrix. Finally, the trace back matrix encodes the optimal way of matching both sequences from a number of possible matches. We now introduce these three matrices.

1. Substitution matrix: in biology a substitution matrix describes the rate at which one amino acid in a sequence transforms to another amino acid over time. The entries of this matrix present the probabilities of transforming one amino acid to another. There are different ways of generating the substitution matrix. The simplest way is to not take into account the amino acid mutation factor, instead just give a score of 1 to the same amino acids and use a score of 0 to a pair of different amino acids.

$$s(i,j) = \begin{cases} 0 \quad if \quad eventi \neq eventj \\ 1 \qquad\qquad otherwise \end{cases}$$

   In this case, the substitution matrix is an identity matrix, the elements of the main diagonal are 1 and all the others are 0.

   To present mutations, a more complicated form of substitution matrix is used. In this matrix, the elements of the matrix off the main diagonal can have a value different from 0, depending on the likelihood and frequency of transformation between the two amino acids.

$$s(i,j) = \log \frac{p_i \times M_{ij}}{p_i \times p_j} \tag{2.64}$$

where p_i, p_j are the frequencies of the amino acid i and j in the available data and M_{ij} is the probabilities of aligning amino acid i and j, the count of times i mutates to j in the historical data stored in the bank.

2. Score matrix: This matrix's elements are similarity degrees of events from the two given sequences.

$$h_{i0} = -\delta \times i, \tag{2.65}$$

$$h_{0j} = -\delta \times j, \tag{2.66}$$

$$h_{ij} = \max\left\{h_{i-1,j} - \delta, h_{i-1,j-1} + s(x_i,y_j), h_{i,j-1} - \delta\right\}, \tag{2.67}$$

where $i = \{1,\dots,len_1\}$, $j = \{1,\dots,len_2\}$. δ is a specific deletion/insertion penalty value chosen by users. h_{i0} and h_{0j} are the initial values needed for the recursive formula in order to compute the entries of the score matrix h_{ij}. x_i and y_j are events at positions i and j from the given sequences. $s(x_i,y_j)$ is the score from the substitution matrix corresponding to events x_i and y_j.

3. Traceback matrix: Elements of this matrix are left, diag or up depending on the corresponding h_{ij} from the score matrix. These entries are built as follows.

$$q(i,j) = \begin{cases} diag & if \quad h(i,j) = h(i-1,j-1) + s(i,j) \\ up & if \quad h(i,j) = h(i-1,j) - \delta \\ left & if \quad h(i,j) = h(i,j-1) - \delta \end{cases} \tag{2.68}$$

This matrix is used to track back from the bottom right corner to the top left corner to find the optimal matching path. Starting from the bottom right element, one moves in the direction given by the element which can be left, up or diag. This leads to another element with its own instruction (up, left or diag). By following the chain of directions the element at the top left corner is reached. The obtained path is the optimal way of matching the two given sequences.

Example for global sequence alignment: given two sequences *ABCD*, *ABD*, find the optimal matching between them using global alignment. Suppose that there are four unique tasks (events) A,B,C,D in the given data set. The corresponding substitution matrix S is illustrated as follows:

$$\left(\begin{array}{ccccc} & A & B & C & D \\ A & 1 & 0 & 0 & 0 \\ B & 0 & 1 & 0 & 0 \\ C & 0 & 0 & 1 & 0 \\ D & 0 & 0 & 0 & 1 \end{array}\right)$$

To compute the entries of the score matrix δ value is needed, let δ be 2, consequently $h_{i0} = -2*i$, $h_{00} = 0$ and $h_{0j} = -2*j$:

$$\begin{aligned} h_{11} &= max\{h_{01}-\delta, h_{00}+s(x_1,y_1), h_{01}-\delta\} \\ &= max\{-4,1,-4\} \\ &= 1 \end{aligned} \tag{2.69}$$

where $s(x_1,y_1) = s(A,A)$, this is the $s(1,1)$ in the substitution matrix and $s(1,1) = 1$. By analogous computing for other h_{ij}, we obtain the score matrix:

$$\begin{pmatrix} & A & B & C & D \\ A & 1 & -1 & -3 & -5 \\ B & -1 & 2 & 0 & -2 \\ D & -3 & 0 & 2 & 1 \end{pmatrix}$$

To get the optimal matching, we build the trace back matrix based on the obtained score matrix. The corresponding trace back matrix's entries are found by determining which factor under the max function on the right hand side of Equation 2.83 (see Equation 2.84) is the maximum while computing the score matrix's entries. For example, $h_{11} = 1 = h_{00} + s(x_1,y_1)$, according to the definition in 3, the corresponding entry q_{11} in the trace back matrix is *diag*.

$$\begin{pmatrix} & A & B & C & D \\ A & diag & left & left & left \\ B & up & diag & left & left \\ D & up & up & diag & diag \end{pmatrix}$$

Finally, the optimal matching path is $\{q_{34} = diag, q_{23} = left, q_{22} = diag, q_{11} = diag\}$.

2.3.4 Local Alignment Algorithm

The aim of local algorithms Smith and Waterman (1981), Waterman (1994) is to find a pair of the most similar segments, from the given sequences. In this algorithm, a matrix of similarity degrees is built based on the following formula:

$$h_{i0} = h_{0j} = h_{00} = 0, \tag{2.70}$$

where h_{i0}, h_{0j} and h_{00} are the initial values for the recursive formula that is used to compute h_{ij}. Note that this is different from the global alignment. The initial values are set to be 0 because in local alignment it is not important where the common segment starts, the aim of the local alignment is to find the most similar segments of two given sequences.

$$h_{ij} = \max\left\{h_{i-1,j}-\delta, h_{i-1,j-1}+s(x_i,y_j), h_{i,j-1}-\delta, 0\right\}, \tag{2.71}$$

where $s(x_i,y_j)$ is the element of the substitution matrix as presented in the previous paragraph for global alignment.

For example, given two sequences *ABCDE* and *EBCAD*, the corresponding matrix of similarity degrees is as follows:

$$\left(\begin{array}{cccccc} & A & B & C & D & E \\ E & 0 & 0 & 0 & 0 & 1 \\ B & 0 & 1 & 0 & 0 & 0 \\ C & 0 & 0 & 2 & 0 & 0 \\ A & 1 & 0 & 0 & 1 & 0 \\ D & 0 & 0 & 0 & 1 & 0 \end{array}\right)$$

The i^{th} event in a sequence can be aligned to the j^{th} event in another sequence, or can be aligned to nothing (deletion). This leads to a number of possible matchings of the two sequences. The optimal pair of aligned segments is identified by first finding the highest score in the matrix. This element is the end of the optimal aligned path. Then, the path is filled by tracking back from that optimal highest score diagonally up toward the left corner until 0 is reached. In this example, the best match is *BC*.

2.3.5 Moving Windows

For many telecommunications businesses, it is an important question of how to cope with ever more data because data grows at a very rapid rate. Due to technological advances in computer capacity, vast amounts of data can be stored for analysis. This is an advantage as we have rich sources of data to use but it also poses the problem of efficiently using such large amounts of data. In the scope of sequential data, data streams are important in some applications, however, it is not tractable or efficient to store the entire data stream during the modelling procedure. The solution is then to store in a window most recent elements in the stream as they are assumed to have most influence on the considered problem. Such a window is called moving or sliding window.

A moving window is important in streaming models and is usually used to aid other analytic methods to cope with sequential data. It can be coupled with KNNs to produce a sequential approach as in Ruta *et al.* (2006) to predict churn, to combine with association rules in order to have frequent sequential patterns (Srikant and Agrawal (1996)) etc. Of these combinations, moving windows coupled with association rules is the most common. The hybrid architecture has essential advantages like enabling association rule algorithms to mine sequential rules (Srikant and Agrawal (1996)) and to reduce the cost in terms of time and memory.

Given the data sample $\{x, y\}$, a window of size $w = 2d$ is defined. A standard learning algorithm, for example, neural networks, decision trees, etc., is then used to map $\langle x_{i,t-d}, \dots, x_{i,t+d} \rangle$ to the output y (Milligan (2001)). Any algorithm can be applied in the window. That means a non-sequential algorithm can be used as a sequential algorithm by the moving window method. In general, there are two important properties in the moving window technique: a) the size of the window (which is usually fixed) and the step size of the window. Research into the effect of differing window and step sizes can be found in the works of Du *et al.* (2004) and Dietterich (2002) respectively.

In the original version of sliding window, the oldest element, description or value inside the window is deleted as a new one comes in Babcock *et al.* (2002). However, we have to be careful not to replace representative elements with outliers. To avoid this issue, an extension of the moving window is created by storing old elements (FLORA algorithm) (Widmer and Kubat (1996)).

This algorithm suggests the idea of multiple windows. In the context of complex data, one approximation function may not be suitable. Hence, an interesting idea is to use multiple windows with different learning algorithms, for example, regression analysis and an HMM etc. depending on the shape of data underlying function.

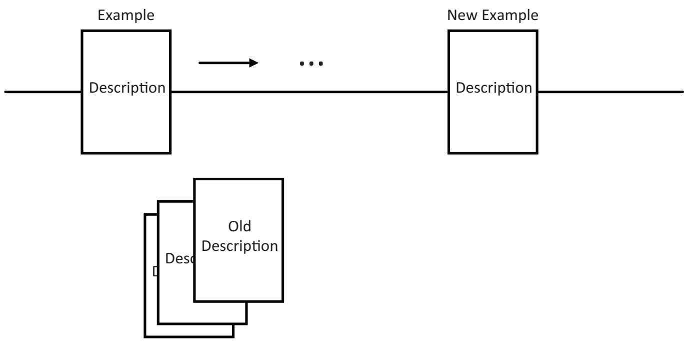


Figure 2.3: Schema of a moving window approach.

2.4 Classification

2.4.1 Classification Overview

Recently data analysis has been enriched and guided by both objectives and tools and it is called data mining. In fact, data mining is just a new term for the higher level of development of data analysis, and its aim is still to extract nontrivial information from data samples. Data mining is widely used in a variety of domains. One of its main and important tasks is classification.

Classification (supervised learning) Duda *et al.* (2001) is an domain in data mining whose aim is to predict group membership for data instances. In other words, the task is to match a new object to a set of classes by a means, named classifier, given a sample of labelled data. For example, classification is used in character recognition, given a number of images of hand-written characters, classify them into one of the classes from the set $y = \{A, \dots, Z\}$.

There are a number of classification approaches in data mining, some common ones are:

- decision trees e.g. Hadden *et al.* (2006), Safavian and Landgrebe (1991), Raedt (1997),
- support vector machine e.g. Archaux *et al.* (2004),
- K nearest neighbour e.g. Ruta *et al.* (2006),
- artificial neural networks e.g. Hadden *et al.* (2006).

To optimise the outcome of classifiers, ensemble classifications are widely used. These approaches are bagging (Breiman (1996)), (Chen and Ren (2009)), boosting (Lemmens and Croux (2006)) and ensemble forests (Lariviere and Van den Poel (2005)).

2.4.2 Classifier Performance Measures

The simplest way to measure the performance of a classifier is to use percentage correct.

DEFINITION 10. *Percentage correct is the ratio of the correct predictions to the total number of the predictions made in the experiment.*

However, this is not the reasonable and useful information one might want to have. In classification problems, researchers are more interested in knowing how many correct answers they made for each class as well as the information about incorrect answers for each class. Recall, precision and confusion matrix are then appropriate. Precision tells us the fraction of the classifications that the classifier assign that are correct. Recall provides information about the fraction of the population of the classified class which are correct predictions for one specific class given by the classifier. Confusion matrices contain information about correct answers and incorrect answers for both class 1 and class 0.

DEFINITION 11. *Precision is the ratio of the classifications that are correct made by the classifier to the total number of classifications.*

DEFINITION 12. *Recall is the ratio of the classifications that are correct made by the classifier to the actual support of the class in the dataset.*

DEFINITION 13. *Given a set of data which consists of subjects labelled as 0 and 1. The number of subjects with label 0 is n_0 and that with label 1 is n_1. Using a classifier to classify the data set, we obtain $n_0^{'}$ and $n_1^{'}$ which are the numbers of subjects classified as class 0 and 1 respectively. The outcome of the classifier can be illustrated in a confusion matrix as in below. Each column of the matrix represents the instances in a predicted class, while each row represents the instances in an actual class*

$$\begin{pmatrix} & & Actual & \\ & & class1 & class0 \\ Predict & class1 & a & b \\ & class0 & c & d \end{pmatrix}$$

where $a+c = n_1^{'}$, $b+d = n_0^{'}$, $a+b = n_1$ and $c+d = n_0$.

2.4.3 KNNs for Classification

KNN is a supervised learning method and has applications in the field of data mining, statistical pattern recognition and many others. They are memory-based reasoning techniques which basically identify analogous cases from history (experience) then use the knowledge gained from these cases to treat the problem at hand. KNNs are common in applications for making recommendations, classification and estimation (prediction). The framework of this method is to find K sequences from a given pool of data which are most similar to the given sequence $S(c,t,\tau)$, where c denotes the class of a sequence (e.g. a customer), t is the starting time and τ is the completion time of a sequence. Each sequence among the K resulting sequences has its corresponding class. The prediction or classification of the given sequence $S(c,t,\tau)$ is determined by the majority class from the K nearest sequences. The advantages of using KNNs is that they treat "as is" data and are quite simple as there is no parameter training required, but nevertheless their performance is quite good as shown in Eastwood and Gabrys (2009). In this work, the authors compare the performances of KNNs and the one of HMMs applied to data from a telecommunication domain. KNNs were applied to discrete data and HMMs to sequential data for the same problem. It was shown that an HMM using sequential data did not much outperform a KNN using discrete data.

Identifying similar sequences requires a distance or similarity measure function. Similarity measures are therefore central to all memory-based reasoning techniques, such as KNNs. There are different measurement approaches which characterise and create different KNNs. When comparing sequences which consist of elements from different fields, comparing one field at a time. The resulting distance of a field contributes its share to the total distance between two given sequences. It is important to note that different fields (numerical, categorical, logical, etc.) can have different impacts on the problem in hand. This effect can be modelled by weighting these individual field distances accordingly to their degrees of influences on the problem.

Even though KNNs are simple the problem of choosing the right number of neighbours exists and there is no mathematical formula for it. One option is to select the optimal K by applying cross-validation procedures. An important characteristics of a KNN is that there is no separate model; the data itself is the model. Any time when a new sequence occurs, it is compared to all the sequences in the training data. Therefore, having the representative set of data helps to reduce computation but still assure the accuracy. Choosing the training data is obviously a non trivial problem.

One of the variations of KNNs is used in Ruta *et al.* (2006) to predict churn in a telecommunications company. The author combined KNN and survival theory. The objective of the method is to predict the remaining path of customers' event sequences. Firstly, K nearest sequences were found by calculating the Euclidean distances between the considered sequence and all sequences in the data sample. Then, in the case where the sequence obtained does not contain churn, the remaining lifetime (up to churn) is calculated by a lifetime function. In this case the author selected the Γ function. The remaining lifetime of a customer at age t is the difference between the customer lifetime and the time period the customer has been with the company. A churn event of a customer is implied from the remaining time which is the simple average of K remaining lifetimes. The framework of this work has been applied to business processes in Ruta and Majeed (2011).

2.4.4 Neural Networks

Neural networks (NNs) are widely used in regression and classification analysis, for example in time series prediction or gesture recognition etc. There are many types of neural networks and the feed-forward type is common.

Neural networks are universal approximators, they can be used in different applications for solving both the predictive as well as classification problems. NNs have the ability to learn and there is no hypothesis/prior distribution assumed for the data in consideration. One of the important advantages of NNs is that they have the capability to deal with non-linear problems. However, the solution of the model is not interpretable, therefore it is called a black box model. Another limitation of NNs is the fixed structure and thus their inability to deal with sequences of different lengths.

2.4.5 Graph transformer networks

This method uses acyclic directed graph with cost (or penalty) over the graph's edges to track multiple hypotheses. Bottou et al. considered this approach for the cheque reading problem of interpreting handwritten characters in Bottou *et al.* (1997) and LeCun *et al.* (1997), given a sequence of handwriting characters. Firstly, the string of characters is cut into pieces. Each piece can be a letter or cannot. A set of all possible paths over the sequence pieces is listed. This set forms the directed graph. Each combined path is one hypothesis. The issue is to select the most reasonable by minimising the penalties assigned to paths in the training data, penalties can be assigned to corresponding edges of the path. Finally, Viterbi algorithm (Forney (1973)) is used to find out the most reasonable path by determining the hypothesis for which the total penalty score of the corresponding path is lowest.

The system is designed as a feed-forward network. The outcome of a module is the input of another, which means the changes (gradient) in a local objective function has influence in the feed-forward linked module. Therefore, to manage the interaction between these modules, a back-propagation procedure can be applied. Each module in the system needs to be differentiable in order to be able to find the global optimum for the overall objective function. Example of a graph transformer network is illustrated in Figure 2.4.

2.4.6 Decision Trees

Decision trees are considered as one of the most common approaches in classification and clustering (Cox Jr and Popken (2002)), (Hadden *et al.* (2006)), (Lemmens and Croux (2006)). It is used widely in a large range of disciplines for example: medicine, game theory, pattern recognition, etc. A tree is structured as a acyclic directed graph which consists of nodes (vertices) and edges. Each branch of the tree is a classification and the leaves of the tree are partitions of the dataset into classes. Every branch has its own probability to follow. All the outcomes (leaves) of a decision tree have their payoffs. Following a certain number of branches, a leaf is reached and the probability of ending at this leaf can then be computed. Briefly, a typical decision tree consists of nodes and branches:

- Node: can be either root which is the start node of the tree with no incoming edges, nodes (also called non-terminal nodes) and leaves which are the end nodes of the tree.
- Branch: a branch is the link between a parent node and a child node.

At each node, criterion is needed in order to split the node into different children nodes (Mingers (1989)). To measure the goodness of the split (implies the information gained by applying the splits), impurity functions can be used:

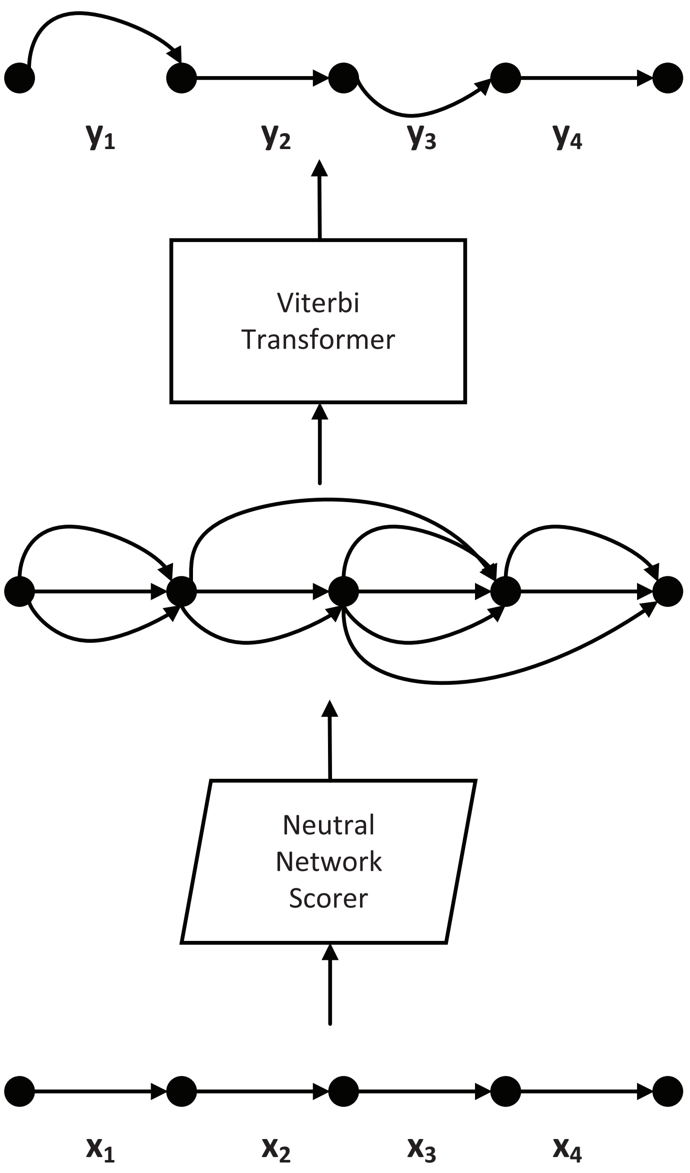


Figure 2.4: Graph transformer networks.

DEFINITION 14. *An impurity function is a function ϕ defined on all K-tuples of numbers $(p_1,\dots,p_K)$ satisfying $p_j \geq 0, j = 1,\dots,K, \Sigma_j p_j = 1$ with the properties:*

i) ϕ is a maximum only at the point $(1/K,\dots,1/K)$ ii) ϕ achieves its minimum only at the points $(1,0,\dots,0)$ iii) ϕ is a symmetric function of $p_1,\dots,p_K$, i.e., if ϕ is permuted ϕ remains constant.

A number of common impurity functions exist: Quinlan (1986), Chang and Martinsek (2004) and Rokach and Maimon (2008)

- Entropy

$$\Sigma_1^K p_j \log \frac{1}{p_j} \tag{2.72}$$

if $p_j = 0$ use the limit $\lim p_j \to 0 p_j \log \frac{1}{p_j} = 0$

- Misclassification rate $1 - max_j p_j$

$$\tag{2.73}$$

- Gini index

$$\Sigma p_j(1-p_j) = 1 - \Sigma p_j^2 \tag{2.74}$$

- Gain ratio

$$\begin{aligned} IM = \ & \tfrac{1}{N}\, Sigma \Sigma x_{ij} \log x_{ij} - \Sigma x_{i\cdot} \log x_{i\cdot} \\ & - \Sigma x_{\cdot j} \log x_{\cdot j} + N \log N \end{aligned} \tag{2.75}$$

(2.76)

$$IV = -\Sigma \frac{x_{i\cdot}}{N} \log \frac{x_{i\cdot}}{N} \tag{2.77}$$

$$GR = \frac{IM}{IV} \tag{2.78}$$

A tree is grown by splitting nodes. The splitting procedure repeats until the stopping criteria are met. These stopping criteria are usually based on conditions like:

- the maximum depth of the tree has been reached,
- the number of leaves are too high or low compared to thresholds,
- the population of a leaf is too low,
- the threshold for splitting criterion is not reached.

Having an over fitted or under fitted tree is very common and not appropriate. Setting too strict stopping criteria can lead to the small and under fitted trees. In the opposite case, if the stopping criteria are loose, over fitted trees are built. To avoid this issue, a tree pruning procedure is added to the tree building process. We can build a tree with not very strict criteria and prune the obtained tree afterwards to avoid the over fitting problem. To prune a tree, different techniques like cost-complexity pruning, reduced error pruning, pessimistic pruning, optimal pruning etc. (Rokach and Maimon (2008)) can be used. An example of a very simple tree pruning process is to delete leaves of which populations are lower than a specific threshold.

One of the advantages of using decision trees is that they are relatively easy to read and to verify (in contrast to the neural networks). Another important advantage of decision trees is that no assumption about the dataset is required, unlike Gaussian processes or HMM where prior distribution of the data set is needed or in HMM where an independence assumption is required. Moreover, decision trees can cope with both numeric and categorical data. Decision trees are also capable of handling error and missing fields in data. There are a number of disadvantages using decision trees, for example, output attributes must be categorical and decision tree algorithms tend to be unstable because of their sensitivity to the training data set.

2.5 Clustering

2.5.1 Clustering Overview

Clustering is one of the main constituent elements in data mining. It is known as an unsupervised learning family. The aim of data clustering is to get the data distributed into a finite number of clusters. In most of the cases, such clusters are based on distance between data points. Hence, a distance measure function is required and is vitally important. Clustering aims to cluster data in a way that each object in a cluster is more similar to other objects in the same cluster than to other objects in other clusters.

There are three main types of clustering algorithm:

- hierarchical clustering: agglomerative clustering (hierarchical clustering) (Rajaraman and Ullman (2011)), (Duda *et al.* (2001)),
- probabilistic clustering: EM algorithm (Berry and Linoff (2004)),
- partitioning clustering: K means clustering, K modes, K prototypes (Rajaraman and Ullman (2011)),

A number of other distinguished clustering types are listed in the following:

- fuzzy clustering (Gabrys and Bargiela (2000)),
- grid based clustering algorithm (Elavarasi *et al.* (2011)) (no distance required, only population of the grid is counted),
- graph-based algorithm Click (Zaki *et al.* (2007)).

Some of the clustering approaches can deal with numerical data only, some can deal with categorical data (Zaki *et al.* (2007)), (Li *et al.* (2004)) and some can deal with mixture of numerical and categorical data (Gabrys and Bargiela (2000)), (Andreopoulos *et al.* (2006)).

As sometimes sequential features describe data best, conserving the sequential nature is appropriate and might be necessary. One option is to consider each sequence as a multivariate feature vector and use vector composition based clustering to cluster the given sequences (Dhillon and Modha (2001)). However, decomposition based approaches require sequences of the same length. Overcoming the issue of having different sequence lengths, there are a large number of HMM-based sequential clustering algorithms and extensions (Smyth (1997)), (Garcia *et al.* (2009)), (Li and Biswas (1999)), (Porikli (2004)) etc.

2.5.2 Clustering Goodness Measures

The criteria used to evaluate the goodness of the clustering approaches can be applied in order to pick/select the best among/from a range of K values. There are internal and external criteria, identified by the information used to evaluate the clustering fitness. When only information of the clustered data is used to evaluate the goodness of the grouping, it is called internal criteria. In this kind of criteria, information about the cohesion inside each cluster, the relationship between clusters (separation) which are characterised by coefficients like radius, diameter, variance, etc. are used to evaluate the clustering. External criteria need some extra information about the label of the data or some (gold) standards built by experts. Some examples of external criteria are entropy measure, F-measure and Kullback-Leibler.

Such criteria change accordingly for different types of clustering. For example, some criteria in probabilistic clustering are entropy criterion, F-measure, Kullback-Leibler measure etc. and criteria in partitioning clustering can be the level of detail, the distortion of the clusters and the interdependence with other object groupings in the data set. That implies, the density/population of a cluster is as important as the distance between clusters. They are taken into account in scattering and separating criteria.

- Entropy in information theory is a measure of uncertainty in a variable.

$$H = -\Sigma_i p_i \log p_i, \tag{2.79}$$

  where p_i is the probability of class i.

- F-measure (or F1score) is a measure of test accuracy and defined as a harmonic mean of precision and recall defined in the previous section.

$$F = 2 \times \frac{(precision \times recall)}{(precision + recall)}, \tag{2.80}$$

- Kullback-Leibler is a non-symmetric measure of the difference between two probability distributions P_1 and P_2, denoted as $d_{KL}(P_1|P_2)$.

$$d_{KL}(P_1|P_2) = -\ln_i \left(\frac{P_1(i)}{P_2(i)}\right) P_1(i) \tag{2.81}$$

- Square error distortion is a measure of the difference between two data sets which are usually two states of a dataset, before and after the change caused by applying something to the original one. Given a set of n points C_1 and a set of m points C_2 , the square error distortion is defined as follow:

$$clusterDistortion = \Sigma_{i,j} d(C_1, C_2)/n. \tag{2.82}$$

  where $i = 1, \ldots, n$ and $j = 1, \ldots, m$.

2.5.3 HMM - Based Sequential Clustering

As sometimes sequential features describe data best, conserving the sequential nature is appropriate and might be necessary. One option is to consider each sequence as a multivariate feature vector and use vector composition based clustering to cluster the given sequences (Dhillon and Modha (2001)). However, decomposition based approaches require sequences of the same length. Overcoming the issue of having different sequence lengths, there are a large number of HMM-based sequential clustering algorithms and extensions (Smyth (1997)), (Garcia *et al.* (2009)), (Li and Biswas (1999)), (Porikli (2004)) etc. In these publications, the authors model individual data sequences by probabilistic models then use likelihood to build a distance matrix. Given N sequences $\{S_1,\ldots,S_N\}$, each sequence s_i is modelled by a HMM from N HMMs, $\{\lambda_1,\ldots,\lambda_N\}$, which have the same number of hidden states. The distance matrix $D=(d_{ij})_{N*N}$ is constructed of elements which are computed using the following formula:

$$d_{ij}=\frac{l(s_i,\lambda_j)+l(s_j,\lambda_i)}{2}, \tag{2.83}$$

where d_{ij} are the distance between sequence i and sequence j, $l(s_i,\lambda_j)$ is the likelihood of a sequence i belonging to a model $\lambda(j)$. The probability of which a sequence fits to an HMM can be computed easily using the forward algorithm.

A number of variations of the HMM likelihood based similarity measurement are listed and proposed in Garcia *et al.* (2009). Of these variations, there is one called BP metric:

$$d_{BP}^{ij}=\frac{1}{2}\left\{\frac{(l_{ij}-l_{ii})}{l_{ii}}+\frac{(l_{ji}-l_{jj})}{l_{jj}}\right\}, \tag{2.84}$$

and one of the same family:

$$d_{YY}^{ij}=|l_{ii}+l_{jj}-l_{ij}-l_{ji}|. \tag{2.85}$$

where d_{ij} is the distance between sequence i and sequence j, L is the likelihood of a sequence belong to a model. The probability of which a sequence fits to a HMM can be computed easily using forward algorithm. One of the variations of the HMM likelihood based similarity measurement is the distance measure obtained by combining Kullback-Leibler distance and HMMs (Garcia *et al.* (2009)).

Traditional clustering techniques are applied to partition the data using the obtained distance matrix or they try to use the same number of HMMs as number of clusters, the philosophy is similar to the gaussian distributions in EM clustering. The issue here is to pick the right number of HMMs (clusters). In addressing this Porikli (2004) introduced an eigenvector based method to automatically compute the optimal number of clusters.

2.5.4 K-means Clustering

K means clustering in data mining is itself an NP hard problem (Kanungo *et al.* (2002)). However, heuristic K means algorithms exist and provide local optimal solutions. Some common K means algorithms are Lloyd algorithm (Kanungo *et al.* (2002)), Bradley, Fayyad and Reina algorithm (Rajaraman and Ullman (2011)). K means clustering is powerful and very simple to understand and to apply. It is as well known in partitioning clustering as the EM clustering in probabilistic clustering. K means is widely used in marketing, medical diagnostic, text mining (Steinbach *et al.* (2000)), image mining etc. because of its simplicity and competence. Many authors have tried to improve and extend the original approach to fit specific problems and/or to make it more powerful. One of the alternative algorithms modifies the original K means to profit from extra information about which data point should or should not be assigned to which clusters (Wagstaff *et al.* (2001)). The constraints on cluster membership are added to the assignment step, together with the distance condition. Another alternative algorithm makes the algorithm faster by using triangle inequality to avoid unnecessary computation (Elkan (2003)).

Users usually choose the initial K. However, there a number of studies investigating the question how to select a reasonable value for K (Pham *et al.* (2005)). K can also be estimated by trial. The criteria used to evaluate the goodness of the clustering approaches can be applied in order to select the best from a range of K values. There are internal and external criteria, identified by the information used to evaluate the clustering fitness.

Given the number of clusters K, first, initiate the corresponding centres. These centres can be data points randomly taken from the available data set or taken in a way that these centres are far away one from another, etc. This depends on the users. Second, each data point is assigned to the closest cluster based on the distance between the data point and K centres. The data type decides which types of distances should be used.

Once all the data points are assigned, a new centre for each cluster is re-determined. For numerical data, such centres are mean values of the elements in the corresponding clusters. Median can also be used here instead of mean and K means becomes K medians. The procedure of assigning data points to clusters and recomputing centres is repeated until it converges.

2.6 Summary

In order to understand the field as well as to find the technical directions for our research, we studied approaches which have been used in sequential analysis and have the potential to solve the problems in the domain. A wide range of methods are considered. They are diverse in methodology (e.g. probabilistic models, non parameters approaches, association rules) and relevant in different features (e.g. temporal characteristics, diverse data, interpretable outcome). Having researched their advantages and disadvantages in line with our problems, a number of candidates have been selected for implementation. We review our decisions according to the experimental results. The following chapters present our proposed approaches regarding the project objectives. Markov models and extensions are chosen to start with in the next chapter.

Chapter 3

Probabilistic Models for Predicting Next Steps and Outcome of Business Processes

3.1 Introduction

This chapter aims to capture sequential patterns which help to predict severe problems as well as next process steps. It is obvious that the failure of a process has a very strong impact on the company profit in either direct or indirect ways. It increases the cost due to response and execution times spent, material, infrastructure and in the long term it causes customer churn (does not reach customer satisfaction). Response time is the time duration that is required for a step in a process instance to go through the corresponding part/step of the information system. Execution time is the time duration needed for a process instance to go through the corresponding information system. Being able to predict problems will allow companies to become proactive. That means companies can have the capability to actively respond to a sudden change during the process execution phase with suitable strategies. If a company manages to produce such prediction with high accuracy, it is useful for damage control hence save the companies' expenses as well as and maintaining good service and customer satisfaction. Even though this predictive area is a very interesting field and can significantly contribute to process analytics, it has not received adequate attention from researchers. Ruta and Majeed (2011) provided the most complete picture with very clear objectives and framework. In their works, the authors tried to predict the remaining path of a given process instance using different mathematical models from data mining including Markov models, decision trees, association rules. Their attempt failed in providing high accuracy prediction for many classes. However, for some classes, higher order Markov models performed well. A very recent work of Folino *et al.* (2012) provides a two-stage forecast on the process performance by first, clustering the process data into groups of similar traces (sequences). Second, distinct predictive models in each cluster are employed to predict the process performance. Authors use decision trees to cluster the data. This work of Folino *et al.* (2012) is inspired by Greco *et al.* (2006).

The project focuses on sequential analysis in order to provide process prediction and warnings. Process mining is a relatively new area and its current techniques are not sophisticated (van der Aaslt et al (2011)). Therefore, a quick look through what has been done as the predictive and warning models in sequential process mining is appropriate but a thorough research in sequential analysis is very important. In the following we introduce some representative works from the very new emerging topic, process prediction:

- In Song and van der Aalst (2007) the authors analysed process performance based on a graph which depicts process events. This helps users to have a "helicopter view" of the process graphically and it is more toward the visualisation direction in process mining.

- In Jagadeesh and van der Aalst (2010) the authors used trace alignment in a preprocessing phase where the event log is investigated or filtered and in later phases when a certain questions need to be answered. It complements existing process mining techniques focusing on discovery and conformance checking.

- In Trcka and Pechenizkiy (2009) the authors predefine a number of templates that are crucial for the problem under consideration. The set of templates can always be extended or modified. This helps users to reduce the search space and direct the process mining in the desired way.

- In Ruta and Majeed (2011) the authors used probabilistic models and decision trees to predict the remaining path of the uncompleted processes based on completed process data.

- In Folino *et al.* (2012) the authors introduced a predictive-clustering approach to first cluster the data then to predict the performance of the considered process instance based on the cluster it assigned to.

In order to predict process events, we look to approaches known from data mining (Berry and Linoff (2004)). Due to the nature of the data, approaches that can deal with temporal sequence data are most relevant. One example from this class of approaches are Markov models (MMs) which have been used to study stochastic processes. In (Papoulis and Pillai (1991)), for example, it has been shown, that Markov models are well suited to the study of web-users' browsing behaviour. Event sequences are used to train a Markov model which encodes the transition probabilities between subsequent events. Event sequences can represent customer behaviour, customer transactions, workflow in business processes etc. The prediction of the continuation of a given sequence is based on the transition probabilities encoded in the Markov model.

The order of a Markov model represents the number of past events that are taken into account for predicting the subsequent event. Intuitively, higher order Markov models described in Section 3.3.1, Chapter 2 are more accurate than lower order Markov models because they use more information present in the data. This has been shown to be true in the prediction of web browsing behaviour (Papoulis and Pillai (1991)). Lower order models may not be able to discriminate the difference between two subsequences, especially when the data is very diverse, i.e. if there are a large number of unique sequences. In Ruta and Majeed (2011) it has been shown that higher order Markov models can be more accurate than lower order models in the context of customer service data.

However, higher order models can suffer from weak coverage in particular when the data is diverse and not clustered. For sequences which are not covered by the model, a default prediction is required. Default predictions typically reduce the accuracy of the model. It is obvious that with increasing order of the model, the computational complexity also grows. In the case of plain Markov models, there are already approaches to deal with the trade-off between coverage and accuracy. These approaches have been introduced in the studies of Deshpande and Karypis (2004), Eirinaki *et al.* (2005) and Pitkow and Pirolli (1999). The general idea is to merge the transition states of different order Markov models, which is followed by pruning 'redundant' states. A particular method is the selective Markov model, an extension of K^{th} order Markov models (Deshpande and Karypis (2004)).

In this chapter, we present a hybrid approach to fulfil the goals of predicting next process steps as stated in the project aims and objectives. Our model is Markov based and it addresses the problem of lack of coverage in higher order Markov models. This approach is a combination of pure Markov models and the sequence alignment technique (Waterman (1994)). The sequence alignment technique is applied when the default prediction is required. This is the case when the given sequence cannot be found in the transition matrix. The matching procedure is applied in order to extract the given sequence's most similar sequences (patterns) from the transition matrix. Based on the prediction for the obtained sequences, the prediction for the given sequence is found.

We are testing our approach on workflow data from a telecommunication business process, but it is likely to be applicable in a variety of other sequential prediction problems – especially if Markov models have already been used and shown to be relevant.

The rest of this chapter is organised as follows. Section 3.2 provides us with problems related to business processes and the role of predictive modelling in business process management, from context, perspectives, issues, data to potential research directions. Section 3.3 introduces the predictive models which are used in this study. It is followed by Section 3.4 which presents the experiments' results and evaluation. Finally, in Section 3.5 our conclusions and future work directions are discussed including the motivation and framework for Chapter 4.

3.2 Business Processes: Context and Data

3.2.1 Business Processes

The essence of effective and efficient business processes has already been introduced briefly in Chapter 1 together with the motivation that leads us to use sequential predictive analysis to improve process performance in general or in other words, to support business process operation. To have an insight view of the process predictive problems and corresponding adequate strategies, the following areas will be reviewed business processes' primary concepts, data and tools to put processes under control, e.g. business process management systems (BPMS) and process mining. This review highlights provides some important information about circumstances of predictive modelling, context and challenges one has to face whilst dealing with such problems.

Process data is a type of workflow data and according to one of the business process definitions: ”a process is a chain of tasks which are performed/executed in certain orders to produce a product or provide a service”. Therefore, the success or failure of a component task has a strong impact on the following ones. Sometimes, the success of a task even has a decisive role in the success of the next one. It is then an advantage to be aware of the future steps in a process instance in order to either avoid problems or develop a strategy for emerging problems.

DEFINITION 15. *A process instance is one execution of the process.*

Business Process Management Systems (BPMS)

A BPMS is a system that allows companies to define their processes and manage the entire process lifecycle in order to build workflows that are more effective, more efficient and more capable of adapting to an ever-changing environment.

- Business process management (BPM) is a holistic, top-down management approach whose goal is to optimise business operations to maintain good service and to obtain customer satisfaction under the companies’ benefit constraints. Being process-centric, BPM involves organising the business around clearly defined and documented processes and managing process lifecycles. With its strong emphasis on continuous process improvement, BPM aims to arm the companies with the flexibility to quickly respond to changes in the competitive landscape.

- BPM is multi-dimensional and normally involves defining and managing the relationships between people, processes, and IT systems.

- BPM aims at promoting continuous process improvement. Process performance is the final goal and used as a measurement to identify and eliminate inefficiencies in business processes. The lifecycle of business processes refers to the cyclical phases of process management including:

    1. design,
    2. monitoring and execution,
    3. diagnostics,
    4. implementation,
    5. configuration.

This kind of system requires consistent development because the effectiveness in process management and enactment is essential. In particular, it is important to overcome the lack of flexibility and low capability dealing with changes in BPMS. To tackle this, building accurate predictive models that enable the pre-active monitoring of time violations is necessary. Despite of this key opportunity, this predictive analysis area has not received reasonable attention from researchers in the domain.

Process Mining

Process mining can be a tool in BPMS. This tool consists of technical approaches which are created to solve BPMS’ problems. Its most common role is to recover the process models. This type of work in process mining is referred to as discovering models. These process models are rebuilt by extracting information from event logs. Process mining is related to workflow mining. In workflow mining, rebuilding the workflow model from the data traces is important. This kind of back track operation is useful since the as-is model (i.e. workflow enactment) can then be compared to the originally designed one (i.e. workflow design). Based on this comparison, the process can be examined for inefficiencies. Adjustments might need to be made somewhere during the execution phase. To model the sequential characteristics of process data, Berlingerio *et al.* (2009) present an extension of the TAS paradigm.

Another type of model in process mining is the conformance model. Such models in process mining check the alignment between the process model obtained previously and the as-is process (e.g. data at event logs).

The last type of process mining is enhancement. This process mining type is to extend and change the existing models to have better process designs and achieve better process performance.

There are two possibilities for process mining to improve business processes: process (re)design and online operational support.

- Redesigning the process is concerned with analysing historical data in order to discover the process and trying to change the process design to improve its performance.

- Providing online operational support is concerned with assisting the process monitoring by pre-empting problems and preparing solutions/strategies to cope with these diagnoses if they occur; directing processes based on the current circumstances. This procedure is performed online with live data.

Researchers in the field have paid a considerable attention in the former direction. A majority of the literature in process mining focuses on discovering and conformance checking models. However, experts have pointed out that it is almost impossible to afford a tailored autonomous system for every business process in most cases. It then may be a better approach to switch to the latter research line. The research of Heinrich and Paech (2013) implicitly strengthens such direction. Their work studied the mutual impact of IT systems and the process performance. Changes in IT systems have influence on business process performance in the sense that if the IT system is not powerful enough, it does not have required capacity to deal with the queuing tasks effectively and the process performance is held down. Analogously, if there is a large queue at an IT port, it then can cause an hanging incident or system breaking down. This proves that to have an optimal process design, the IT system state is a constraint. Hence, if we do not have a very powerful IT system, we should resort to other solution by having optimal process design conditioned by the IT system state and providing online operational support.

Researchers who follow the latter research line probably have the advantage of being pioneers in a new area. There might be promising results waiting to be unearthed. It is however, a big challenge building predictive models for business process data. Challenges in predictive business process analytics are embedded in the problem itself and the process data.

Predictive Problems and Challenges There are a number of issues related to the global scope of predictive modelling business processes. The idea of building such predictive models is illustrated in Figure 3.1.

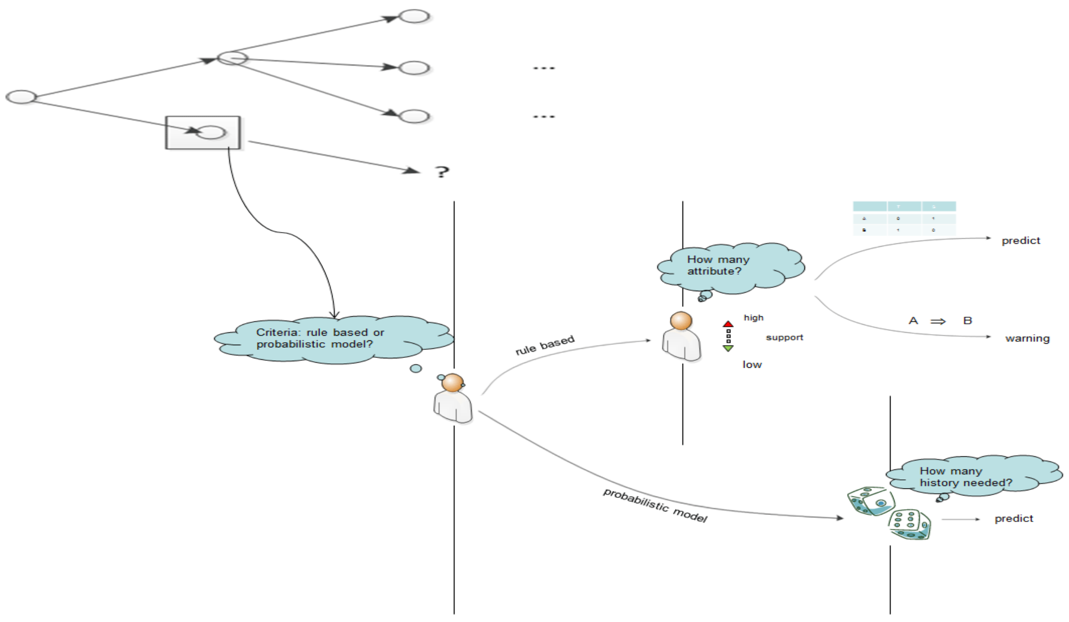


Figure 3.1: A combining model for sequential business process prediction and recommendation.

Workflow management systems are known for having less flexibility and being unable to react to external changes. Predictive models are expected to overcome this limitation. Such models are built from historic data to aid online operational support by providing prediction, detecting errors and recommending strategies. Being aware of coming problems, actors can be more active and more able to deal with the situation adequately. This is not a simple task, these models have to have the ability of both accurately deal with big data which is sequential and diverse, and requiring low processing time as with online support, in order that issues are able to be detected early enough for the company to react adequately.

Combining process mining and data mining is also hard as classical data mining techniques cannot deal with such a complex problem and data alone. The problem requires an appropriate combination of different techniques with the potential to address the objectives of the project as well as the characteristics of the available data.

Another conventional research direction is to combine visual analytics and process mining.

Business Process Data and Challenges Associated

In telecommunications, business processes are executed over very large information systems. A lot of them are legacy processes, that means they are transferred from one information system to another. There are a large number of IT systems and actors involved in executing a process. An actor is a subject which can be a person, a part of a IT system etc., whose role is to perform a task in order to complete a given process. Typically, process data is highly dimensional and complex.

Whilst working with business process data, event logs are a very common concept that should be mentioned. These logs are where the process events information is stored with attributes including starting time, closing time etc. These event logs can be stored in logfiles, mail logs, message logs, database logs etc. In process mining, all techniques assume that it is possible to record events (activities) sequentially where each event is a defined task in a process. Business process data is in consequence, considered sequential in nature.

- Finding, merging and cleaning data: Since a business process can spread through a large number of IT systems, it can be difficult to locate all data related to a business process. Data from different departments/subsources need to be merged/ synchronised in order to be ready for analysis. This task is not easy when the system is huge. Cleaning data is also very important as we always have noise in data, especially when the system is not autonomous.

- Complex event logs have diverse characteristics: different event logs have different characteristics, one might have a very large number of cases, unique tasks, unique paths or prototypes, another one might have too few cases such that it is not representative or not large enough to draw any stable information/rules. A prototype is a distinct ordering of executed process tasks found in the data.

- Sequential characteristics of process data are hard to work with since there do not seem to be effective sequential approaches from data mining which are aligned to our problem.

- Drift concept: We assume that it is possible to record events sequentially. However, in practice, parallel structures are known for their advantages, for example, in computing parallel algorithms are famous and efficient. The philosophy is the same in business process where most of the time, things are done in parallel whenever possible. The parallel structures are converted into sequential structure in workflow data by splitting events from the parallel events. The issue occurs when an event starts before another but finishes after. The question is how to order these two events. This is the concept drift problem.

- Data labelling: we do not have information about process outcomes in many cases. It is then necessary to find criteria to determine if a process instance of the given data set is a success or a failure.

- Missing data and outliers are common in data analytics. We obviously have to face these difficulties in our area. For instance, to filter out outliers, process instances which consist of one task (event) are eliminated from consideration.

We now move to the relevant works in the process predictive area.

Related work

In Chiu and Khoo (2005), the authors pointed out some difficulties dealing with time series data, for instance, lack of independent property, not stable and heterogeneity. They also listed three common methods for tackling temporal data which are probabilistic models, sequential analysis techniques and logit, a regression based method. They then introduced their own approach and claimed its higher accuracy compared to the three aforementioned methods. The proposed approach is also regression based but the relationships between an outcome variable and a set of explanatory variables are modelled. It models difference across time (nonstationary) and across groups (group heterogeneity) and addresses a number of difficult problems involved in modelling the effects of recent events on subsequent events within a series.

Trace alignment in Greco *et al.* (2006) (process sequences are called traces): The idea of their work is to find different prototypes. Business processes can have complex dynamics that lead to the issue of requiring expensive and long analysis for modelling such data. Sometimes it is even infeasible under the companies' profit viewpoint. In response, there are a certain number of approaches introduced in the literature to accommodate the design of complex workows by means of process mining techniques. Facing the fact that it is hard to find a generic process model for the enactment process data (Greco *et al.* (2006)), the authors had the idea of building different process models for subsets of the process instances which have similar executions. Symbolic process sequences are compared and grouped into subsets respecting the similarity between sequences. These resulting clusters are assumed to be different process models. They used K means clustering to cluster feature vectors generated from process traces. These features are selected based on their impact on the process performance. Each process trace is a Boolean tuple which is ready for the K means to be applied to.

Context aware prediction on business process execution in Folino *et al.* (2012): Here, the authors are inspired by the work of trace alignment mentioned previously. They established a predictive approach based on context awareness. Process sequences are clustered using decision trees first. The obtained clusters are then modelled individually in order to predict the process performance. In their work, the authors choose process remaining time as a performance feature. They have this information available for all of the process traces in the historical data set. This work is one step further than the work of Greco *et al.* (2006). Here, the authors do not stop at clustering process data but also investigate the process performance. However, both of the works do not take into account the sequential characteristics of the data, which might play an important/decisive role in process outcomes (performance).

From local patterns to global models: Towards domain driven educational process mining (Trcka and Pechenizkiy (2009)): Starting from the same philosophy as Greco *et al.* (2006) and Folino *et al.* (2012) which is to deal with smaller scopes of a process then integrate such individual results, the authors assume that based on domain knowledge a set of pattern templates can be defined. The resulting templates can then be used to direct process executions. These templates are considered to be crucial for the process and they can be updated over time.

Supporting process mining by showing events at a glance (Song and van der Aalst (2007)): The aim of process mining is to extract information from event logs to be used in analysing specified problems. Performance analysis is one such problems. There are two branches of performance analysis in process mining: data based and model based. In the data driven approaches, all the statistical information are used in addition to graphs and figures generated by visualisation tools. The issue here is that there is no information about the process model involved and thus problems like delay or bottlenecks are hard to identify. The process driven approaches use the process model to check the process instance under consideration by projecting it on the model. The disadvantage of the later approach family is that sometimes there is no generic accurate process model, in contrast the model can be very unstable, flexible. To this end, the authors propose a method which combines the two aforementioned families by using the data as it is and providing a helicopter view to users. They use the event logs to build a dot chart.

Business process forecasting in telecommunication industry (Ruta and Majeed (2011)): The authors focus on the predictive side of process mining. A number of data mining approaches are used and tested in the paper, including Markov models, decision trees and association rules from sequential pattern mining. They employed sliding window in order to capture the sequential characteristics of the data. Even though the models performance is not good in general, it provides some hints for those who are interested in predictive process analysis. For example, whilst classifying the data, the correctness of classification for some classes is very high.

3.3 Predictive Models

This section presents Markov models and their extension that can help enhancing the current approaches to process management in a telecommunication business context by predicting future events based on the current and past data. Here, a business process instance (S_j) is a composition of discrete events (or tasks) in a time-ordered sequence, $S_j = \left\{s_0^j, s_1^j \dots s_{n_j}^j\right\}$, s_j takes values from a finite set of event types $E = (e_1, \dots, e_L)$. Apart from its starting time t_i^j and duration of T_i^j, each of these events has its own attributes. For simplicity, we assume that a process does not contain any overlapping events, that means there are no parallel structures.

The goal of our predictive models is to predict the next event s_{i+1}^{N+1} following event s_i^{N+1} in a given process instance $S_{N+1} = \{s_0^{N+1}, s_1^{N+1}, \dots, s_{i-1}^{N+1}\}$ based on the data from closed process instances $S = \{S_0, S_1 \dots S_N\}$. A simple predictive model can be illustrated in Figure 3.2.

We first investigate some probabilistic approaches which are known for their strength in dealing with temporal data. In the following subsection, such family of approaches will be discussed including Markov models (MMs) and hybrid Markov models (MSAs).

3.3.1 Markov Models

The idea of the model is to use $k > 0$ steps to predict the following $(k+1)^{\text{th}}$ step. Given a sequence of random variables $\{X_n\}$, a k^{th} order Markov model uses the last k steps to predict

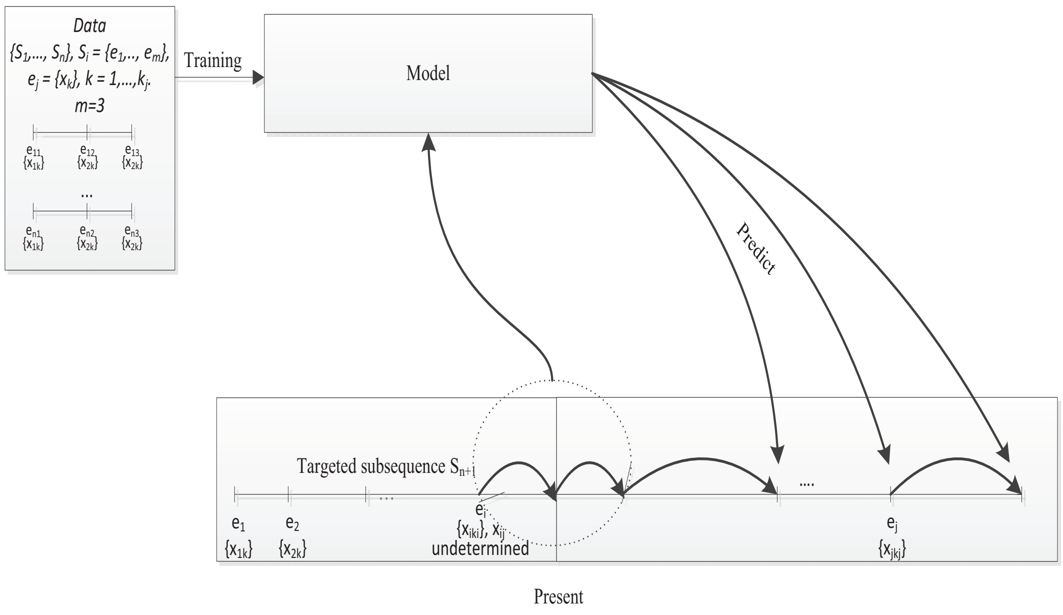


Figure 3.2: A simple predictive model.

the next one:

$$p(x_i|x_{i-1},\ldots,x_{i-n}) = p(x_i|x_{i-1},\ldots,x_{i-k}). \tag{3.1}$$

Higher order Markov models can be expected to be more accurate but they have weak coverage property. This is discussed in Chapter 2. Lack of coverage eventually reduces the accuracy of the higher order Markov models. In addition, it is well known that with the increasing in the order of the model, the computational complexity also grows. Researchers have worked on approaches to overcome the problem. The principle of such approaches is to merge the transition states of different order Markov models, after pruning the 'redundant' states. Particular methods are $AllK^{th}$ Markov models and its extension selective Markov model (Deshpande and Karypis (2004)).

3.3.2 Hybrid Markov Models

The work presented in this chapter investigates the idea of combining higher order Markov models with a sequence alignment technique (Waterman (1994)) in order to maintain the high accuracy of the higher order Markov models and, at the same time, compensate for their lack of coverage. When a higher order Markov model is employed to predict process events and the given sequence has no match in the transition matrix M, we would be forced to produce a default prediction which would be detrimental to the model accuracy. To this end, we present our approach, MSA for Markov Sequence Alignment, which is based on the assumption that similar sequences are likely to produce the same outcome. Basically, the steps that MSA took to identify the next step for a previously unseen sequence S_{new} has no match in the transition matrix as follows:

- Compare S_{new} with all sequences (patterns, states) in the transition matrix, and extract l states which are most similar.
- Generate prediction based on these l selected states. For each state i in the transition

matrix $M = (m_{ij})$ there is a corresponding predictive row vector, $(m_{i1}, \ldots, m_{iJ})$. The predicted next step of state i is the task j with $m_{ij} = \max(m_{i1}, \ldots, m_{iJ})$. Thus, the predictions for the l selected states contain l^* unique events, $l^* <= l$, $E^* = \{e_1, \ldots, e_{K^*}\}$. Then, the final prediction for S_{new} is the most frequent event in E^*. If, however, there is no single most frequent event in E^* then the one with highest probability is selected. Alternatively, we go back to the l most similar states and select the sequence that is most similar to S_{new} and use its predicted step as the prediction for S_{new}. If there is no single most similar sequence, we select the one with the highest frequency from this set.

In order to compare S_{new} with the sequences in the transition matrix, we need to be able to define and measure their similarities or distances. Here, the sequence alignment technique is applied (Waterman (1994)). In particular, we adopt the edit (deletion and insertion) weighted distance function for its flexibility in determining the degree of similarity. Given two sequences, a weight is assigned to each pair of events at the same position in the two sequences. The weight is 0 if the two events are identical, 1 or δ (a specified value) if they are different. In the case where both events of the pair must be deleted in order to keep the longest possible similar subsequences, the corresponding weight is 1. If only one of the two events must be deleted the weight is δ. The sum of weights of the comparison is then used to determine the similarity of sequences. The lower the weight is, the more similar two sequences are.

This algorithm starts by constructing a matrix used to find these longest possible subsequences and the corresponding weights. For example, given two sequences *ABCDE* and *EBCAD*, the matrix looks as follows:

$$\left(\begin{array}{cccccc}
 & A & B & C & D & E \\
E & 0 & 0 & 0 & 0 & 1 \\
B & 0 & 1 & 0 & 0 & 0 \\
C & 0 & 0 & 1 & 0 & 0 \\
A & 1 & 0 & 0 & 0 & 0 \\
D & 0 & 0 & 0 & 1 & 0
\end{array}\right)$$

Next, the first events A and E from both sequences are deleted which results in a weight of 1. The second event in both sequences is B, and has weight 0. The fourth event A is deleted from the sequence *EBCAD* in order to match two Ds in fourth and fifth positions in the two sequences. We add δ to the sum of weights. Finally, we delete the last event E from the sequence *ABCDE*, adding another δ. The total weight for matching the two sequences is $w = 1 + 0 + 0 + \delta + \delta = 1 + 2\delta$.

The following example will illustrate our method for the case of the transition matrix of a third order Markov model:

$$\left(\begin{array}{cccc} & A & B & C \\ ABC & 0.1 & 0.2 & 0.7 \\ CBC & 0.1 & 0.5 & 0.4 \\ BAA & 0.2 & 0.5 & 0.3 \end{array}\right)$$

and a given sequence *BCC*. This sequence has not occurred before and is not stored in the matrix. The aim is to find the most similar sequences from the matrix and use their predictions to generate the prediction for the given sequence *BCC*. We first match *BCC* to *ABC*:

$$\left(\begin{array}{cccc} & A & B & C \\ B & 0 & 1 & 0 \\ C & 0 & 0 & 1 \\ C & 0 & 0 & 1 \end{array}\right)$$

the weight of this comparison is $w = \delta + 0 + \delta = 2\delta$. Similarly, the resulting weights of matching *CBC* and *BAA* against *BCC* are $w = 2\delta$ and $w = 2$, respectively. Let δ = 0.4 and $l = 2$, the two chosen sequences in this case are then *ABC* and *CBC* and their predictive vectors are $(0.1, 0.2, 0.7)$ and $(0.1, 0.5, 0.4)$, respectively. Based on these vectors, the predictive next steps are $\{B, C\}$. The weights and frequency of occurrence of these two events are equal hence, the next step of the given sequence *BAC* is predicted to be *C* due to higher transition probability, $m_{13} = 0.7$ which is greater than $m_{22} = 0.5$.

Algorithms 1 and 2 illustrate the procedure of matching two sequences by deletion and insertion of symbols with penalties.

3.4 Evaluation

Having defined MSA, we now present a discussion of our empirical evaluation aimed at assessing the efficiency as well as exploring some potential future research direction for employing our Markov extension approach in business process event forecasting. Our model has been implemented in Matlab and run against two data sets of process instances from the records of a major telecom company. The first dataset (DS1) consists of telecommunication line fault repair records for a nine month period. The second (DS2) covers a one month period and represents a process for fixing broadband faults.

3.4.1 Process Analysis and Data Preprocessing

In these datasets, the population of process instances turned out to be very diverse and not straightforward to work with. In particular, *DS*1 is very diverse and contains 28963 entries, 2763 unique process instances, and 285 unique tasks. The process instance length varies from 1 to 78. On the other hand, *DS*2 represents a significantly smaller scale set with only 5194 entries, 794 process instances, and 10 unique tasks. The process instance length varies from 2 to 32.

Figure 3.3 shows a visualisation of 10 percent of set *DS*1. The diagram has been created using BT's process mining tool Aperture (Taylor *et al.* (2012)) which uses workflow

Algorithm 1 Function scan($row, col, length$): identifies the optimal editing process to match two sequences via deletion/insertion of symbols

```
1: $commonIndexes1(length) = row$
2: $commonIndexes2(length) = col$
3: **if** $(length > maxCommonLength)$ **then**                ▷ store the longest path
4:     $maxCommonLength = length$
5:     **for** $i = 1 : maxCommonLength$ **do**
6:         $maxCommonIndexes1(i) = commonIndexes1(i)$
7:         $maxCommonIndexes2(i) = commonIndexes2(i)$
8:     **end for**
9:     $maxCommonScore =$
10:    $calculateScore(n, maxCommonIndexes1, maxCommonIndexes2, maxCommonLength)$
11: **else**
12:    **if** $(length == maxCommonLength)$ **then**
13:        $score = calculateScore(n, commonIndexes1, commonIndexes2, length)$
14:        **if** $(score < maxCommonScore)$ **then**  ▷ if the paths are equal, store the smallest score
15:            **for** $i = 1 : maxCommonLength$ **do**
16:                $maxCommonIndexes1(i) = commonIndexes1(i)$
17:                $maxCommonIndexes2(i) = commonIndexes2(i)$
18:            **end for**
19:            $maxCommonScore = score$
20:        **end if**
21:    **end if**
22: **end if**
```

Algorithm 2 Funciton calculateScore(n, maxCommonIndexes1, maxCommonIndexes2, maxCommonLength): computes the score of matching two sequences

```
1: $totalPoints = 0$
2: $totalDeltas = 0$
3: $maxCommonIndexes1(maxCommonLength + 1) = n + 1$
4: $maxCommonIndexes2(maxCommonLength + 1) = n + 1$
5: $last1 = 0$
6: $last2 = 0$
7: **for** $i = 1 : maxCommonLength + 1$ **do**
8:     $dist1 = maxCommonIndexes1(i) - last1 - 1$
9:     $dist2 = maxCommonIndexes2(i) - last2 - 1$
10:    **if** $(dist1 > dist2)$ **then**
11:        $noPoints = dist2$
12:        $noDeltas = dist1 - dist2$
13:    **else**
14:        $noPoints = dist1$
15:        $noDeltas = dist2 - dist1$
16:    **end if**
17:    $totalPoints = totalPoints + noPoints$
18:    $totalDeltas = totalDeltas + noDeltas$
19:    $last1 = maxCommonIndexes1(i)$
20:    $last2 = maxCommonIndexes2(i)$
21: **end for**
22: return $totalPoints + totalDeltas * deltaValue$
```

data to create a process diagram. For this process, the diagram illustrates the complexity of the workflow data. It is basically impossible to visually analyse or understand the process from this figure. This is a typical scenario for processes that have evolved over time and are poorly documented.

The diagram shows the complexity of the process. An example of a readable diagram that helps understanding the process, a diagram of another real process from the same company generated by Aperture is shown in Figure 3.4

We also tried to build a Markov tree where at each node the procedure of splitting is just based on the supports of the unique tasks followed by the current node. This tree was built in the first place for the purpose of the next process steps. However, as can be seen from Figure 3.5, the tree is very large. Hence, it can be used as another illustration for the diversity of our available data.

To be able to have visualised analysis, a smaller tree is illustrated in Figure 3.6.

The data we used for our experiment was stored in flat files and each row (record) contains an unique event instance with its corresponding process name, date and other required attributes. When reading data into memory, each record also stored the three immediately preceding events as well as the next three future events for ease of access in the implementation of our algorithm.

3.4.2 Results

To evaluate MSA, we benchmarked our model with three other approaches:

- *RM - Random Model*: in order to find the next task following the current one, we randomly select from the set of potential next tasks. For example, if from the historical data, we know that tasks following A belong to the set $\{C,D,E\}$, we randomly select a value from that set as the predicted next step.

- *All K^{th} Markov Models*: a number of different order Markov models are generated from first order to K^{th} order. Given a sequence, we start with the highest order Markov model. If the given sequence cannot be found in the transition matrix, we create a new shorter sequence by removing the first event in the sequence. We then continue the procedure with the next lower order Markov model until we either find a match in a transition matrix or after trying the first order MM a default prediction is required.

- *HMM - Hidden Markov Models*: we tested several first order HMMs with different lengths for the input sequences and different number of hidden states and picked the best one.

For each comparison, we used 90% entries of the data sets as training data and the remaining 10% were used as test data. For each sequence in the test data, we checked if our model can correctly predict the next step of a particular task (i.e. we measured the number of successful predictions). The results were evaluated using 10-fold cross-validation. The results are displayed as a percentage. After the 1^{st} order to 7^{th} order MSAs were built, we investigated different values for l, the number of sequences (patterns) in the transition matrix which are most similar to the given sequence. l was varied from 1 to 7, and the results show that $l = 5$ is optimal for DS1 and $l = 3$ is optimal for DS2.

We now look at the specific results.

Figures 3.7 and Figure 3.8 illustrate MSAs accuracy against the two data sets, DS1 and DS2 respectively. The results show that by incorporating the default prediction improvement module, MSA outperforms other comparable models, especially when the order of Markov increases. This is because the relevance of the comparison between sequences is directly proportionate to their lengths. As can be seen, the second order MSA performs best among a range of different orders of MSA for the case of DS1, about 27% correct. The third order MSA works best in the case of DS2, it provides about 70% correct. Nonetheless, the highest performance order depends on the data set. For the data set which provides good performance with relatively high Markov order (i.e. 5^{th} and above), the role of our default prediction improvement module becomes highly significant.

Figure 3.9 shows the performances of all models against the two datasets. As can be seen, RM performs worst with only around 10% success for DS2. When applied to the bigger dataset DS1, the result goes down to nearly 2% (14 times worse than that of MSAs' average performance). This can be explained by the fact that with DS2, each task can have an average of 7 potential next tasks whereas this figure is nearly 30 for DS1. Thus, the probability of picking the right next task reduces as the set size increases. The results highlight the difficulty of handling complex data sets. When the data is not too diverse (i.e. DS2), MSA (fifth order) obtains the highest number of correct predictions with a result of 63%. Compared to other benchmarks, MSA results are better than a fifth-order Markov model which achieves 57%, an All K^{th} (K = 5) Markov model, which achieves 60%, and an HMM which achieves 43%. Please note that we did not pick the most accurate MSA and All K^{th} Markov model to compare to HMM and RM. The reason is that All K^{th} Markov models and MSAs only have an advantage with higher orders. In order to see how well they cope with default predictions, we chose a 5^{th} order MSA and an All 5^{th} Markov model.

When the data is more fragmented as in the case of DS1, the effectiveness of our model is reduced. However, even though the performance on DS1 is not as good as the performance on DS2, our model still performs an important role considering the requirement of selecting a single correct task from 30 potential candidates on average. Even in this case we still manage to outperform the HMM result by 20% and also outperform the All 5^{th} Markov model result by 3%.

3.5 Conclusions

In this chapter, we have introduced several models for predicting the next step in business processes in a telecommunication services context where no formal description of the processes exists or the real process significantly deviates from the designed path prototype when applied in real environments and under realistic constraints. Specifically, we first applied Markov and Hidden Markov models to generate probability matrices of moving from one event to others; and then using them to provide predictions. Next, we have proposed a novel predictive model (MSA) which is an extension of Markov models employing sequence alignment. In order to analyse its effectiveness, we have empirically evaluated MSA against two datasets from a major telecommunication company with varying degrees of diversification. The results have clearly demonstrated that MSAs outperform both Markov models and HMMs by at least 10%.

In addition, we have also shown that higher order Markov models outperform first order Markov models. The default prediction module we introduced significantly improves the corresponding original Markov models result when the order of model is high (starting from 5^{th} order MM). Although the improvement provided by the new default prediction module is quite significant, higher order MSAs do not outperform second order MSA in the case of the data set DS1 and third order MSA in the case of the data set DS2.

The results of the experiments show that the significant improvement achieved by adding the default prediction module still cannot help to bring our predictive models into a situation where they can provide live support to the process as the accuracy is considerably lower than would be required. Other potential solutions need to be investigated aiming to predict next process steps and process outcomes.

However, these experiments also motivate us to use sequence alignment technique as a similarity measure to group sequences which have similar characteristics and then tackle them separately. The next chapters will look into: (1) using sequence alignment, sliding window and K-nearest neighbour to predict the outcome of the process and (2) K-means clustering to cluster data into K groups and then treat each group with suitable approaches.

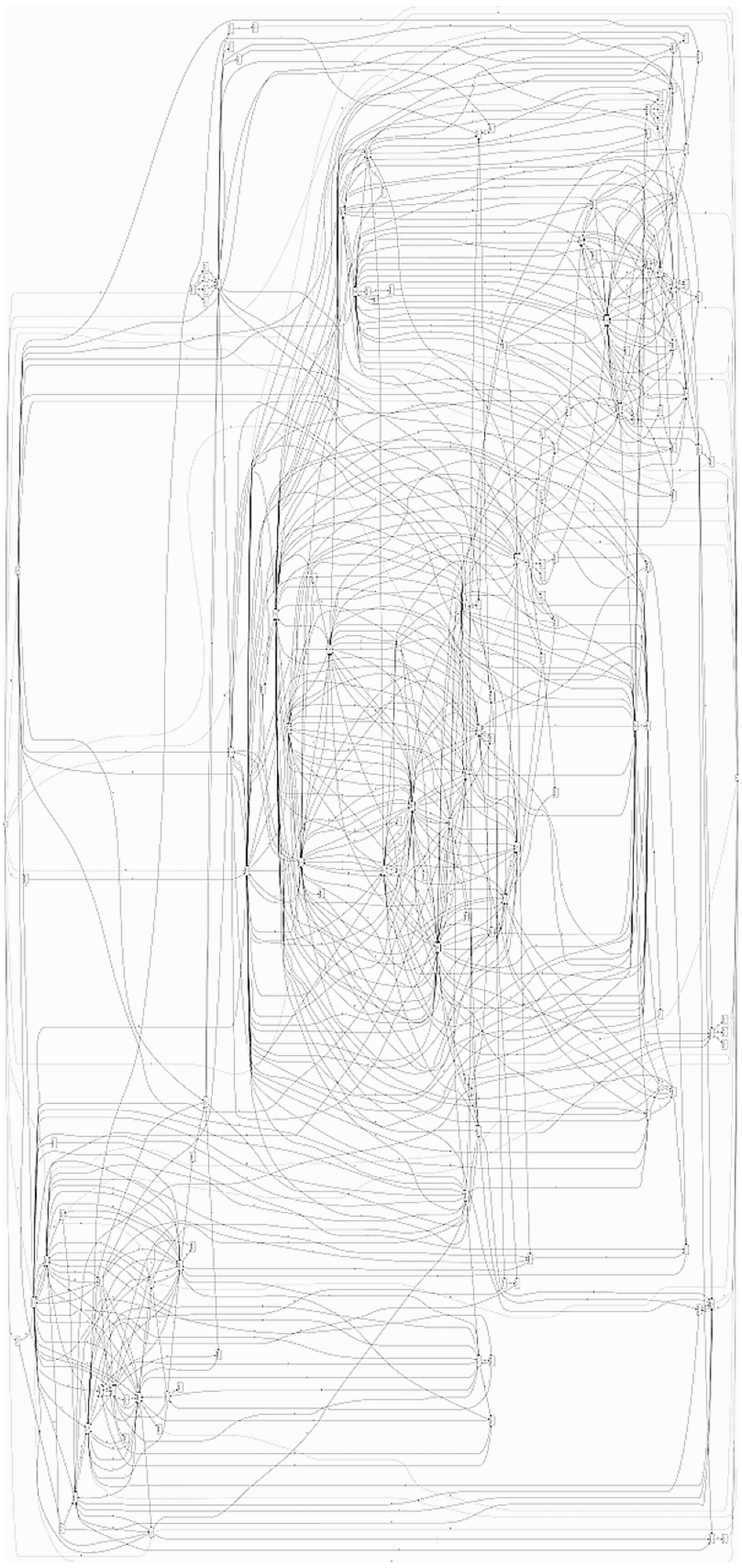


Figure 3.3: Process model obtained by using Aperture for DS1 visualising a highly complex process.

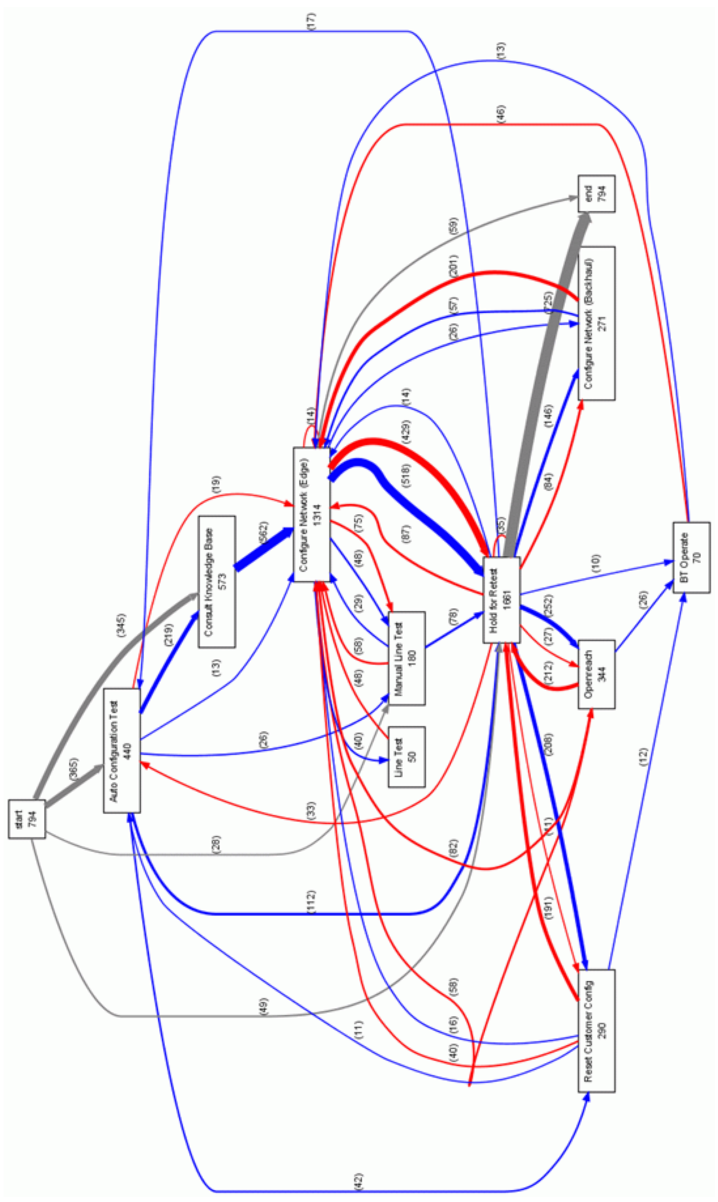


Figure 3.4: An example of a simple process generated from a real process data using Aperture.

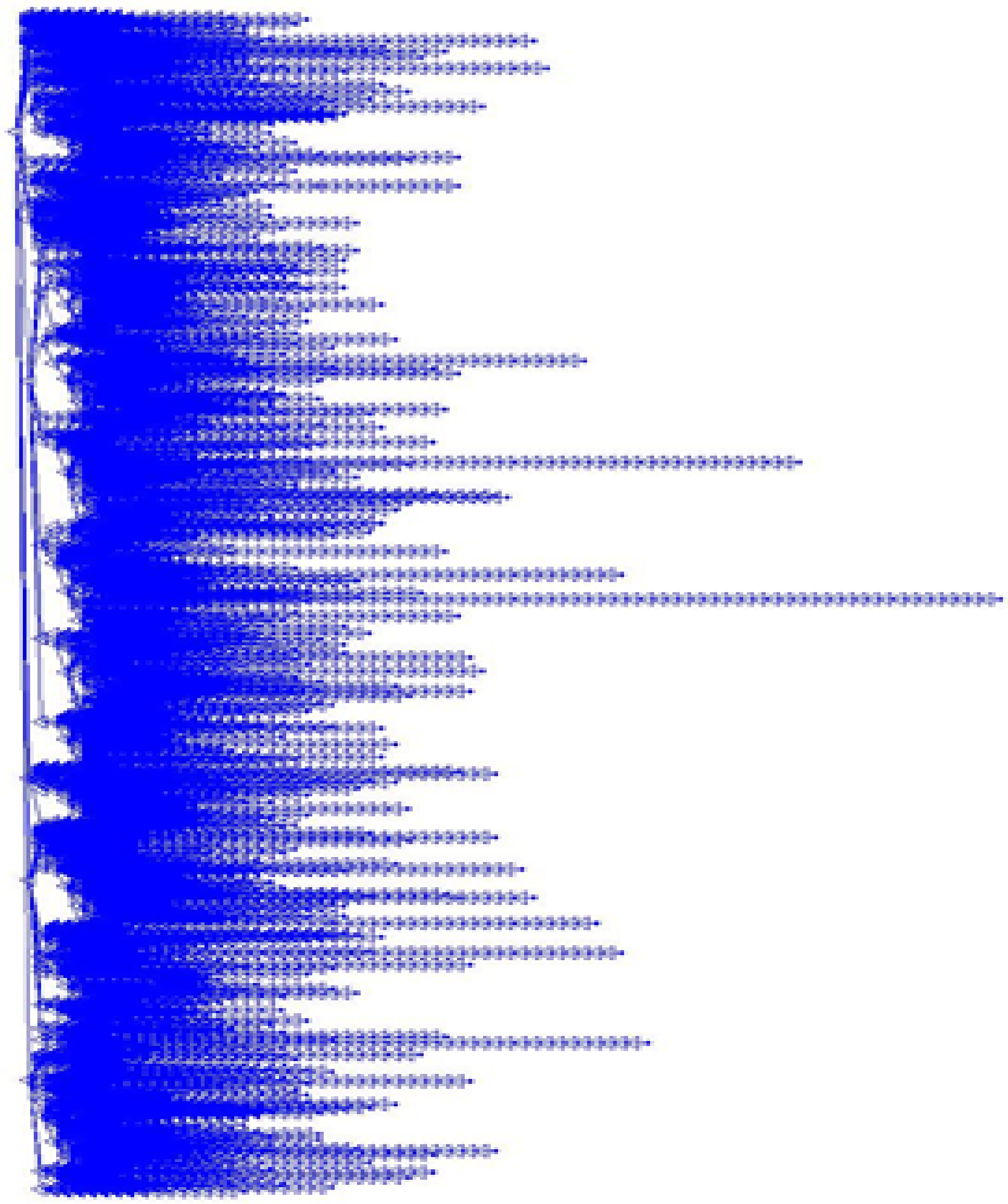


Figure 3.5: A Markov tree built from the dataset DS1.

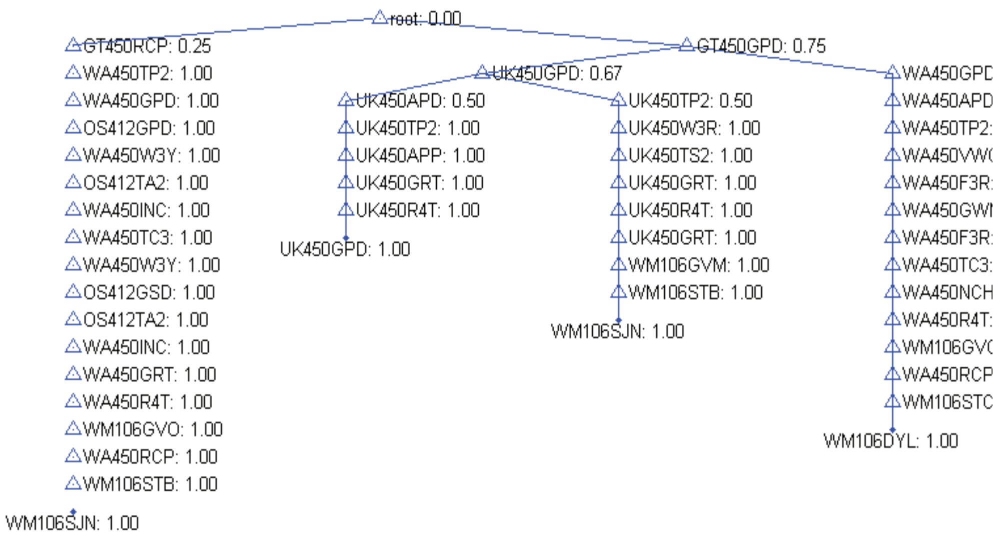


Figure 3.6: Part of the Markov tree built from the dataset DS1.

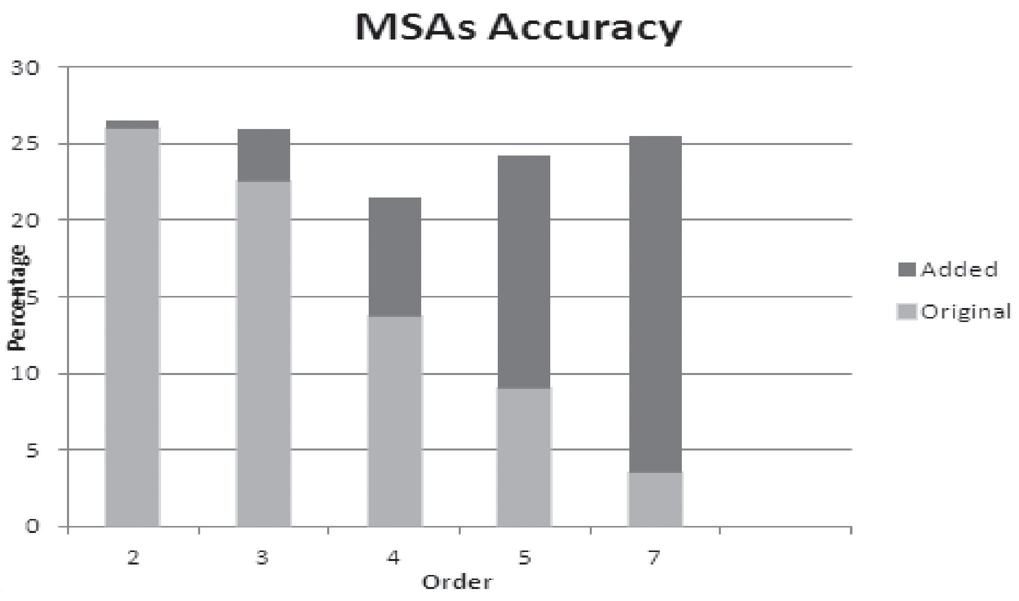


Figure 3.7: Percentage of correct predictions before and after applying the default prediction module in Markov models using dataset DS1.

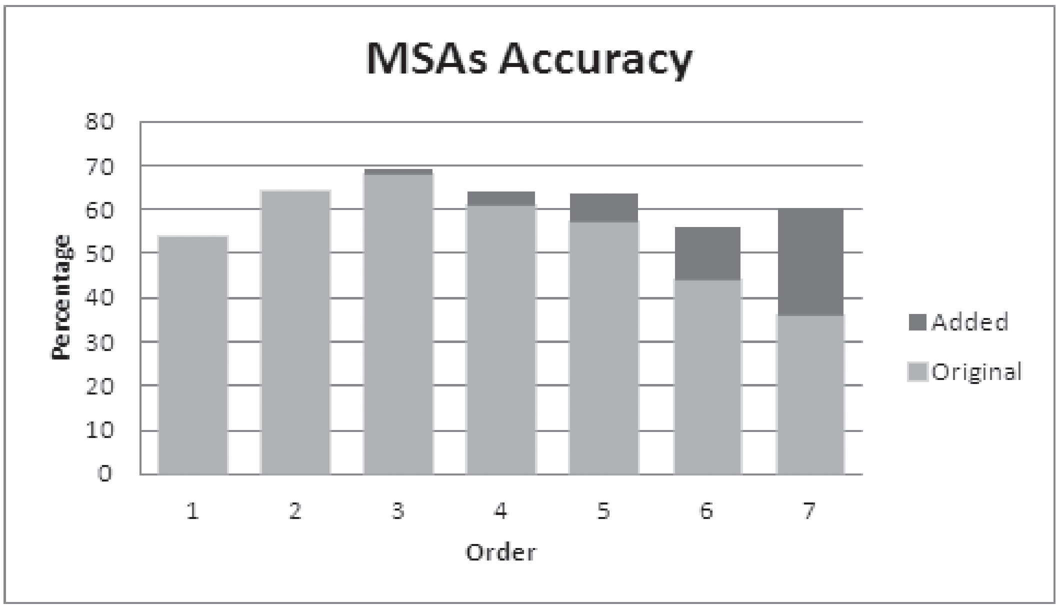


Figure 3.8: Percentage of correct predictions before and after applying the default prediction module in Markov models using dataset DS2.

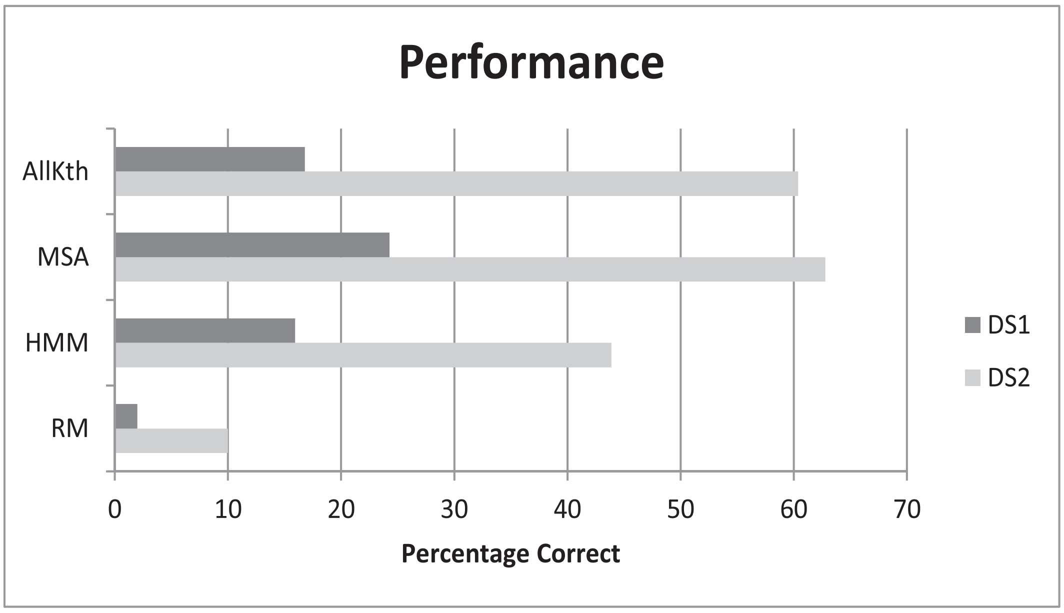


Figure 3.9: Percentage of correct predictions of different models on DS1 and DS2 datasets.

Chapter 4

Process Outcomes Predicting and Process Failure Warnings

Business process data is sequential in nature and can be very diverse. Thus, there is a need for an efficient sequential forecasting methodology that can cope with the diversity of the business data. In response to these requirements, our first attempt was to use probabilistic models which were presented in the previous chapter. The results show insufficient performance for the domain. To continue the research line with lessons and hints gained from the previous work, in this chapter, we propose an approach which is a combination of KNN (K nearest neighbour) and sequence alignment for predicting process outcome. The proposed approach exploits temporal categorical features of the extracted data to predict the process outcomes using sequence alignment technique. It also addresses the diversity aspect of the data by considering subsets of similar process sequences, based on KNNs. We have shown, via a set of experiments, that our model offers better results when compared with original KNNs and random guess-based methods. We also introduce a rule based technique based on GOSPADE, which detects the repetitions of a certain task and investigates the relationship between them and the process failure. The results are demonstrated in a comprehensive performance study on real business process data sets.

4.1 Introduction

In process mining, predictions and recommendations based on models learnt from historical data can be used for online maintenance because being aware of what might happen helps managers to set up suitable strategies to intervene on time (van der Aaslt *et al.* (2011)). For example, it may be of interest to investigate certain loops that might lead to process failure; suggesting an optimal way to complete the process starting from the current step etc.

Similar to data from business process executions, customer behaviour data also consists of asynchronous sequences and is considered as a kind of stochastic process. Customer behaviour data is described by sequences of interactions between the customers and the company while business processes are described by sequences of process events, i.e. steps or tasks. In both cases the events and the entities (customers or jobs) that move through the process instances are described by additional attributes. Therefore, in this study we use both types of data, business process data and customer behaviour data in order to test out if our proposed model is generic in terms of dealing with sequential data.

Due to the complexity and diversity of process data as well as the sequential nature of process data, it seems to be poorly aligned with any single classical data mining technique. This chaper introduces a new K nearest sequence method which is able to deal with sequential data. Our objective is to group similar sequences together expecting that sequences which behave similarly in earlier steps go to the same final step. In order to create a KNN model which addresses the diversity and the temporal character of the data, an original KNN is combined with sliding window and sequence alignment approaches.

KNNs belong to the family of supervised learning algorithms which have applications in many fields. In our area of application we consider a number of sequences in which we want to find K sequences which are most similar to a given sequence $S(s_1^{(j)},\dots,s_{n_j}^{(j)})$, where j is the identity of the sequence. The similarity is determined using a distance function and, the prediction should be the majority class among the K nearest sequences' classes. This method is used in Ruta *et al.* (2006) to predict churn using data from a telecommunications company. The authors combined KNN and the theory of survival. In their work, Euclidean distance is used to calculate the distance between the given sequence and all sequences in the data sample. As our data consists of event sequences, in this study we use the edit distance in the process of comparing two sequences.

It is also of interest to look into loops, which occurred in the data, to identify if some of them might lead to process failure. Hence, apart from the proposed predictive models, in this chapter we employ a potential technique in order to provide warnings about process failure. It is based on a special algorithm in association rules called GOSPADE (Leleu *et al.* (2003)). The proposed technique enables us to detect loops and the links between them and the process failure.

The problem can be formulated as follows: A business process instance (S_j) which is a composition of discrete events (or tasks) in a time-ordered sequence

$$S_j = \left\{s_1^{(j)}, s_2^{(j)}, \dots, s_{n_j}^{(j)}\right\}.$$

s_k takes values from a finite set of event types $E = \{e_1,\dots,e_L\}$. Apart from its starting time $t_i^{(j)}$ and ending time $T_i^{(j)}$, each of these events has its own attributes. For example, consider a process of making pizza. If we ask one hundred chefs to prepare one pizza each, giving them a recipe and recording steps they are doing, we then have a data set of one hundred sequences. Each sequence is called a process instance. Each step they make is called an event (task), like turn on oven, preheat oven, set temperature, prove dough etc. Events can be done in different orders, have different time durations and can be done by different chefs. If chefs preheat the oven whilst the dough proves so there is potentially a parallel structure. Additionally, chefs may have their own way of making pizza, so they might slightly change the recipe. Let's say some steps can be skipped or some steps take longer to finish than the designed version. Here, the chef is one attribute of the process instance, for each event we have attributes like time, duration etc.

The goal is to predict the outcome (success/failure) of a given process instance $S_{N+1} = \left\{s_1^{(N+1)}, s_2^{(N+1)}, \dots, s_{i-1}^{(N+1)}\right\}$ based on the data from closed process instances $S =$

$\{S_1, S_2, \ldots, S_N\}$ and to determine if the consecutive repetition of a task S_j potentially leads to a failure for the considered process instance.

The remainder of this chapter is organised as follows: Section 4.2 presents the proposed models for predicting the process outcome. Section 4.3 discusses the GOSPADE algorithm based loop failure detection (LFD) technique. It is followed by Section 4.4 with a brief review of the data used in our analysis before we present the results of our experiments. In Section 4.5, we conclude the chapter and discuss research directions for the next chapter.

4.2 Sequential K Nearest Neighbours

To determine sequence similarity both distance measures and similarity measures can be used because distance and similarity are dual concepts. For numeric variables well-known distance measures exist and can be easily applied. However, in sequence analysis sometimes we have to work with sequences which are constructed from symbols, e.g. categories (churn prediction), phonemes (speech recognition) or characters (hand-writing recognition) etc. We need specialised functions which have the ability of measuring the similarity of symbolic sequences. Some types of distances commonly used are listed below:

- Euclidean distance in metric space (Berry and Linoff (2004)).
- Hamming distance (Berry and Linoff (2004)).
- Edit distance, the so-called distance of insertion and deletion with penalties (Waterman (1994)).

Among these distance measure functions, Hamming distance also can be used for categorical variable but the draw back of Hamming distance is that it can only cope with sequences of similar length. Therefore, edit distance is more suitable especially when we want to adopt the sequence alignment technique from biology.

4.2.1 Sequence Alignment

Sequence alignment is very common in bio-informatics and has a relatively long history in this domain. The target entities of sequence alignment in bio-informatics are amino acid sequences of proteins, DNA sequences etc. Sequence alignment is used for a number of purposes (Needleman and Wunsch (1970)), for example, to compare new DNA sequences to DNA databases and transactions of DNA sequences into amino acid sequences are compared to protein databases in order to verify if the relationships that are found between them could have occurred by chance. Algorithms used in sequence alignment are mainly divided into global alignment and local alignment. Global alignment provides a global optimisation solution, which spans the entire length of all query sequences. In contrast, local alignment aims to find the most similar segments from two query sequences. In this work, both types of alignment are investigated to verify which one is effective in determining the similarity between the two sequences, the overall comparison or the most similar consecutive segments. The similarity between process sequences is used to predict the process outcomes.

4.2.2 KNNs Combined with Sequence Alignment

KNN is one of the classical approaches in data mining (Berry and Linoff (2004)). It is a none parametric approach hence one of its advantages is that there is no parameter training procedure required. The model automatically selects sequences from the continuously updated data. KNN can be used as an original non sequential approach (Eastwood and Gabrys (2009)) or extended into sequential approach (Ruta *et al.* (2006)). The core idea is to find similar sequences which have a common behaviour and outcome. This expectation is understandable and is common, for example in biology, similar DNA or protein sequences are hoped to have the same shape function.

To conserve the sequential nature of business processes, we want to extract K similar sequences in terms of their temporal characteristics not numerical quantities. That is why the initial idea is to adopt the sequence alignment approach from biology and combine it with KNNs. The resulting approach is named KNsSA (K nearest sequence with sequence alignment).

To estimate the similarity between two sequences, the longest common segments between them can be used a as measure and local sequence alignment can be applied directly. Alternatively, two sequences can be aligned globally, this finds the optimal matching by aligning from the first event to the last one of the two sequences. In this case, global matching allows us to compare the sequences in a whole rather than focus on their most similar consecutive segments.

The available data consists of sequences of different lengths and sometimes the difference between their lengths is large. To avoid a negative impact of a large difference in sequence lengths we can employ a sliding window technique when using global alignment (Dietterich (2002), Du *et al.* (2004)). The idea is to put and fix the given sequence inside the window. This window is then moved along the data sequences. For each sequence, the most similar segment (subsequence) is selected and the score of this segment is chosen as the representative score for the sequence. The window size needs to be flexible in our model because of the difference in the length of given sequences. Anytime when we need to predict the outcome for a new process instance, the length of this given sequence is chosen as the size of the window. One remark here is that we only deal with the temporal order in sequences, we have not taken into account the duration of the time interval for these events to occur in the window. This can be investigated as well if it is useful in which case the window size is parameterised by the duration of the time interval.

4.3 GOSPADE based Loop Failure Detection

Intuitively, repetitions might slow down the completion of a process. Hence, we would like to look into the data and search for rules related to task repetition (loops) and the process outcomes (e.g. success/failure). The GO-SPADE algorithm in association rules was designed to deal with consecutive data and it is suitable for detecting loops. We now introduce some preliminary concepts in association rule mining in order to help to illustrate the principle of the GOSPADE algorithm which is briefly discussed afterward.

- A rule of length k (composed of k elements) is called k-frequent sequence (pattern).

- Prefix p of a k-frequent rule z is the subsequence of z which consists of the first $(k-1)$ elements.
- Suffix s of a rule z is its last element.
- The location information of a pattern is listed and stored in a table, called idList. This information consists of the Id of the sequence in which the pattern occurred, the position of the pattern in the sequence and the rule itself (each pattern has its own idList). These sequence Ids and positions are denoted sid and eid respectively (please see Tables 4.1 - 4.4). While scanning through the data, each time the considered pattern occurs, a new row is added to the list for storing the information about the Id of the corresponding data sequence as well as the position of the pattern in the sequence. If the pattern is of length 1 then eid is the position of the task in the given sequence, otherwise eid is the position of the last task in k-frequent sequences when $k > 1$. To present the repetition of a pattern, instead of storing repeatedly it in different rows in the list, only one row is added to store the pattern by listing all of the eids. A new row is not required because the sequence Id and the pattern itself are the same for these repetitions. The eids of the replaced pattern are then written in interval form $[i, i+j]$, where i, $i+1$, ..., $i+j$ are the eids of the consecutive repeats of the pattern. If the pattern occurs only once at position i, we write $[i, i]$.

The GO-SPADE algorithm consists of two stages

- Generate candidates step: In the original algorithm, SPADE distinguishes two types of patterns, event patterns and sequence patterns. These types of patterns are determined based on the relationship between a prefix and its corresponding suffix. If the last item of the prefix p occurs at the same time as the suffix s, the pattern is called event pattern and is denoted ps. If the suffix s occurs strictly after the last item of the prefix p then the pattern is called sequence pattern and it is denoted $p \to s$. In our case, the available data consists of sequences of events. As mentioned in the introduction parallel structures are not considered and we are looking at sequence patterns. However, we work with loops only so the only IdList we have to investigate is IdList of length 1. It then does not matter if we are dealing with event patterns or sequence patterns.

  Let us consider event patterns for the sake of simplifying the explanation of the algorithm. For more detail, refer to Leleu *et al.* (2003). In the first step, candidate frequent patterns are generated. To generate frequent patterns of length $k+1$, all frequent patterns of length k which have the same prefix $k-1$ are considered. Once we have all k frequent patterns with the same $k-1$ prefix, the corresponding IdLists are used. Consider two k-sequences $z_1 = (p_1, s_1)$, $z_2 = (p_2, s_2)$, $p_1 = p_2$ and the corresponding IdLists IdList(z_1) and IdList(z_2). If prefixes p_1 and p_2 are the same, z_1 and z_2 are then merged to generate a new frequent sequence z. The generated sequence is $z = p_1 s_1 s_2$.

  To compute the IdList for the generated sequence IdList(z), a join operation is used. If sid1 is the same as sid2 then compare the eid of suffixes s_1, s_2 if eid1 is smaller

sid	eid	Task	Loop
P1	[1, 3]	A	L
P2	[2, 2]	A	No
P4	[1, 1]	A	No

Table 4.1: The corresponding IdList of pattern of length 1 IdList(A).

sid	eid	Task	Loop
P1	[5, 5]	B	No
P2	[1, 1]	B	No
P2	[3, 3]	B	No
P3	[1, 1]	B	No
P4	[2, 3]	B	L

Table 4.2: The corresponding IdList of pattern of length 1 IdList(B).

than eid2 then chose eid2 as the eid of the new frequent sequence in case the support of generated sequence is bigger than minimum support.

- Counting support step: in GO-SPADE, it is easy to count the support of the generated sequences. It does not need to go through the database to count the support of a candidate sequence. All the required information is already stored in its IdList.

Example Given four process instances (sequences) $P1$: $AAACB$ - success, $P2$: BAB - failure, $P3$: BCD - success, and $P4$: $ABBCDD$ - failure. Assume that we want to detect loops and just need to generate the IdLists of single events. The 1-IdLists generated from the given sequences are illustrated in Tables 4.1 - 4.4:

From the IdLists for A, B, C and D, we select all the loops to generate a new list as illustrated in Table 4.5. Based on the resulting list, we compute the percentages of failure and success. This GOSPADE based technique is the loop failure detection technique.

The pseudo code illustrated the proposed algorithm is given in Algorithms 3 and 4:

4.4 Evaluation

4.4.1 Data Preparation

We carried out a number of experiments based on records from three real processes ($DS1-3$) from BT, a multi-national telecommunications company. In these datasets, the population of process instances turned out to be very diverse and not straightforward to work with. $DS1$ represents a small scale set with only 10000 entries, 633 process instances, and hundreds of unique tasks. $DS2$ is also a real process with 11839 entries, 576 process instances with

sid	eid	Task	Loop
P1	[4, 4]	C	No
P3	[2, 2]	C	No
P4	[4, 4]	C	No
P3	[3, 3]	C	No

Table 4.3: The corresponding IdList of pattern of length 1 IdList(C).

sid	eid	Task	Loop
P3	[3, 3]	D	No
P4	[5, 6]	D	L

Table 4.4: The corresponding IdList of pattern of length 1 IdList(D).

sid	eid	Outcome	Task	Loop	Support	Confidence
P1	[1, 3]	S	A	L	25%	1.0
P4	[2, 3]	F	B	L	25%	1.0
P4	[5, 6]	F	D	L	25%	1.0

Table 4.5: Loops and corresponding process outcomes found by LFD technique from a studied data set.

Algorithm 3 $-$**Proc** : $generateSingleSpades(uniqueTasks, processes)$

```
1: $generateSingleSpades = []$
2: for all $uniqueTask \in uniqueTasks$ do
3:     $spade = $ new$Spade(level = 1, task = uniqueTask)$
4:     $spade.contents = []$
5:     for all $process \in processes$ do
6:         for all $position \in [1 : process.totalTasks]$ do
7:             if $process.tasks(position) = uniqueTask$ then
8:                 $content = last(spade.contents)$
9:                 if $(content \neq empty)$and$(content.lastIndex = position - 1)$ then
10:                    $content.lastIndex = position$
11:                else
12:                    $content = $ new$SpadeContent(processIndex = process.index, firstIndex = position, lastIndex = position, loop = false)$
13:                    $spade.contents \leftarrow content$
14:                end if
15:            end if
16:        end for
17:    end for
18:    call$checkSpadeLoop(spade)$
19:    $generateSingleSpades \leftarrow spade$
20: end for
```

Algorithm 4 $-$**Proc** : $checkSpadeLoop(spade)$

```
1: for all $position \in [1 : length(spade.contents)]$ do
2:     if $(position > 1)$ and $(spade.contents(position).processIndex = spade.contents(position - 1).processIndex)$ then
3:         $spade.contents(position - 1).loop = true$
4:         $spade.contents(position).loop = true$
5:     end if
6:     if $(spade.level = 1)$ and $(spade.contents(position).firstIndex = spade.contents(position).lastIndex)$ then
7:         $spade.contents(position).loop = true$
8:     end if
9: end for
```

different lengths, and also has hundreds of unique tasks. The lengths of process instances in *DS*2 vary considerably. Last of all, *DS*3 is a customer behaviour data set used for churn prediction. This data set consists of 8080 customer records, each record is a sequence of events related to a customer. There are four types of events in churn data, churn, complaint, repair and provision. These customer event sequences are also of different lengths. Among these 8080 sequences, only 161 sequences contain a churn event. This shows the skewness of the data. Therefore, we artificially created new datasets with different ratios between non-churn and churn sequences in order to reduce the skewness of the data and to investigate if this characteristic of the data has a strong influence on the prediction model.

The content of these datasets were stored in flat files where each row (record) contains a unique event instance with its corresponding process name, date and other required attributes (the number of these attributes varies from twenty to forty). When reading data into memory, each record is stored together with the three immediately preceding events as well as the next three future events for ease of access in the algorithm implementation.

As we aim to predict the process outcome and to provide warning about process failure, it is necessary to label the outcome as either success or failure and this needs to be done before the proposed method can be applied. There is, however, an issue of data avaibility as for some of these processes, it is non-trivial to define process success or failure.

To solve the problem, we have to look for different criteria to classify process instances into success/failure for individual datasets. One such criteria is the duration used to complete a process instance. That is to label a process instance as a success if the time duration used to complete the process instance is under a chosen threshold, otherwise, it is labelled as a failure. In the case of *DS*2, the difference between the actual delivered date and the delivered date promised to the customer is used as the criterion to determine the success and failure instead. Specifically, if the actual delivered date is before the agreed date, that process instance is classified as success, otherwise, it is classified as failure. In contrast to *DS*2, *DS*1 and *DS*3 (churn and no churn labels) have available labels and therefore, they can be used directly as input for the proposed model. On a similar note, *DS*3 is a special case where it contains the customer behaviour data for churn prediction.

The complete time durations of each process instances are highly variable. Having a distribution of complete time durations can help to filter out all the uncompleted process instances. It also helps to provide an overview of complete time duration values and to find a suitable threshold. For these reasons, the complete time duration is calculated and plotted in Excel. For the purpose of providing an overview of processes' completion time, DS2, DS4 and DS5 are used. Similar to DS1-3, DS4 and DS5 are real processes in BT. Their completion time duration distributions are given in Figures 4.1, 4.2, 4.3 and 4.4.

Based on the complete time duration distributions, the threshold for each data set is chosen based on the mean and variance values of the corresponding distribution. The GO-SPADE algorithm is applied on the studied datasets, including DS1-3.

4.4.2 Results for Predictive Models

To evaluate KnsSAs, we benchmarked our models with two other approaches:

- *RM - Random Model*: in order to find the outcome of the process, we randomly

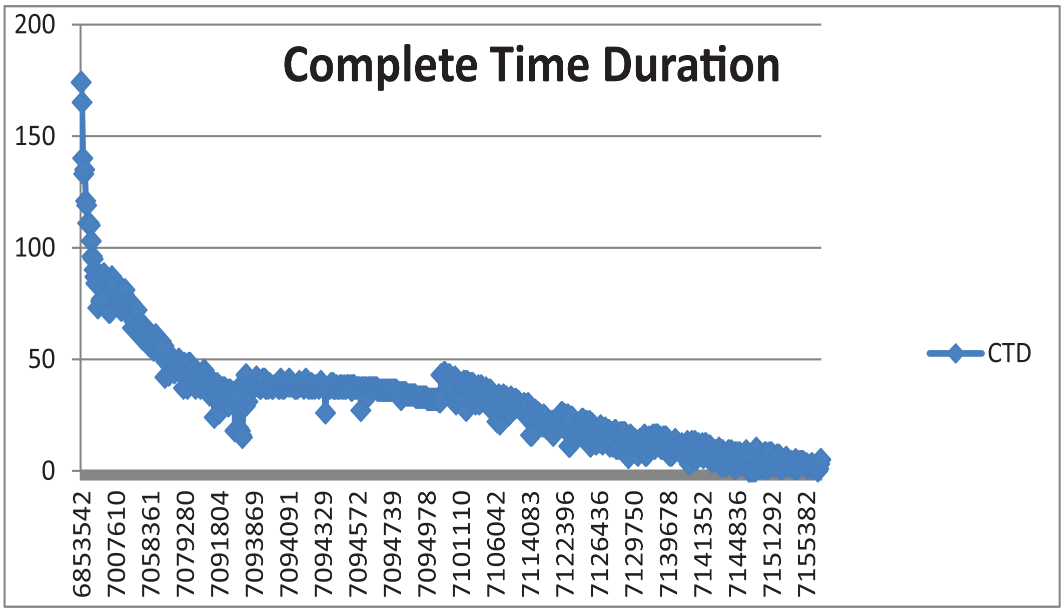


Figure 4.1: Process complete time duration distribution of the Showcase1 dataset.

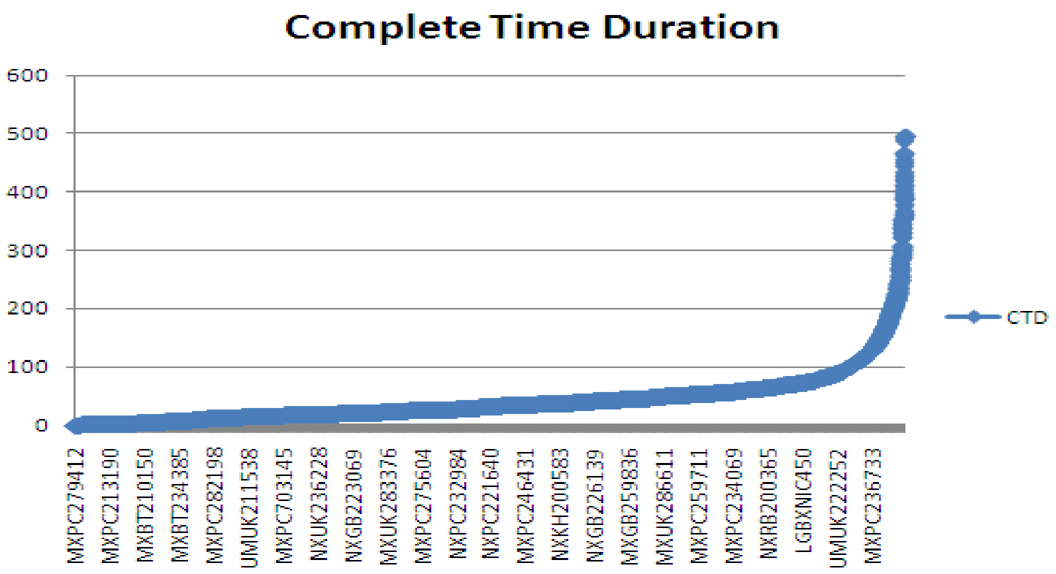


Figure 4.2: Process complete time duration distribution of the Showcase1 dataset, full version.

generate a number between 0 and 1, if the generated number is greater than 0.5 the outcome is success (1) and vice versa if the generated number is smaller than 0.5 the outcome is failure (0).

- *Original KNN*: we chose K nearest sequences in terms of having common unique tasks. For example, given two sequences ABD and AAC, there are one A, one B and one D in the first sequence, there are two A's and one C in the second one. Each unique task can be considered as one category, distance for each category is computed then the sum is taken to obtain the total distance between any two given sequences.

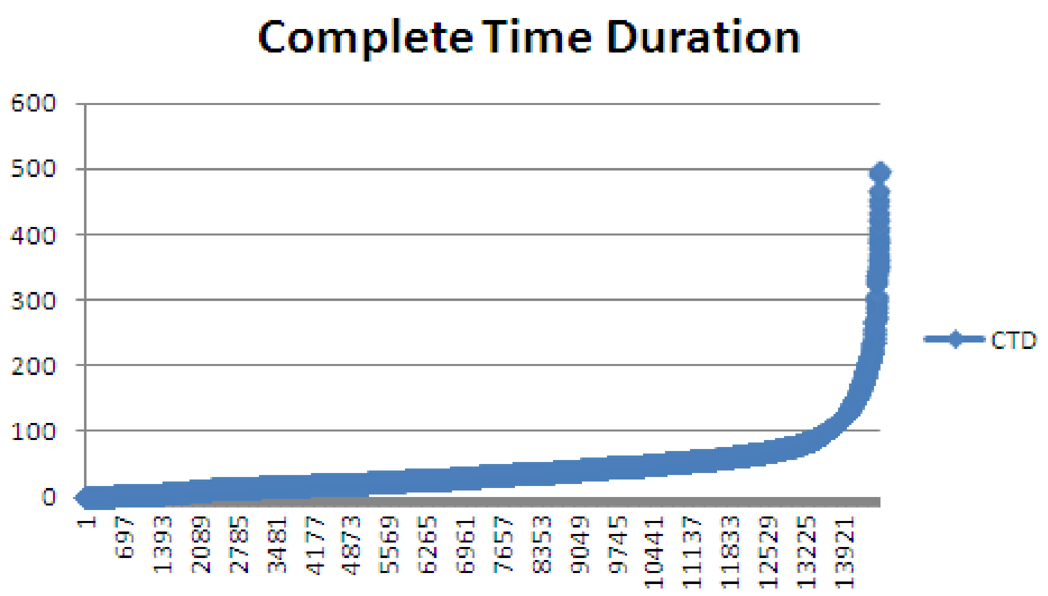


Figure 4.3: Process complete time duration distribution of the Showcase1 dataset, filtered version (all durations ≤ 500 hours).

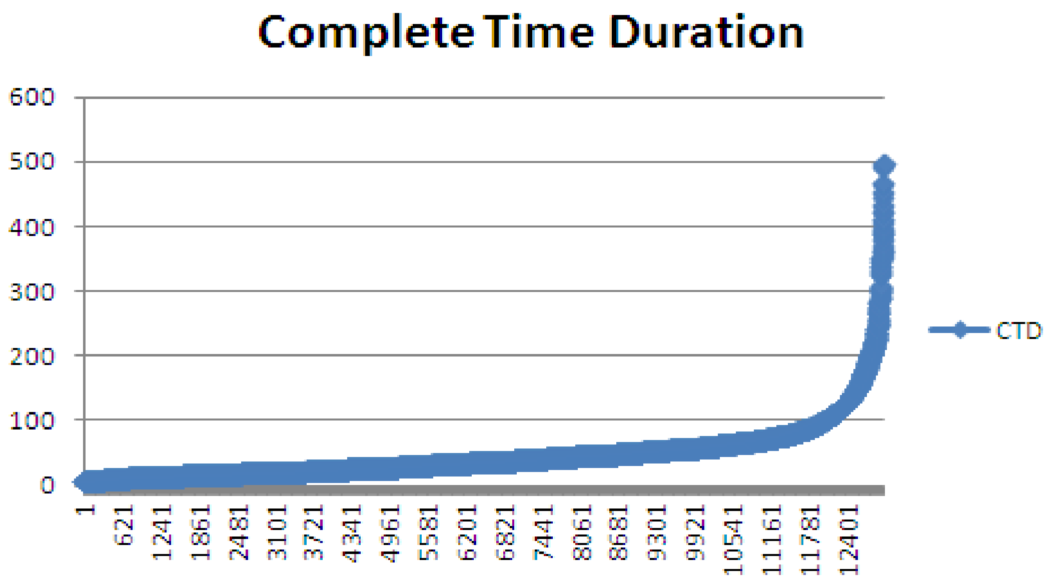


Figure 4.4: Process complete time duration distribution of the Showcase1 dataset, filtered version (all durations ≤ 500 and ≥ 6 hours).

For instance, the two sequences given above consist of four categories A, B, C and D. The distance of category A is $d_A = 1$, and those of categories B, C, D are $d_B = 1$, $d_C = 1$ and $d_D = 1$ respectively. The resulting total distance of these two sequences is $d = d_A + d_B + d_C + d_D = 4$.

For the proposed models, we investigate the effect of K as it is important to get a reasonable number of similar sequences. As the labels are 0, 1, we decide to select odd values for K so we can always extract the outcome/label of the given sequence based on the

K obtained sequences. The data sets *DS*1 and *DS*2 have a large number of unique tasks and the difference between the lengths of the sequences is substantial. Intuitively, the value of *K* should be small taking into account the diversity of the data.

The results of applying local KnsSA to data sets *DS*1 and *DS*2 are presented in Figure 4.5.

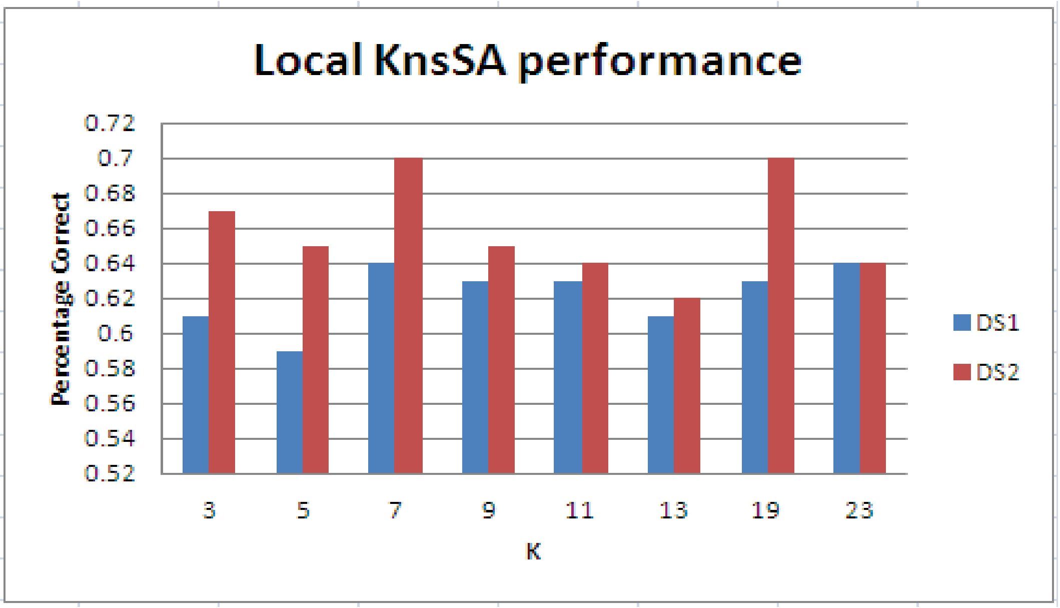


Figure 4.5: Percentage of correct predictions of local KnsSA using data sets DS1, DS2.

We then tested global KnsSA using the same datasets. The results are illustrated in Figure 4.6

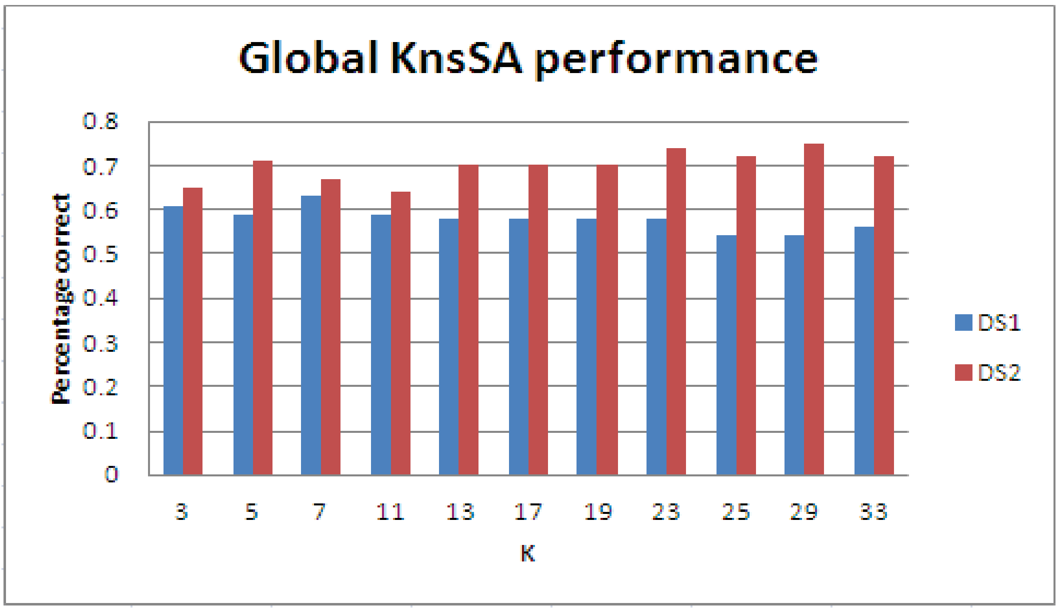


Figure 4.6: Percentage of correct predictions of global KnsSA using data sets DS1, DS2.

Figures 4.5 and 4.6 show that both global KnsSA and local KnsSA are more accurate when they are applied to $DS2$. For $DS2$, global KnsSA with a higher value of K provides a better performance. When $K = 29$ the performance of the global KNsSA is 75%. On the other hand, applying global KnsSA to $DS1$ did not achieve similar high performance. The highest correct percentage obtained is only 64% with $K = 7$. Moreover, when K is increased, global KnsSA's performance on $DS1$ decreases. This can be explained with the observation that $DS1$ is more diverse than $DS2$.

Global and local KnsSAs do not show any difference when they are applied to $DS1$. However, they show some differences in the case of $DS2$. This indicates that there are no important segments which have influence on the process outcome in $DS1$.

The performance of the original KNN, random guess and the proposed models, local and global KnsSAs, applied to the three data sets are presented in Figure 4.7.

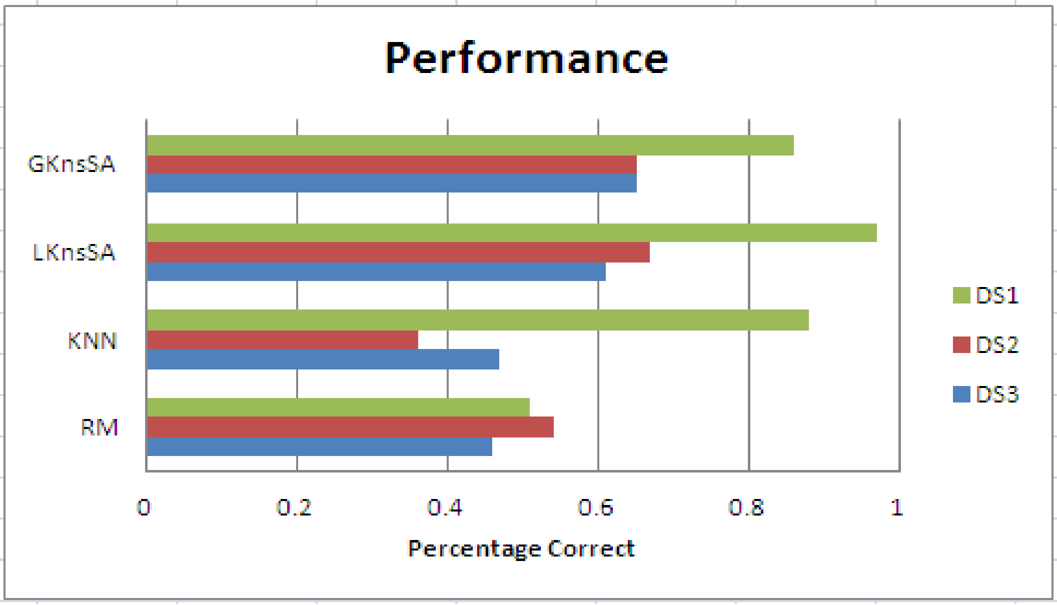


Figure 4.7: Percentage of correct predictions of different models on data sets DS1, DS2 and DS3.

The results show that the proposed models outperform both benchmark models, original KNN and random guess. Especially, in the churn data, the proposed models capture churn events with a high degree of success whilst the other models do not. This also implies that the temporal characteristics of the data are important for predicting the process outcome.

We now present the results of experiments which apply the two models, global and local KnsSAs to the churn data, $DS3$. The results are presented differently from the results for $DS1$ and $DS2$. This change is mainly made to adequately deal with the skewness of the churn data, the proportion between class 0 and class 1 in $DS3$ is 98:2 whilst that proportion for $DS1$ and $DS2$ is about 40:60. There are four tables which illustrate different aspects of the experiments objectives. Table 4.6, shows the difference obtained by varying K, using local KnsSA and the original churn data. Table 4.7 demonstrates the results obtained by

Results/K	3	5	7
Total test data	809	809	809
Actual tests successful	793	793	793
Actual tests failure	16	16	16
Predicted tests successful	803	805	805
Predicted tests failure	6	4	4
Corrected predictions	799	797	797
Failed predictions	10	12	12
Predicted tests success correct	793	793	793
Predicted tests failure correct	6	4	4
Correct ratio	0.99	0.99	0.99

Table 4.6: Local KnsSA applied on original DS3 with different K, K = 3, 5 and 7

Results/ratio	0.05	0.10	0.15
Total training data	511	875	1189
Total test data	59	100	135
Actual tests successful	45	83	120
Actual tests failure	14	17	15
Predicted tests successful	45	84	124
Predicted tests failure	14	16	11
Corrected predictions	57	93	127
Failed predictions	2	7	8
Predicted tests success correct	44	80	118
Predicted tests failure correct	13	13	9
Correct ratio	0.97	0.93	0.94

Table 4.7: Local KnsSA applied on original DS3 with different churn and non churn ratios $K = 3$

using local KNsSA when artificial data was created by changing the ratio between churn and no churn sequences in order to decrease the skewness of the data. Table 4.8 shows the performance of global KnsSA when applied to the former artificial data sets. Finally, Table 4.9, shows the performance of global KnsSA combined with sliding window using the artificial data sets.

It can be seen from Table 4.6 that $K = 3$ is the best case for the churn data as our dataset is not large. Also, when the original data was modified in order to reduce its skewness, the performance of the model in terms of the churn detection objective improved even though the overall performance worsened. Intuitively, when the population of churn sequences strongly dominates, it is very likely that our model could not catch the full churn set. It is shown in Table 4.6 that there are only 6 predicting cases with churn and all of them are correct. In the actual test data, there are 16 cases with churn. In other words, its precision is 6/6 (100%) and its recall is 6/13 (46.15%). Although our model did not catch all churn cases, it achieves 100% correctness regarding the churn cases that it predicted. With the amended data, the overall performance of the model is reduced as well as the prediction rate for churn. Nonetheless, it is still of interest because out of 14 churn cases in the testing data, our model predicts 14 churn cases and 13 of them are correct (e.g. both precision and recall are 13/14 - 92.86%). This amended data set consists of 161 (roughly 2% of the

Results/ratio	0.05	0.10	0.15
Total training data	504	839	1208
Total test data	58	95	137
Actual tests successful	43	79	121
Actual tests failure	15	16	16
Predicted tests successful	48	90	134
Predicted tests failure	10	5	3
Correct predictions	43	82	118
Failed predictions	15	13	19
Predicted tests success correct	38	78	118
Predicted tests failure correct	5	4	0
Correct ratio	0.74	0.86	0.86

Table 4.8: Global KnsSA applied on original DS3 with different churn and non churn ratios with $K = 3$

Results/ratio	0.05	0.10	0.15
Total training data	495	864	1212
Total test data	57	96	137
Actual tests successful	41	83	121
Actual tests failure	16	15	16
Predicted tests successful	36	93	135
Predicted tests failure	21	5	2
Correct predictions	34	84	121
Failed predictions	23	14	16
Predicted tests success correct	27	81	120
Predicted tests failure correct	7	3	1
Total windows processed	28215	84672	166044
Correct ratio	0.60	0.86	0.88

Table 4.9: Global window-KnsSA applied on original DS3 with different churn and non churn ratios with $K = 3$

original data set) churn cases and 10% of the non churn cases of the original data set.

The results in Tables 4.7, 4.8 and 4.9 show that local KnsSA outperforms global KnsSAs when applied to the churn data set. This might be caused by the fact that in customer behaviour sequences, only a set of special segments has strong influence on churn actions.

4.4.3 Results for Warning Technique LFD

The results of the experiments on analysing the relationship between loops and process outcomes using LFD applied to $DS1$, $DS2$ and $DS3$ are illustrated in the Tables 4.10, 4.11 and 4.12 correspondingly.

As the goal is to verify if there is a link between a specified pattern and the outcome, specifically there is a link between a specified loop and failure, minimum support and minimum confidence are varied in order to filter out loops which are not frequent and not interesting. When minimum support is raised, only loops with high frequency are considered. In general, there are no loops in the data sets $DS1$ and $DS2$ which have high support and confidence.

Total looped processes , Total processes Success ratio 72.88% Failure ratio 27.12%						
sid	eid	Outcome	Task	Loop	Support	Confidence
9	[1, 2]	F	A	L	0.31	1.00
13	[2, 3]	S	A	L	0.17	1.00
17	[2, 3]	S	A	L	0.31	1.00
23	[2, 3]	S	A	L	0.31	1.00

Table 4.10: Loops and corresponding process outcomes found by the LFD in DS1.

Total looped processes 99, Total processes 576 Success ratio 17.17% Failure ratio 82.83%						
sid	eid	Outcome	Task	Loop	Support	Confidence
416	[41, 42]	F	C	L	0.19	1.00
416	[1, 2]	F	B	L	0.21	1.00
421	[15, 16]	F	B	L	0.21	1.00
464	[17, 18]	F	B	L	0.21	1.00
471	[16, 17]	S	B	L	0.21	1.00
479	[34, 35]	S	B	L	0.21	1.00

Table 4.11: Loops and corresponding process outcomes found by the LFD in DS2.

Total looped processes 1135, Total processes 8080 Success ratio 87.66% Failure ratio 12.34%						
sid	eid	Outcome	Task	Loop	Support	Confidence
1042	[1, 3]	F	4	L	0.89	1.00
1043	[1, 2]	S	4	L	0.89	1.00
1044	[1, 3]	S	4	L	0.89	1.00
1046	[1, 2]	S	4	L	0.89	1.00
1047	[1, 2]	S	4	L	0.89	1.00
1048	[1, 2]	S	4	L	0.89	1.00

Table 4.12: Loops and corresponding process outcomes found by the LFD in DS3.

The experiments do not show the link between loops and failure as expected whilst applying LFD to $DS1$. It is understandable as the loop task is 'contacting customer'. Obviously, people who executed this process tried to provide good service by repeatedly contacting customers. In the case of $DS2$, the results of the experiments, in contrast, show the link between loops and failure. The data set $DS3$ is a special case, there is no link between loops and failure or success because in general there are loops in each customer behaviour sequence, loops in task 2 and task 4 have 84% and 89% support respectively.

We also started to investigate if cycles in a process instance have influence on the process outcome. The idea of this work as well as the principle are similar to the loop failure link technique. We look at IdLists to see if there is any task which occurs at least twice but non-consecutively in a process instance. The work is in progress.

4.5 Conclusions

KNNs as a classical data mining approach have been widely used for modelling and predicting customer behaviour. Continuing the research line which was set from the beginning, this chapter addresses some shortcomings of these predictive models, which occur when they are used for sequential data. We also propose some extensions to KNNs. These extensions were introduced in order to capture the temporal characteristics of the data and to profit from KNNs ability to deal with diverse data. Our extensions are tested on real business process data from a multinational telecommunications company and the experiments provide some interesting results. First, the influence of global and local algorithms change depending on the data employed. For example, in the $DS1$ dataset, there is not much difference between using global KnsSA and local KnsSA. However, for $DS2$, global KnsSA is more accurate and in the case of $DS3$ local KnsSA outperforms global KnsSA. Even though the highest performance when predicting process outcome of the proposed models is just 75%, it outperforms the original KNNs, which proves that it is important to use sequential data in our problem.

Of all experiments, churn prediction shows the most interesting result. As churn is a rare event (2% of the data consist of churn), in the testing data set, there are only around 15 churn cases. Our models correctly capture most of these cases and percentage correct of churn prediction is very high. In the case of local KnsSA with $K = 3$, applied to $DS3$, out of 14 churn cases, the model predicts that there are 14 churn cases and 13 of them are correct.

It is interesting to see how the link between loops and process failure changes from case to case (different processes). In $DS1$, a specific loop implies that it is likely the process instance will end up with success. In $DS2$, there are certain loops which lead to process failure with high probability.

This chapter confirms our initial point of view that it is hard to model diverse sequences in a generic way to predict the outcome and that it is important to use the temporal characteristics of the data. Hence, KNN is a good candidate because KNNs treat sets of similar sequences in the same way. Moreover, by combining sequence alignment and KNNs we can achieve better results since this helps to compare two process instances based on the ordered events themselves. This encourages us to adopt the strategy of first grouping data into similar groups and then dealing with them using suitable approaches, which will be the baseline of the next chapter.

Chapter 5

Sequential Clustering

Next process step prediction is an important problem in process analytics since it can be used in process monitoring to preempt problems, allowing excellent process execution. We use logfiles from a workflow system that records the sequential execution of business processes to train analytic models. Each process execution results in a timestamped event or task sequence. The main issue of analysing such event sequences is that they can be very diverse, e.g. they have different lengths, many unique tasks, starting tasks and ending tasks etc. and there can be a huge number of different sequences. We are looking for models that can effectively handle diverse sequences without losing the sequential nature of the data. Based on the results of the previous two chapters, our strategy is to cluster the data into different groups and build suitable mathematical models for the individual resulting clusters. Our approach is based on clustering event sequences into different groups. Each resulting group consists of similar sequences. The challenge is to identify a similarity measure that can cope with the sequential nature of the data. After clustering we build individual predictive models for each group. To realise these, we first employ K-means, a simple Euclidean space clustering algorithm, and extend it into a categorical-sequential clustering algorithm by combining it with sequential alignment as a distance measure. This is a different approach compared to the the model based techniques using hidden Markov models (HMMs) that are typically employed in clustering sequential data. Finally, we treat each resulting cluster separately by building variants of a Markov model of different orders, expecting that the representative characteristics of each cluster are captured.

5.1 Introduction

In order to achieve operational excellence, companies must run efficient and effective processes (Ruta and Majeed (2011)), (Tsui *et al.* (2005)). They must also be able to predict if processes will complete successfully or run into exceptions in order to intervene at the right time, preempt problems and maintain customer service. To understand the as-is processes that may be spread across a number of legacy systems, which in reality are often poorly documented, is helpful for companies.

It is a real challenge to build such models for business process data for many reasons. The two main reasons which we think are the most dominant.

- Business process data is sequential in nature.

- Business process data can be very diverse, especially legacy processes. Factors like degree of freedom whilst executing processes, large IT systems and external factors reflect on the data and the complexity is explicitly visible in the structure of the asynchronous data sequences (i.e. the workflow of the executed process).

As mentioned in the preceding chapters, even though there is a rich source of mathematical models in data mining, there are not many sequential approaches. Also, these existing sequential approaches are not very effective in solving this particular problem. Facing these two issues: complex sequential data and lack of effective sequential approaches, the approach investigated in this chapter is to 'divide' the process into smaller groups of tasks/steps and at each group, build a model accordingly. Such models in process mining are called local models (Trcka and Pechenizkiy (2009)), (van der Aaslt and Weijters (2004)). The shortcoming of these local models is that the links/coherence and the interaction between some events from different event logs are lost.

The idea of building local models is to deal with event logs/operational states separately in order to understand what is going on around these individual event logs or groups of event logs. This method is very effective in checking the conformance between the process model and the event logs. It also can improve as well as control the operation of the considered logs. However, we might miss the links and the interaction between events from different event logs. This is a shortcoming because in process data temporal characteristics are very important. We propose another strategy addressing the complexity and diversity of process data. It is to partition process data into groups of sequences of similar characteristics. The resulting groups are desired to be main prototypes of the process. Proposed mathematical models are chosen according to the properties of the sequences in the resulting groups. The fort point of this method is that it overcomes the drawback of the aforementioned problem embedded in local models: keep the form of the process (sequential data); discriminating and adequately dealing with different representative prototypes. The process data are divided differently with respect to these two methods (local models and clustered data models). This is illustrated in Figure 5.1.

The above motivates our choice to design a sequential clustering algorithm as we want to compare the sequences themselves (as in string comparison) to obtained subgroups of sequences which are similar in the way events have occurred. Existing sequential clustering approaches are found in the works of Smyth (1997), Garcia *et al.* (2009) etc. The principle of these approaches is to build a distance measure matrix by first, modelling the data sequences one by one then comparing the likelihood of a sequence fitting to a chosen model. Any probabilistic model can be used here to describe the data e.g. linear autoregressive, graph-based models etc. HMM based sequential clustering is the most common and has shown its great performance in certain fields where data consists of continuous and/or long sequences. To fit data sequences to descriptive models, the data is assumed to have some properties or prior probabilistic distribution. For example, to be able to apply HMMs, the data must be the independently distributed. That means, in this family of approaches, sequences are approximated and compared indirectly by investigating how well they fit to a model. It might be better to compare the sequences directly based on the events and the

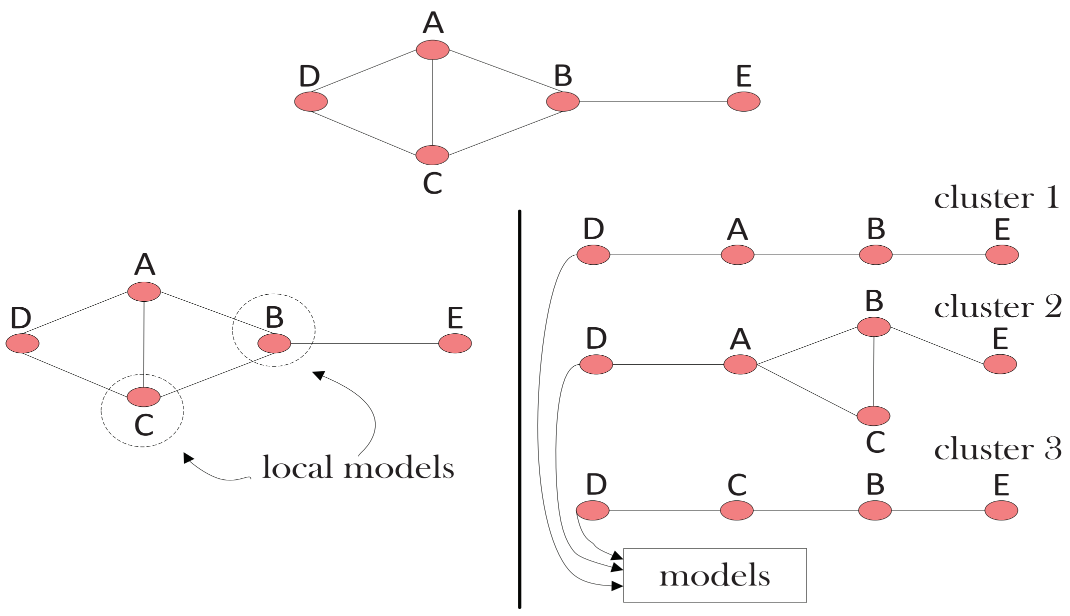


Figure 5.1: Data partitioned vertically (local models) and horizontally (sequential clustering)

order in which they occurred rather than to build a HMM for each sequence in our event sequence data, in particular for short length sequences.

To this end, we use sequence alignment to match all sequences pairwise and the outcomes of the matching are used as a similarity measure function. This similarity measure function is sequential and holds distance properties in metric space. This new measurement for sequential clustering compares sequences using events and their corresponding orders, not feature vectors. Another advantage of using sequence alignment as a similarity measure function is that the criterion for similarity can be changed by using local alignment or global alignment. It is important as for different data sets, different factors might have impact on the clustering. With local alignment, the most similar consecutive segments between sequences is the decisive factor and for global alignment, the similarity is based on the entire sequence. The framework is to establish a categorical-sequential variation of K means clustering to cluster the studied data first. A couple of hybrid Markov models, an extension to Markov models, are applied to each resulting cluster. Data used in the experiments is data from real business processes from a large telecommunications company.

The rest of the chapter is organised as follows: Section 5.2 presents the sequence alignment technique and introduces clustering and the proposed sequential clustering approach. Experiments and experimental results discussion take place in Section 5.3. Finally, Section 5.4 will conclude and draw a future research directions based on hints obtained from the former section.

5.2 K means Clustering Extension

5.2.1 Sequence Alignment

In sequence alignment both global alignment and local alignment are common and can be used to build distance matrices. These distance matrices are then used in K means clustering. The purpose is to find most suitable way to compare two sequences: is the total similarity is more important or a common consecutive segment most important in terms of the impact to the outcome of processes.

The main idea is to match sequences pairwise and the corresponding degrees of similarity are used as distance measure in the process of partitioning data points (sequences) into clusters in the following section. To compare any two objects, a similarity measure or a distance measure function is needed. As we are working on event sequences which are categorical variables, a distance measure function from biology is adopted, as well as sequence alignment technique.

Given two sequences, the matching procedure of sequence alignment includes the actions of deleting, inserting events from both sequences with regard to the best matching score. Whenever a deletion or insertion is taken place, there is a corresponding penalty added to the total score of matching. There are two (three) important matrices associated with sequence alignment algorithms (depending on local or global alignment is used). These matrices and the principle of global and local alignments are presented in detail in Section 2.3, Chapter 2.

5.2.2 K means Clustering

Given the number of clusters K, first, the corresponding centers are initialised. These centres can be data points randomly taken from the available data set or taken in a way that these centres are different from each other etc. This depends on users. Second, each data point is assigned to the closest cluster based on the distance between the data point and K centres. The data type decides which distance (similarity) function should be used.

Once all the data points are assigned, a new centre for each cluster is determined. For numerical data, such centres are mean values of the elements in the corresponding clusters. Median can also be used here instead of mean and K means becomes K medians. The procedure of assigning data points to clusters and recomputing centres is repeated until it converges.

5.2.3 K means Variant for Event Sequences

The sequence matching degree presented earlier is used as similarity measure in our proposed K means clustering which is called SA based clustering throughout this chapter. We directly use ordered events of data sequences to compare sequences because there is no mean value for each cluster. This renders the procedure of choosing the next iteration center of a cluster as a mean value impossible. We choose one sequence in each cluster from which the total distance to other members of the cluster is smallest as the new centre.

Given a set of N event sequences, first, all possible pairs of different sequences in the data set are matched using the sequence alignment technique. The resulting scores are stored in a square matrix $(d_{ij})_{N*N}$, where $i, j = \{1, \dots, N\}$. The values of elements of the matrix $(d_{ij})_{N*N}$ are the corresponding highest scores in the score matrix (see Section 2.3).

We start with $K = 2$ and increase this value of K until the performance of the clustering reaches a specific threshold to ensure the goodness of the clustering. This threshold is chosen by users. Sequences are assigned to K clusters based on the degrees of similarity taken from matrix $(d_{ij})_{N*N}$. Once all the sequences are distributed, new centers of the K clusters are recomputed as follows:

$$c_{jnew} = min_j \Sigma_{i,i,j \in C_j, i \neq j} d_{ij}^2, \tag{5.1}$$

where c_{jnew} is the new center of j^{th} cluster C_j. The procedure of updating new centres is repeated until it converges. The idea of choosing a new centre from data points in the corresponding cluster was used in the GRGPF algorithm (Rajaraman and Ullman (2011)). It is a representing clustering approach in non-Euclidean space. In this algorithm, the new centre is the data point in the cluster which has the minimum sum of the squared distances from itself to other points in the cluster. This minimum is called ROWSUM.

Algorithm 5 −**Proc** : $findMaxCommonLength(src, dst)$

```
1: $n = length(src)$
2: $m = length(dst)$
3: $h = zeros(n, m)$
4: for all $i = 1 : n$ do
5:     for all $j = 1 : m$ do
6:         $hh_{i_1 j_1} = (i > 1) and (j > 1)\ ?\ h_{(i-1),(j-1)} : 0$   ▷ value of $h_{(i-1),(j-1)}$ if i and j are valid indexex, 0 otherwise
7:         $hh_{i_1 j} = (i > 1)\ ?\ h_{(i-1),j} : 0$
8:         $hh_{i j_1} = (j > 1)\ ?\ h_{i,(j-1)} : 0$
9:         $hs = hh_{i_1 j_1} + ((src_i = dst_j)?1 : 0)$   ▷ Either 0 or 1
10:        $hd = hh_{i_1 j} - Wd$   ▷ Adjust the value by a constant Wd
11:        $hi = hh_{i j_1} - Wi$   ▷ Adjust the value by a constant Wi
12:        $h_{ij} = max(max(hs, hd), max(hi, 0)))$   ▷ max of (hs,hd,hi)
13:    end for
14: end for
15: $findMaxCommonLength = max(h_{ij})$
```

Algorithm 6 −**Proc** : $distanceMatrixSA(processes)$

```
1: $n = length(processes)$
2: $d = zeros(n, n)$
3: for all $i = 1 : n$ do
4:     for all $j = 1 : n$ do
5:         $process1 = processes_i$
6:         $process2 = processes_j$
7:         $d_{ij}$ = findMaxCommonLength(process1.tasks, process2.tasks)   ▷ Construct a matrix of max common length between processes
8:     end for
9: end for
10: $distanceMatrixSA = d$
```

Algorithm 7 −**Proc** : $distanceMatrixHMM(processes)$

```
1: $n = length(processes)$
2: $d = zeros(n,n)$
3: $l = sum(length(processes))$
4: $Q = \frac{1}{n} * l$
5: for all $i = 1 : n$ do
6:     $prior1_i = normalize(rand(Q,1))$
7:     $transmat1_i = mk\_stochastic(rand(Q,Q))$
8:     $obsmat1_i = mk\_stochastic(rand(Q,0))$
9: end for
10: for all $i = 1 : n$ do
11:     for all $j = 1 : n$ do
12:         $process1 = processes_i$
13:         $process2 = processes_j$
14:         $loglik_i = dhmm\_logprob(process1.tasks, prior1_j, transmat1_j, obsmat1_j)$
15:         $loglik_j = dhmm\_logprob(process2.tasks, prior1_i, transmat1_i, obsmat1_i)$
16:         $d_{ij} = \frac{1}{2} * (loglik_i + loglik_j)$
17:     end for
18: end for
19: $distanceMatrixHMM = d$
```

Algorithm 8 −**Proc** : $assignSequencesToClusters(k, n, disMatrix, centers, processes)$

```
1: $clusterIndexes = zeros(n)$
2: $support = zeros(n)$
3: $cluster = empty(n)$
4: for all $i = 1 : k$ do
5:     $cluster\{i\} \leftarrow centers_i$        ▷ clusteri contains the indexes of processes
6:     $clusterIndexes(centers_i) = i$        ▷ clusterIndexes indicates the cluster index each process belongs to
7: end for
8: for all $i = 1 : n$ do
9:     if $clusterIndexes_i == 0$ then        ▷ Check only unassigned process
10:         $index = j|min(disMatrix(i, centers_j))$        ▷ Find the cluster where the central point is closet to this process
11:         $clusterindex \leftarrow i$        ▷ Assign this process to that cluster
12:         $clusterIndexes_i = index$
13:     end if
14: end for
15: for all $i = 1 : k$ do
16:     $clusters \leftarrow cluster\{i\}$
17:     $support_i = length(cluster\{i\})$        ▷ Support of a cluster is its length
18: end for
19: $assignSequencesToClusters = [clusters, support]$
```

Algorithm 9 $-$**Proc** : $reComputeCenters(method, k, n, distanceMatrix, oldCenters, clusters, processes)$

```
1: $newCenters = []$
2: $distances = []$
3: for all $i = 1 : k$ do
4:     $temp = (method = HMM)$ ? 999999999 : -999999999    ▷ indicate upper/lower value depending on method
5:     for all $j = 1 : length(clusters_i)$ do
6:         $sigma = 0$
7:         for all $kk = 1 : length(cluster_i)$ do
8:             if $kk \neq j$ then
9:                 $sigma = sigma + sqrt(distanceMatrix(clusters_{i_j}, clusters_{i_{kk}}))$    ▷ Calculate distance between this process and all other in the same cluster
10:            end if
11:        end for
12:        if $((method = HMM)$ and $(temp > sigma))$ or $((method = SA)$ and $(temp < sigma))$ then
13:            $temp = sigma$    ▷ Select new center point based on the minimal/maximum distance, depending on method
14:            $newCenter = clusters_{i_j}$
15:        end if
16:    end for
17:    $newCenters_i = newCenter$
18:    $distances_i$ = sqrt(distanceMatrix(newCenter, $oldCenters_i$))    ▷ Calculate the distances between old & new center points
19: end for
```

5.3 Evaluation

5.3.1 Research Path

Graph and Business Processes

If we represent our process data in a form of a graph, these models can then be used at different vertices of the represented graph. A hybrid predictive mathematical model is required to be flexible to be able to deal with different patterns existing in process data. Theoretically, it is possible to retrieve the process logic by mining rules at each node of the diagram using different mathematical models. If we can build a detailed model with high accuracy, not only online operational support is achieved but also the process logic recovery task can be attempted. However, this is not the case in practice as explained in Chapter 1.

The final model should have the form of a graph. Ideally each vertex should contain the probabilities of MMs corresponding to the task in the vertex, the rules and a decision tree. However, since the training data is complex and diverse, there will be new vertices or edges when a given sequence is executed even when the model has been fully trained. Hence, the question is of whether and when to insert a new vertex or a new edge to the graph? The issue arises because of the changes of the whole parameters (probabilities, rules) of the existing model which occur whilst adding new vertices or edges to the graph. Therefore, a suitable threshold is needed in order to decide when to add them.

To investigate the possibility and circumstances of the success of the framework, we

Algorithm 10 $-$**Proc** : *main*

```
 1: $n = length(processes)$
 2: $method = HMM/$
 3: $disMatrix = (method = HMM)\ ?\ distanceMatrixHMM(processes) : distanceMatrixSA(processes)$    ▷ Calculate the distance matrix based on method
 4: $clusDistorsion = 0$
 5: $k = 1$
 6: repeat
 7:     $k = k + 1$    ▷ At each loop increase the number of clusters by 1
 8:     $centers = first(k, randperm(n))$    ▷ Randomly select center points for k clusters
 9:     $totalLoop = 0$
10:     repeat
11:         $[clusters, support] = assignSequencesToClusters(k, n, disMatrix, centers, processes)$    ▷ Assign processes to k clusters
12:         $[newCenters, newDistances] = reComputeCenters(method, k, n, disMatrix, centers, clusters, processes)$    ▷ Find the new centers for all clusters
13:         $unchanges = findDifferences(newCenters, centers)$
14:         if $unchanges < k$ then
15:             $centers = newCenters$
16:             $countDistance = count(i|newDistances_i \geq \frac{length(centers_i)}{4}))$    ▷ Find clusters where distance between old/new centers is significant
17:         end if
18:     until $unchanges = k$ or $\frac{countDistance}{k} > THRESHOLD_1$    ▷ Stop if we have done k loops or distance count is greater than a threshold
19:     $clusDistorsion = 0$
20:     for all $i = 1 : k$ do
21:         for all $j = 1 : support_i$ do
22:             $d = disMatrix(centers_i, clustersi_j)$
23:             $clusDistorsion = d * d + clusDistorsion$    ▷ Calculate club distorsion
24:         end for
25:     end for
26: until $k \geq n$ or $clusDistorsion > THRESHOLD_2$    ▷ Stop if we have done n loops or club distorsion is greater than a threshold
```

first study the possibility of presenting a process as a graph and the complexity of the obtained graph if found.

It is relatively straightforward to represent business process workflow as a diagram or a graph. Each task (or event) in a process sequence can be assigned to a vertex in such graph, which, in turn, can be constructed as follows: all the unique tasks in the given dataset form the vertex set of the corresponding graph. Each pair of consecutive tasks in any process instance is then represented by an edge.

As the data is typically diverse, it is common when a given process sequence has tasks (as well as edges) which have never occurred in the existing graph. The problem here is that adding a new vertex or an edge to the existing graph causes trouble. This statement is made based on the argument that all the related probabilities, and/or coefficients, of the related models will change accordingly. To get insight of the issue, the frequency of new vertex or edge occurrence needs to be investigated. The results are then used to determine how often a vertex should be added. For example, if the support of the vertex or edge is too small, its influence on the overall graph can be neglected.

We have conducted a number of experiments on the dataset DS1 to determine the frequencies of newly occurring vertices/edges. Specifically, a relatively small size of the data set is selected, the corresponding presented graph is then constructed. Then, more data is added and new vertices and edges need to be inserted into the existing graph. The fraction of inserted vertices and edges is recorded. This procedure is repeated until the whole data set is processed. We are also interested in finding out how big the trained data set is required so that the rate of new vertices and edges can be considered negligible[1]. All the work was done in Matlab except the graph visualisation. We borrow the visualisation tool from Java. Codes from Jung library are used directly.

The first experiment was carried out with 20% of the extracted dataset. The obtained graph is shown in Figure 5.2.

Each time 10% of the extracted dataset is added to the previous data sample in the previous experiment. The next graph was of 40% of the data sample and is illustrated in Figure 5.3.

In Figures 5.4 and 5.5, 80% and 90% data sample graphs are illustrated.

It is quite obvious, from Figures 5.2-5.5, that when a small portion of the extracted data set is considered as the trained data, the obtained graph is not generic enough. Any time when 10% of the given data set is added into the trained data set, there are a considerable number of new vertices appeared, around 30% of the existing vertices. Not until the trained data set reaches $60\% - 70\%$ of the population of the original dataset then the rate of new vertices decreases to $10\% - 15\%$.

Decision Trees at Graph's Vertices

The second step is to build predictive models at each vertex of the obtained graph to predict the next process steps. Here, we chose decision trees for experiments because of their competences (e.g. easy to interpret, simple to build). This is to investigate how hard it

[1]The figures in this section only represent a small sample data set since it is not possible to display all the data.

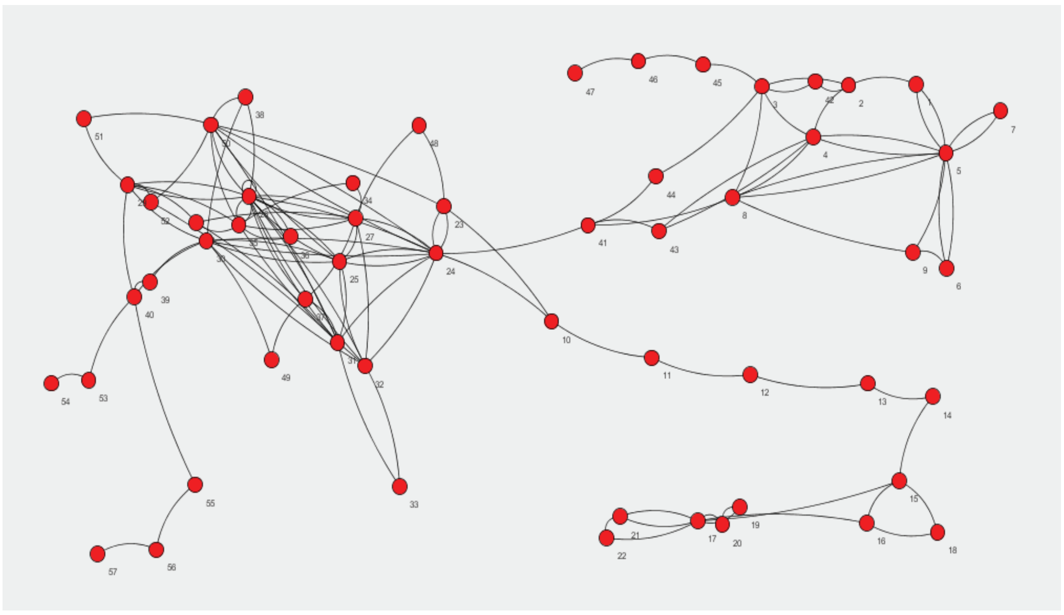


Figure 5.2: Graph presents 20% of the extracted dataset DS1.

is to have accurate next process step prediction.

To build such decision trees, tasks and process attributes are used as criteria for tree splitting procedure. Here, we chose gain ratio to measure the goodness of the split. A tree is started with a root which is a dummy node, it is then developed by a splitting procedure. At each tree node, available attributes are tested and the one with highest gain ratio is then selected as the splitting criterion of the node. The node is then split into child nodes using the obtained criterion. These trees are limited to have at most 5 levels as our data does not have enough useful attributes. The support of each node is set to be n, a specified threshold chosen by the author. If the support of the node is smaller than n then that node is pruned.

Also we do not want to have the overfitting problem in which the tree is too similar to a diagram which presents the data. There are likely to be paths/ leaves with very small probabilities. The trees are then stored at corresponding vertices of the process presenting graph as illustrated in Figure 5.6. The Jung library allows us to visualise the graphs and trees. In the visualisation of the decision trees, our decision trees are hidden, they only occur whilst we point the mouse at a vertex of the given graph.

From the available data set it can be seen that for each attribute there are not many corresponding values. Intuitively, this property of the data is not very helpful since it is not reliable to base on to split the trees.

We have seen that building graphs from diverse data results in unstable output and the obtained decision trees do not have the ability of providing useful rules to predict next process steps (this would be process logic recovery). As an attempt to overcome this issue, we investigate clustering the data into different groups of similar sequences. Hence, instead of one graph we have several graphs which are expected to be representative prototypes. Second, different mathematical models are used for the resulting individual clusters.

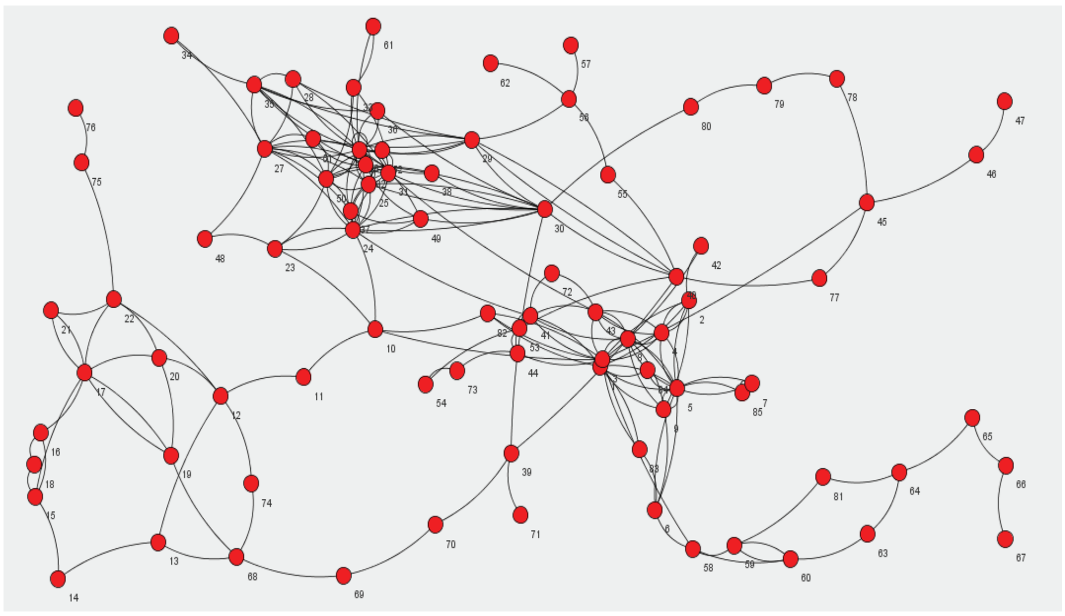


Figure 5.3: Graph presents 40% of the extracted dataset DS1.

5.3.2 Experimental Results

We carried out a number of experiments based on records from two real processes ($DS1-2$) from the same telecommunications company providing the data for the earlier chapters.

To benchmark our proposed clustering we use HMM based sequential clustering:

- *HMM based clustering*: Each data sequence is described by one HMM. The distance between any pair of sequences is computed based on the log-likelihood of fitting a sequence to the descriptive HMM of the other sequence. Once the distance matrix is built, data sequences are clustered using K-means clustering. One remark is that whilst computing the new centre of a cluster using Equation 5.1, d_{ij}^2 should be replaced by d_{ij} as the distance matrix obtained by HMM based method consists of elements of negative values. We use an available HMM toolbox in Matlab to develop the HMM based clustering.

- *SA based clustering*: The distance matrix is obtained by matching all possible pairs of sequences. The framework is then analogous to the former clustering.

The final goal of clustering business process data in this work is to improve the performance whilst predicting next process steps. We use Markov models and one of its extensions to predict the next process step in the experiments. The performance of these models applied to clustered data are used to evaluate the clustering strategy. In other words, they are proofs for verifying the impact of the strategy on the predictive capability.

- *MM - Markov Model*: In order to find the next task following the current one, we build transition matrices of different order Markov models. For example, with first order Markov model, from the historical data, we know that tasks following A belong to the set $\{C,D,E\}$ and the number of unique tasks. We compute the probabilities of

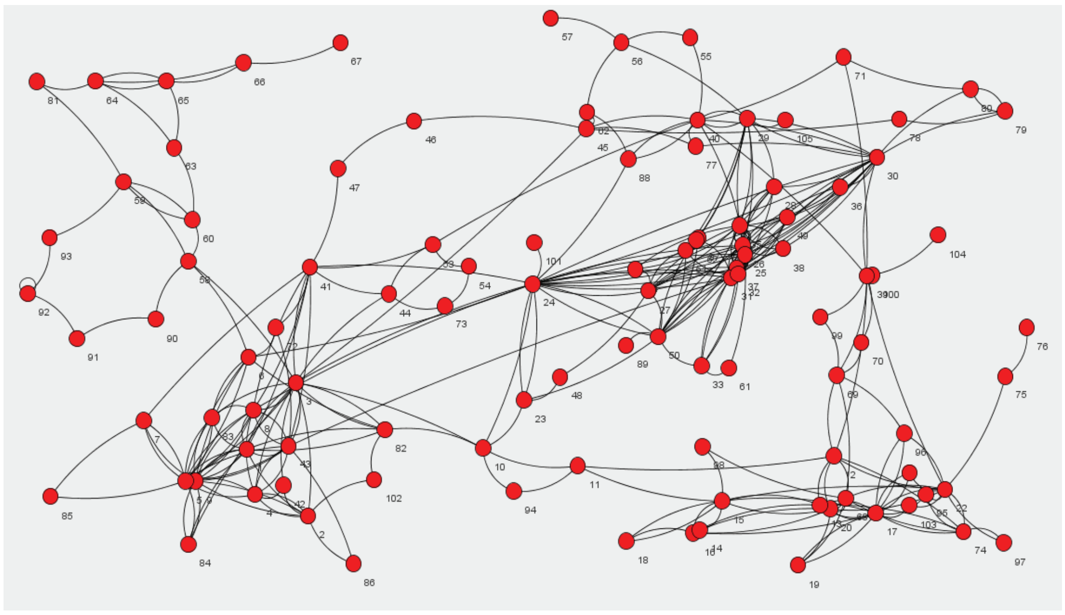


Figure 5.4: Graph presents 80% of the extracted dataset DS1.

jumping from task A to other unique tasks, by counting the number of occurrences for each unique task then dividing to the number of transitions (from task A to all the unique tasks) (Deshpande and Karypis (2004)). In this example, only the transitions from A to C, D and E have probabilities greater than 0, the ones for other unique tasks (if they exist) are equal to 0. The resulting probabilities form one row in the transition matrix.

- *MSA - Hybrid Markov Models*: Here, a default prediction improvement module is added to higher order Markov models to obtain better accuracy. The default prediction is improved by comparing a new pattern to all the patterns from the transition matrix using sequence alignment. The most similar pattern found from the matrix is used as a substitution for the given one to improve the prediction.

Our aim is to improve the accuracy of the prediction on the original data by clustering the data into groups of sequences with similar characteristics. Each group requires a suitable predictive model. To verify if we can cluster the data into such groups and if we can improve the accuracy by finding a suitable model for each group, we first present the performance of MMs and MSAs applied to $DS1$ and $DS2$:

The results shown in Figures 5.7 and 5.8 show that the performance of these predictive models are quite low, only about 25%. We then introduce the performance of the same models applied to the clustered data of the same dataset. Both sequential methods, HMM based and our proposed clustering (SA based), are implemented to build distance matrices. Sequential K means HMM based and SA based are used to cluster $DS1$, $DS2$ into appropriate numbers of clusters. Figures 5.9[2] and 5.11 illustrate the performance of MMs and MSAs applied to 3 and 6-cluster-$DS1$, which we obtained by SA based clustering,

[2]MMci: Markov models applied to i^{th} cluster obtained by applying SA based clustering to DS1.

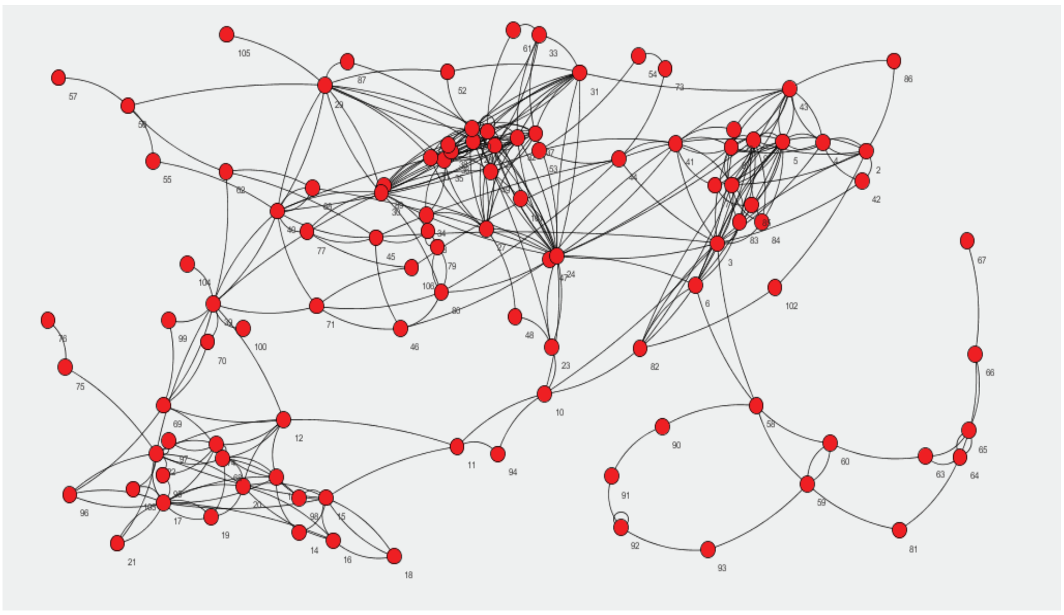


Figure 5.5: Graph presents 90% of the extracted dataset DS1.

respectively and Figures 5.10[3] and 5.12 illustrate the same with HMM based clustering.

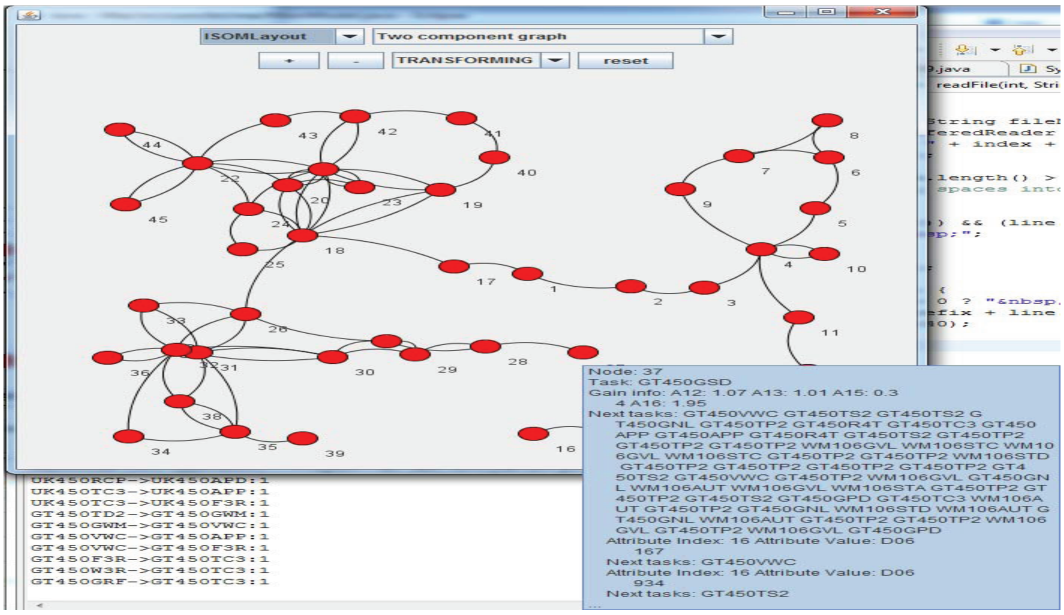
As can be seen, in the case of K means SA based with $K = 6$, the MMs and MSAs applied to cluster 5 have significantly higher performance. The highest performance is 78.69 % (third order MSA) which is almost four times greater than the performance of the same model applied to the original $DS1$. It is about 2.5 times higher compared to the performance of the other clusters. Applying MMs and MSAs on clusters 4 and 6 provides better accuracy (27.27% and 28.94% respectively) than applying these on the original $DS1$ (23.76%).

In contrast, there is not much difference in terms of performance of these predictive models whilst applied to the original dataset $DS1$ or the clustered data using K means HMM based. With $K = 3$, in both cases HMM based and SA based, there is not much change in terms of the accuracy of the next process step prediction regarding the accuracy of the same models applied on the original $DS1$.

The significant accuracy improvement in cluster 5 is the proof for our initial intuition that if we manage to have subsets of data which consist of similar sequences then there exists a suitable model which performs well in each subset. Also, these models perform better in certain subsets than others and they absolutely perform better when applied to the clustered data than to the dataset in whole (before being clustered). It indirectly proves that our proposed sequential clustering approach performs well when similar sequences are naturally clustered. Our SA based clustering is more suitable for this type of data than the common HMM based clustering.

In the case of dataset $DS2$, there is a slight improvement in terms of prediction accuracy after clustering the data for both clustering approaches. The highest performance of fourth order MSAs is about 27% applied to cluster 2 obtained by SA based and HMM based

[3]MMci: Markov models applied to i^{th} cluster obtained by applying HMM based clustering to DS1.



```
String file
feredReader
" + index +
;
.length() >
 spaces int
) && (line
sp;";
;
 {
 0 ? "&nbsp.
efix + line
0);
```

```
UK450RCP->UK450APD:1
UK450TC3->UK450APP:1
UK450TC3->UK450F3R:1
GT450TD2->GT450GWM:1
GT450GWM->GT450VWC:1
GT450VWC->GT450APP:1
GT450VWC->GT450F3R:1
GT450F3R->GT450TC3:1
GT450W3R->GT450TC3:1
GT450GRF->GT450TC3:1
```

```
Node: 37
Task: GT450GSD
Gain info: A12: 1.07 A13: 1.01 A15: 0.3
   4 A16: 1.95
Next tasks: GT450VWC GT450TS2 GT450TS2 G
   T450GNL GT450TP2 GT450R4T GT450TC3 GT450
   APP GT450APP GT450R4T GT450TS2 GT450TP2
   GT450TP2 GT450TP2 WM106GVL WM106STC WM10
   6GVL WM106STC GT450TP2 GT450TP2 WM106STD
    GT450TP2 GT450TP2 GT450TP2 GT450TP2 GT4
   50TS2 GT450VWC GT450TP2 WM106GVL GT450GN
   L WM106AUT WM106GVL WM106STA GT450TP2 GT
   450TP2 GT450TS2 GT450GPD GT450TC3 WM106A
   UT GT450TP2 GT450GNL WM106STD WM106AUT G
   T450GNL WM106AUT GT450TP2 GT450TP2 WM106
   GVL GT450TP2 WM106GVL GT450GPD
  Attribute Index: 16 Attribute Value: D06
     167
  Next tasks: GT450VWC
  Attribute Index: 16 Attribute Value: D06
     934
  Next tasks: GT450TS2
...
```

Figure 5.6: Graph with a tree built for a vertex in the considered graph.

clusterings comparing to 20% to whole $DS2$. The performance of the first to fifth order MMs and MSAs applied to clusters 1 and 2 obtained by clustering $DS2$ in the case SA based clustering are illustrated in Figure 5.13.

When clustering $DS2$ using the two methods, only two clusters are formed. When we decrease the goodness of the clustering, more clusters are obtained, however, many of them have very low populations. The results of the experiments on clustered $DS1$ and $DS2$ show that different clusters need different predictive models. Higher order MMs and MSAs are especially good for data sequences in cluster 5 generated from $DS1$ using sequential K means clustering with $K = 6$. None of these models works well on other clusters from both data sets $DS1$ and $DS2$. It is sensible to say that the experimental models are not good for clustered data in the case of $DS2$ as this data set is relatively small.

In summary, the performance of our predictive models are improved for all datasets. The improvements are varied from dataset to dataset, and from cluster to cluster but there is always an improvement.

5.4 Conclusions

In order to deal with representative process prototypes differently, we first attempt to cluster process data into different groups of similar sequences. Such data consists of discrete symbolic sequences. After studying a number of available sequential clustering approaches, in this chapter we introduced a new sequential clustering approach which is suitable for business process data. We also use the common HMM based sequential clustering in order to verify performance improvements for our proposed approach. We then use predictive models to predict next process steps. We significantly improve the next process step prediction in one cluster of one of the used data sets. This implies the data has been successfully clustered in a natural way and proves our strategy right. Moreover, the performance of chosen

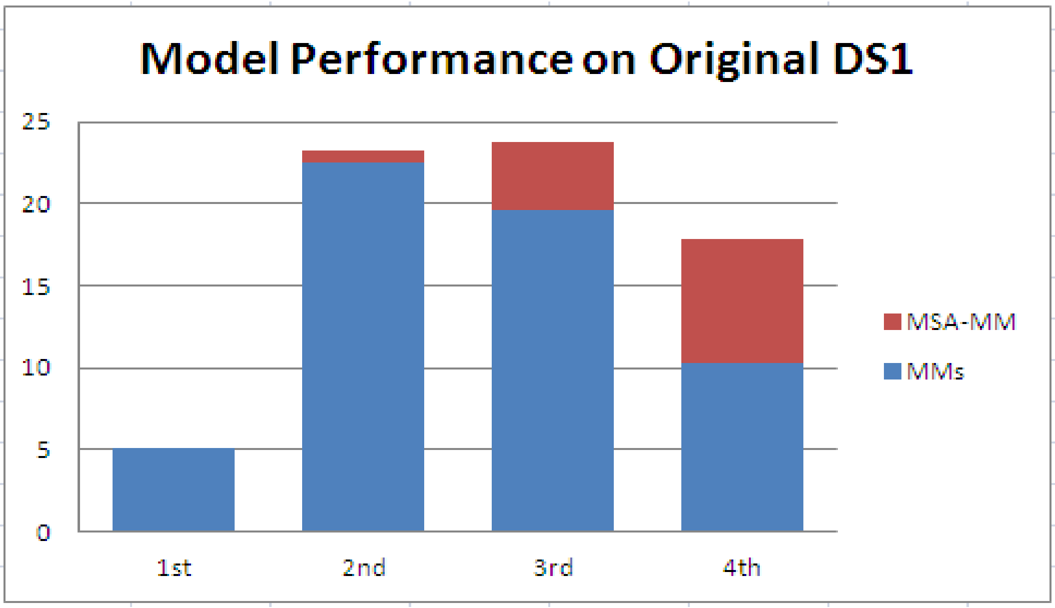


Figure 5.7: Percentage of correct process next step predictions of MM and MSA using dataset DS1.

predictive models applied to clustered data obtained by using SA based clustering is better than the one using HMM based clustering.

The experimental results encourage and motivate us to continue and extend our work. Future work will explore different predictive approaches to profit from their abilities in clustered data. We are ultimately interested in recovering the process logic and our recovered process can be the combination of a number of representative prototypes.

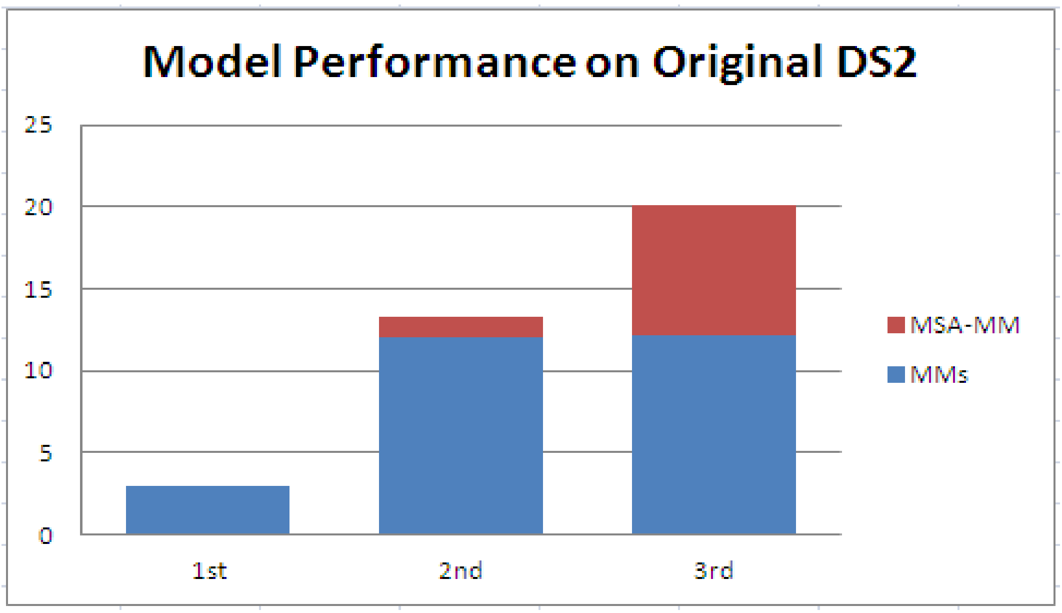


Figure 5.8: Percentage of correct process next step predictions of MM and MSA using dataset DS2.

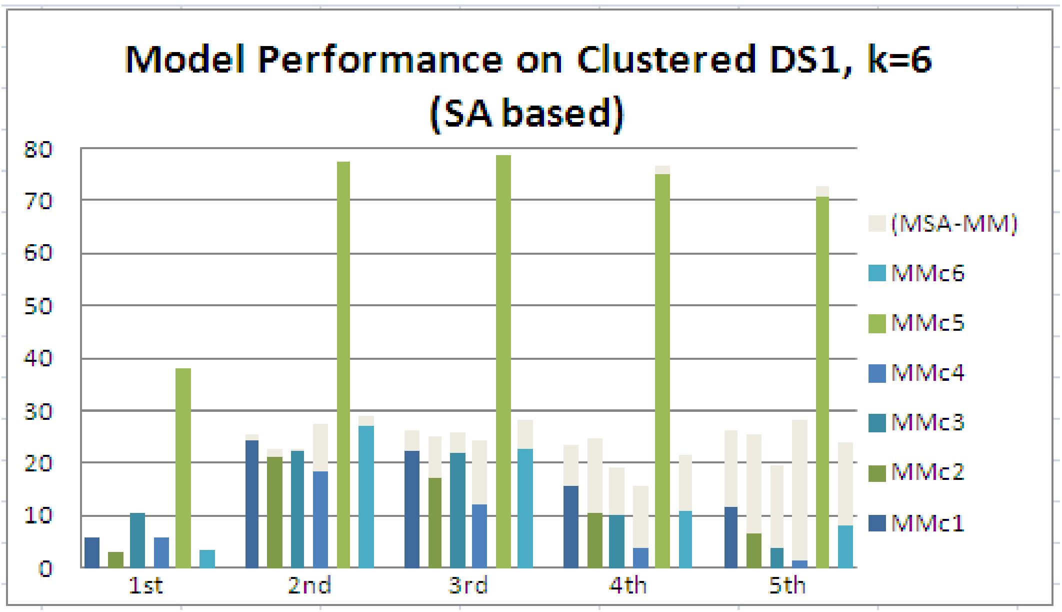


Figure 5.9: Percentage of correct next process step predictions of different order MMs and MSAs using 6 clusters obtained by applying SA based clustering to the data set DS1.

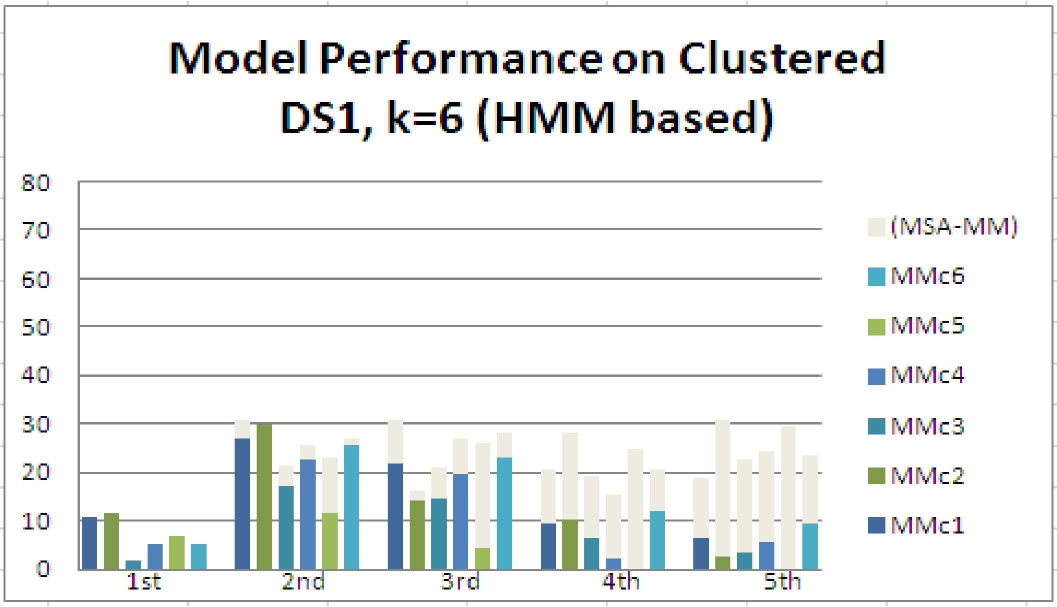


Figure 5.10: Percentage of correct next process step predictions of different order MMs and MSAs using 6 clusters obtained by applying HMM based clustering to the data set DS1.

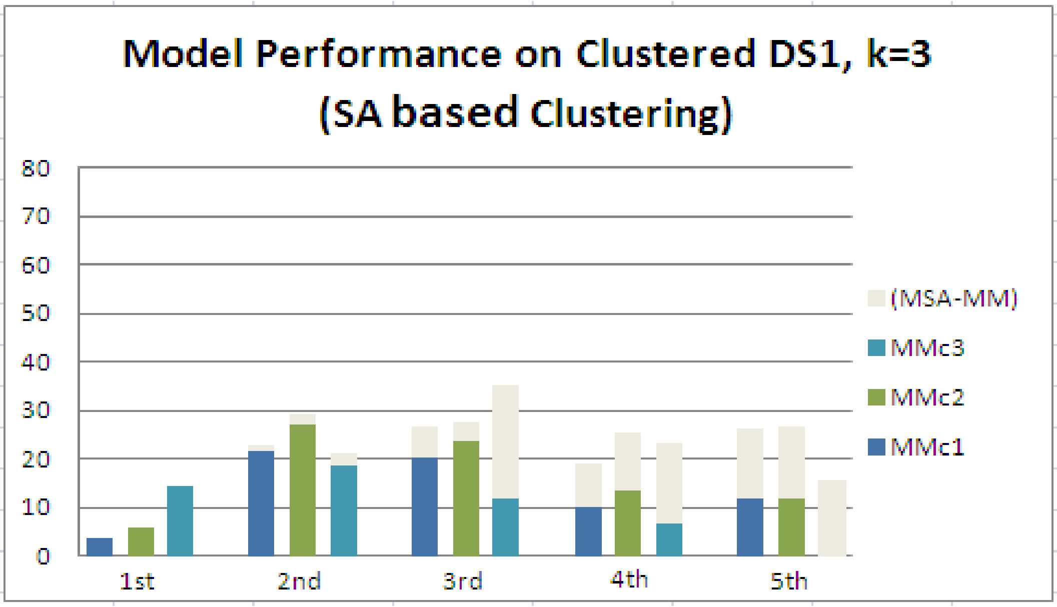


Figure 5.11: Percentage of correct process next step predictions of different order MMs and MSAs using 3 clusters obtained by applying SA based clustering to the dataset DS1.

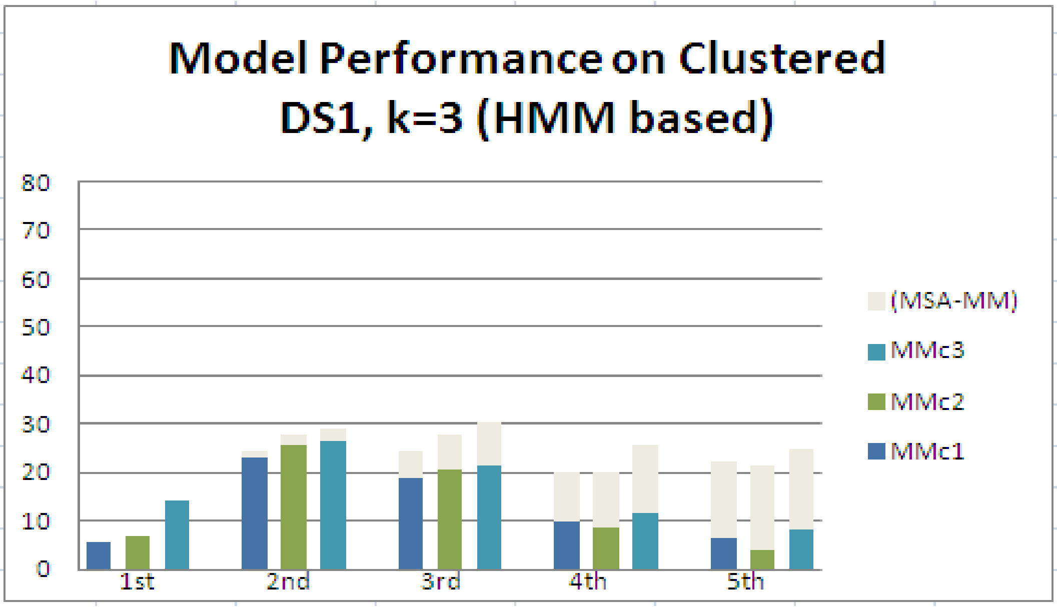


Figure 5.12: Percentage of correct next process step predictions of different order MMs and MSAs using 3 clusters obtained by applying HMM based clustering to the dataset DS1.

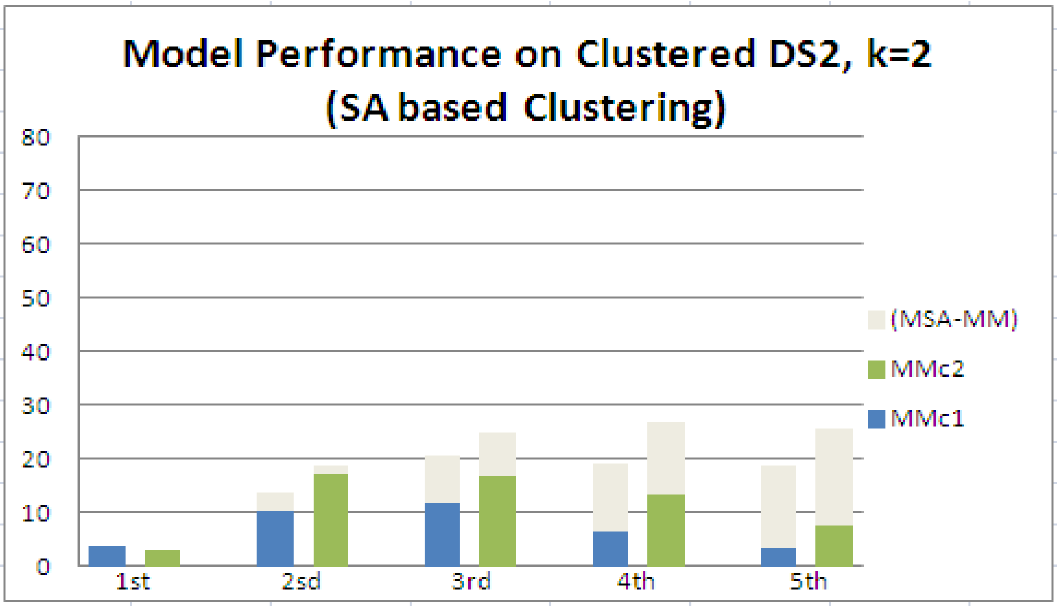


Figure 5.13: Percentage of correct next process step predictions of different order MMs and MSAs using 2 clusters obtained by applying SA based clustering to the dataset DS2.

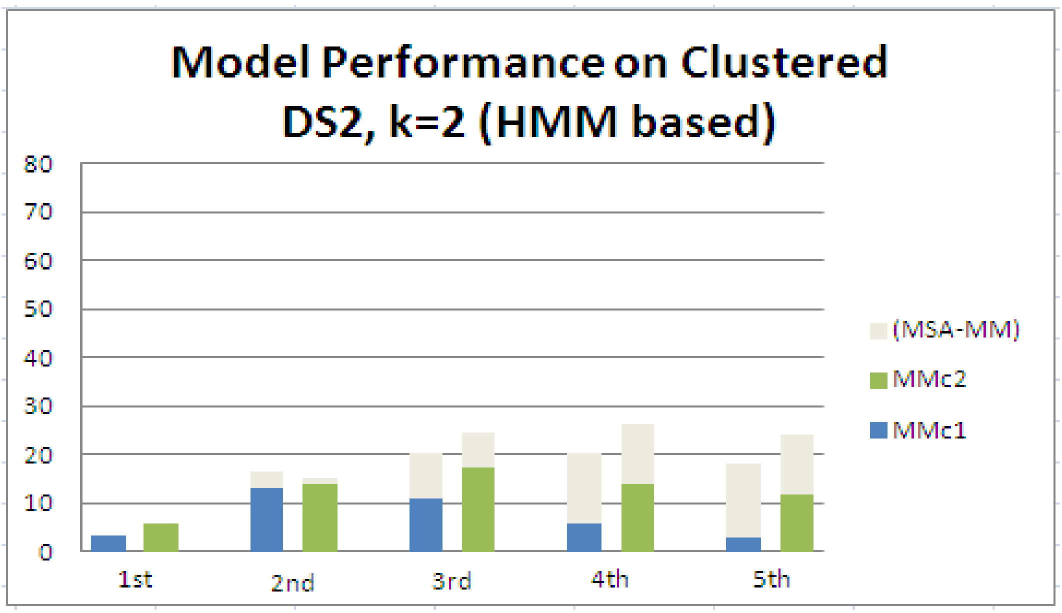


Figure 5.14: Percentage of correct next process step predictions of different order MMs and MSAs using 2 clusters obtained by applying HMM based clustering to the dataset DS2.

Chapter 6

Conclusions and Future Work Directions

6.1 Project Summary

The main goal of this project was to build and evaluate a class of hybrid mathematical models which are accurate and flexible enough for the emerging area of online operational support within process mining. The purpose of implementing the robust operational support is to overcome various issues with practical process execution to achieve excellent execution contributing to efficient and effective processes in large companies.

Understanding the background for this problem including motivation, problem statement, challenges and related works, is essential. Once we have the overview of the problem, deep research needs to be done in order to be able to find the research directions for the project. There are two possibilities to improve process performance, one is to support the redesign phase aiming for excellent processes; the other is to online support process operation by preempting problems thus being able to put companies in an aware position/attitude and suggesting strategies to resolve such problems. The latter aspect of process mining improves process performance by saving cost for the company, time spent and possibly improving customer satisfaction as well. Despite their important role, predictive models, which support process operation, do not get worthy attention from researchers in the field. Predictive analysis in process mining is immature. There are only a few works on this topic and most of them are very basic. Having reviewed the research area, we are motivated to choose the predictive sequential analysis in our project for these two reasons, immature area and online support. Another crucial factor which has strong impact on our decision on the project topic is that in fact, most companies cannot afford tailored autonomous BPMS for individual processes. Being a lead multi telecommunications company with long history, BT has a large number of legacy processes which pass from one IT system to another. It is almost not possible to start from the scratch to build up new processes with autonomous BPMSs and get rid of the legacy elements.

Process data is diverse and sequential, it consists of asynchronous event sequences, hence there seems to be no single approach in classical data mining that is capable to solve the problems stated. In addition, online support requires predictive models with high accuracy and low running times. The running time constraint adds more difficulty to our task as the data can be very large.

Besides the aforementioned difficulties, we also have the advantage of being able to touch real process data and work with people who are involved with these processes. This helps to direct our research together with the theoretical strategy.

Having set clear project objectives, the aim is then to identify data mining/machine learning techniques which could be useful for attaining these objectives. As process data is sequential in nature, we focus on sequential approaches expecting them to cope well with problems stated in our project's objectives and aims. A thorough literature review into sequential methods was carried out, some non sequential methods in data mining are also investigated due to their competences in dealing with diverse data and which are adaptable to sequential data by coupling with other techniques, for example, sliding window. Relating to sequential analysis there are three main categories of approaches. The first approach is the probabilistic family with approaches including hidden Markov models, Markov models and Gaussian processes. Finding sequential patterns in very large datasets is a rising topic. Temporal association rules has shown its power in various fields, starting from marketing (basket analysis). Sequential rule mining is the second category. The last one among these three main categories was listed as the logit family. However, due to the characteristics of our problems, we did not go towards that direction. Dealing with the diversity of process data, the strategy of dealing with alike entities (event sequences) similarly and adequately is opted. Therefore, sequential clustering and classification are the focus instead of logit approaches. In Chapter 2, some approaches from different categories, which may perform well in solving our problems, are studied in order to select the best.

The first priority criterion for choosing data mining approaches is the sequential property. The second one is about being able to cope with data diversity. One important factor of note is that our sequences are composed of events, hence, approaches which can only deal with numerical variables are not relevant. In order to conserve the occurrence order of events of process sequences and capture such order, we need a sequential approach which can deal with sequential and diverse data as well as categorical (symbolic) data. We aim to design a hybrid model which is the combination of different mathematical models and each of these models contributes its advantages to different perspectives in the stated problem. To be able to keep the event occurrence order of our process data and to deal with them naturally and sequentially, we adopt the sequence alignment technique from biology. This technique helps us to match event sequences in a pairwise fashion. The matching of any two given sequences has an outcome which is the similarity degree between them.

The probabilistic methods, specifically Markov models and their extensions, are presented in Chapter 3. The proposed approach is to predict next process steps which also means recovering the process logic if the model has high accuracy. Initially, higher order HMMs are candidates for our problem however, it is not simple to generate higher order HMMs. Therefore, we investigated Markov models. To make sure that we do not loose the accuracy of the predictive performance by opting to the simpler solution, the accuracies of HMMs and MMs when applied to our process data are compared. The results show that HMMs do not produce better predictions than MMs. It is assumed that such results are caused by the fact that there is no hidden factor which drive the process or the data is too diverse hence a much larger dataset is needed in order to be able to train accurate HMMs.

To simplify and retain accuracy, we opt to build higher order Markov models. Higher order Markov models have better accuracy than lower ones but do come with the issue of lack of coverage. To improve MMs accuracy, higher MMs are used for their strength while their weakness (low coverage) is addressed by adding a default prediction improvement technique. This is a sequence alignment based technique. The experimental evaluation was operated using two datasets which are real business processes from BT. The experimental results of this chapter show that our proposed approach improves the models' performance significantly. Despite these improvements in model performance compared to the ones from the original models and extensions, the model still does not reach the level where it can be relied on for online process maintenance. This is mainly caused by the fact that our available data is too diverse which is not an advantageous environment for probabilistic models where support is important.

Hints obtained from experimental results in Chapter 3 drive our research towards a sequential classification approach. This approach is to predict if a process instance is running into success or failure. Being aware of predictive models' low accuracy on our data and of loops in process executions, we also introduce in this chapter a rule based technique for process failure prediction. This technique examines the link between task repetition (consecutively) and process failure. These methods are implemented using three datasets from the same telecommunications company. Among these datasets, two of them are data drawn from real processes in this company, the other is a customer behaviour data (churn data). We bring in the latter dataset as churn data consists of asynchronous sequences of events so these datasets share some common characteristics. The methods are then tested if it is generic by using data from a different problem. The chosen classifier in this work is KNN. It is adapted to a sequential approach by coupling with sequence alignment. The novel approach shows high accuracy when applied to the churn dataset but did not show any relevance in dealing with the other two business process datasets. The loop failure detection technique also provided its performance differently according to datasets. Hence, there is no such special rule drawn from the used data by the technique that is valid for all datasets. The loop-failure relationships varies from case to case instead.

There followed the work in Chapter 5. It is here that a complex model is built in order to first partition the data naturally and sequentially into different groups. Each group contains similar sequences. Second, predictive models are employed to each cluster independently in order to find suitable models for individual clusters. In consequence, the performance of prediction for the whole dataset increases. The strategy adopted in this chapter aimed at "horizontall" localising business process sequences by dealing with different representative prototypes found by dividing data sequences into groups of similar sequences. It is different from what is commonly done in process mining: localising process data (local learning) by "vertically" dividing business processes into different operational states with data sequences are divided into subsequences. The experiments in this work are twofolds. Data is partitioned into groups and these groups are used as input for predictive models. The performance of these models applied to data in individual groups are used to evaluate the strategy.

6.2 Project Achievement, Contributions and Limitations

In this Section, the main achievements of the project, with regard to the aims and objectives set from the very beginning, are discussed. These discussions are based on evidence gained by the work throughout this thesis, especially over the previous three chapters with novel approaches and their assessments (experiment evaluations).

6.2.1 Proposed an Extension Markov Model

Although the performance of the proposed model is not high enough to be launched in online operational support for processes, it significantly improves MMs' performance and it also outperforms some other extension MMs (Le *et al.* (2012)). The idea of improving the default prediction is appropriate and it is reasonable to use event orders to compare sequences.

6.2.2 A New Sequential KNN

Usually the non sequential classical data mining technique KNN is coupled with sliding window to form a sequential approach. We established a simple KNN using similarity measurement obtained from the matching of any two given sequences. The matching is realised by local or global algorithms from sequence alignment. Our extension KNN takes into account the order of events in one sequence (Le *et al.* (2013b)). This contributes to the KNN family and to sequential approaches in data mining.

Another contribution of this method is that it performs well on the available churn dataset. This is promising in applying to churn data in different fields apart from telecommunications. This fact is illustrated by a number of experimental results presented in Chapter 4.

6.2.3 Warning for Process Failures

As it is not easy to predict next process steps in general, a partial solution is to control some key events. In more detail, the objective here is to find rules which can warn when there is a threat to the completion of the process. For example, a rule that warns about the process termination can be constructed (based on the average time duration of the remaining paths) if there are sequences of consecutive repetition of task A in the input data. Similarly, another rule can be constructed based on the link between the repetition of A and the failure event (process failure).

6.2.4 Flexible Hybrid Predictive Strategy

The performance of our predictive models and classifiers given in Chapters 3 and 4 do not assure online process performance improvement. The solution to this issue is to cluster the data into different groups whose characteristics are expected to be similar within one group and to be different between groups (Le *et al.* (2014)). We successfully built a hybrid model which is flexible in terms of dealing with different patterns in data, in the other words with different prototypes (e.g. different execution of a process). This model is the combination of two steps and multiple predictive models. The first step is to cluster data sequentially. The second step is to select suitable mathematical models for individual clusters. This strategy is proven right by evidence given in Chapter 5.

6.3 Contributions and Limitations

6.3.1 Contributions

Contributions of the work to the project in particular and to process mining and data mining in general are listed in the following:

- A brief review of process management in terms of process predictive analysis.

- Review of a range of mathematical methods in data mining whilst paying much attention to sequential approaches.

  These approaches are candidates for the projects goals and selected to be used in solving the project's problems after initial study. We reviewed three groups of approaches: (1) sequential probabilistic methods including MMs, HMMs, GPs, Conditional random fields; (2) sequential pattern mining including association rules and one of its common algorithms GOSPADE and sequence alignment with local and global algorithms; (3) Sequential classification and clustering. In this category, common approaches like KNNs, K means and its extensions are researched. Additionally, some nonsequential approaches which perform well in dealing with diverse data and that are convertible into sequential approaches are reviewed. These methods are NNs, KNNs, decision trees.

- Extending MMs by tackling the default prediction factor.

- Novel algorithm for loop failure link detection technique (LFD).

  Consecutive task repetition is an issue and listed as a challenge in the process manifesto. One is question posed: how to extract loops and investigate their impacts on processes in general and process performance in particular. Addressing this, we introduced a rule based technique which is first able to detect loops in processes from the data pool and second detecting causal rules about the link between loops and process failure. This contributes to the resolution of one of the listed challenges in process mining. The description of the algorithms are given in Chapter 4.

- Introduced a sequential KNN, called KnsSA, which is suitable for symbolic data sequences.

- Empirical evidence for the effectiveness of horizontally dividing process sequences into different prototypes in predicting next process steps.

- Defined the framework for a hybrid flexible predictive model

  We constructed a hybrid model that combines a number of approaches. The model needs to be flexible as it has to be able to provide predictions, to discover the process logic, and also be able to give warnings about likely process failures. More importantly, the term *flexible* here also means that the hybrid model can effectively adapt and respond to changes in data patterns.

6.3.2 Limitations

The results show that none of the proposed predictive methods can cope with the stated problem alone. The performance is not good enough to assure a good prediction. Thus, the output from the models could not yet be used to predict the next process steps and the remaining path.

6.4 Directions for Future Work

The following list makes some suggestions to expand the work in this thesis.

- Enrich the desired complex model by adding new relevant predictive models to the hybrid one.

  The thesis constructs a hybrid model that combines a number of approaches. Following the works done previously, it is possible to build more predictive models for the clusters, not just MSAs of different orders and KnsSA. Of the methods reviewed in Chapter 2, immune inspired classifiers and artificial neural networks are potential candidates.

- Add a new module to the model whose role is to find and suggest an optimal way to complete a given process instance from the current step. If the graph that presents the process is weighted and directed, given an uncompleted sequence, the optimal remaining path is the optimal way to finish the given subsequence. Specifically, weights are assigned to each edge of the graph based on the frequency of the occurrence of that edge in the training data set. This weight choosing procedure is just an initial option and ways to determine weights for the graph still need to be investigated.

  However, in the graphs which represent processes, there is no ending vertex and being able to reach the assumed ending node does not mean the process is completed. This is caused by the fact that there can be a number of ending nodes, starting nodes and there are edges linking them. The solution is to find a template of key tasks (events) that one must complete in order to finish the process successfully. These tuples of events are then checked regularly on the way of finding the optimal remaining path to assure process completion.

  To find the optimal remaining path of a given subsequence, we can try to find the shortest path in the corresponding graph using the philosophy of combinatorial optimisation. This graph has a source corresponding to the current task and one sink connected to the terminating tasks of the process. From the original graph, all the vertices that occur in the given subsequence will be deleted, except for the current task. The vertex in the original graph corresponding to the current task is the source of the new graph. To create a sink, a new vertex is added and is linked to all the vertices which appear as terminating tasks of the process.

- Extend the research of loop failure links by considering the impacts of cycles on the process outcomes as well.

- Look into application in other industries. Predictive models have applications in various fields. It is therefore a good opportunity to test out our proposed models using data from other fields. Our first attempt was successful and it inspired us to go on trying to apply such models on other fields. The promising MSA model whose accuracy significantly improved comparing to the original MMs, motivated and encouraged us.

Bibliography

Agrawal, R. and Srikant, R. (1994). Fasts algorithms for mining association rules in large databases. In *VLDB '94 Proceedings of the 20th International Conference on Very Large Data Bases*, pp. 487–499.

Agrawal, R. and Srikant, R. (1995). Mining sequential patterns. In *ICDE '95 Proceedings of the Eleventh International Conference on Data Engineering*, pp. 3–14.

Andreopoulos, B., An, A., and Wang, X. (2006). Bi-level clustering of mixed categorical and numerical biomedical data. *International Journal of Data Mining and Bioinformatics Archive*, **1**(1), 19–56.

Anne, R. (2010). *Process Mining Conformance and Extension*. PhD Thesis. Technische Universiteit Eindhoven, Eindhoven, The Netherlands, 2010.

Archaux, C., Laanaya, H., Martin, A., and Khenchaf, A. (2004). An svm based churn detector in prepaid mobile telephony. In *International Conference on Information & Communication Technologies: from Theory to Applications (ICTTA)*, pp. 19–23.

Ascarza, E. and Hardie, B. (2009). Modeling churn and usage behavior in contractual settings.

Babcock, B., Babu, S., Datar, M., Motwani, R., and Widom, J. (2002). Models and issues in data stream systems. In *Proceedings of the Twenty-first ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems*, pp. 1–16. ACM.

Baum, L., Petrie, T., Soules, G., and Weiss, N. (1970). A maximization technique occurring in the statistical analysis of probabilistic functions of markov chains. *The Annals of Mathematical Statistics*, **41**(1), 164–171.

Berlingerio, M., Pinelli, F andNanni, M., and Giannotti, F. (2009). Temporal mining for interactive workflow data analysis. In *KDD '09 Proceedings of the 15th ACM SIGKDD international conference on Knowledge discovery and data mining*, pp. 109–118. ACM.

Berry, M. and Linoff, G. (2004). *Data Mining Techniques: for Marketing, Sales, and Customer Relationship Management*. Wiley, Newyork.

Bottou, L., Bengio, Y., and LeCun, Y. (1997). Global training of document processing systems using graph transformer networks. In *In Proceeding of Computer Vision and Pattern Recognition*, pp. 490–494.

Breiman, L. (1996). Bagging predictors. *Machine Learning*, **24**(2), 123–140.

Chang, Y. and Martinsek, A. (2004). *Applied Sequential Methodologies*. Marcel Dekker, New York.

Chen, T. and Ren, J. (2009). Bagging for gaussian process regression. *Neurocomputing*, **72**(7-9), 1605–1610.

Chiu, M. and Khoo, L. (2005). A new method for analyzing sequential processes : Dynamic multilevel analysis. In *Small group research ISSN 1046-4964*, vol. 36, pp. 600–631. SAGE, California.

Cox Jr, L. and Popken, D. (2002). A hybrid system-identification method for forecasting telecommunications product demands. *International Journal of Forecasting*, **18**(4), 647–671.

Deshpande, M. and Karypis, G. (2004). Selective markov models for predicting web page accesses. *ACM Transactions on Internet Technology (TOIT)*, **4**(2), 163–184.

Dhillon, S. and Modha, S. (2001). Concept decompositions for large sparse text data using clustering. *Machine Learning*, **42**, 143–175.

Dietterich, T. (2002). Machine learning for sequential data: A review. In *Structural, Syntactic, and Statistical Pattern Recognition* (Caelli, T., ed.), pp. 15–30. Springer, New York.

Du, Y., Liang, Y., Jiang, J., Berry, R., and Ozaki, Y. (2004). Spectral regions selection to improve prediction ability of pls models by changeable size moving window partial least squares and searching combination moving window partial least squares. *Analytica Chimica Acta*, **501**(2), 183–191.

Duda, R., Hart, P., and Stork, D. (2001). *Pattern Classification*. Wiley, New York.

Eastwood, M. and Gabrys, B. (2009). A non-sequential representation of sequential data for churn prediction. In *Proceedings of the KES2009 Conference*, pp. 209–218. Santiago, Chile.

Eirinaki, M., Vazirgiannis, M., and Kapogiannis, D. (2005). Web path recommendations based on page ranking and markov models. WIDM '05, pp. 2–9.

Elavarasi, A., Akilandeswari, J., and Sathiyabhama, B. (2011). A survey on partition clustering algorithms. *International Journal of Enterprise Computing and Business Systems*, **1**(1), 478–484.

Elkan, C. (2003). Using the triangle ilequality to accelerate k-means. In *20th International Conference on Machine Learning (ICML-2003), Washington DC, 2003*, pp. 2–9.

Folino, F., Guarascio, M., and Pontieri, L. (2012). Context-aware prediction on business process executions. In *new Frontiers in Mining Complex Patterns (NFMCP2012)*, pp. 152–159.

Forney, G. (1973). The viterbi algorithm. In *proc. IEEE*, vol. 61, p. 268278.

Gabrys, B. and Bargiela, A. (2000). General fuzzy min-max neural network for clustering and classification. *IEEE Transactions on Neural Networks*, **11**(3), 769–783.

Garcia, D., Parrado, E., and Diaz-de Maria, F. (2009). A new distance measure for model-based sequence clustering. *IEEE Transactions on Pattern Analysis and Machine Intelligent*, **1**(7), 1325–1331.

Greco, G., Guzzo, A., Pontieri, L., and Sacc, D. (2006). Discovering expressive process models by clustering log traces. *IEEE Transactions on Knowledge and Data Engineering*, **18**(8), 1010–1027.

Hadden, J., Tiwari, A., Roy, R., and Ruta, D. (2006). Churn prediction: Does technology matter. *International Journal of Intelligent Technology*, **1**(1), 104–110.

Han, J. and Fu, Y. (1995). Discovery of multiple-level association rules from large databases. In *VLDB '95 Proceedings of the 21th International Conference on Very Large Data Bases*, pp. 420–431.

Heinrich, R. and Paech, B. (2013). On the prediction of the mutual impact of business processes and enterprise information systems. In *Lecture Notes in Informatics*, vol. P-239, pp. 157–170. Software Engineering 2013.

Jagadeesh, R. and van der Aalst, W. (2010). Trace alignment for process mining: Opportunities for process diagnostics. In *Lecture Notes in Business Process Management*, vol. 6336, pp. 227–242. Springer, Berlin.

Kanungo, T., Mount, D., Netanyahu, N., Piatko, C., Silverman, R., and Wu, A. (2002). An efficient k-means clustering algorithm: Analysis and implementation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, **24**, 881–892.

Lafferty, J., McCallum, A., and Pereira, F. (2001). Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *ICML '01 Proceedings of the Eighteenth International Conference on Machine Learning*, pp. 282–289.

Lariviere, B. and Van den Poel, D. (2005). Predicting customer retention and profitability by using random forests and regression forests techniques. *Expert Systems with Applications*, **29**(2), 472–484.

Le, M., Gabrys, B., and Nauck, D. (2012). A hybrid model for business process event. In *32nd BCS-SGAI International Conference on Artificial Intelligence*, pp. 179–192.

Le, M., Gabrys, B., Nauck, D., and Martin, T. (2013a). Knns and sequence alignment for churn prediction. In *33rd BCS-SGAI International Conference on Artificial Intelligence*, pp. –.

Le, M., Nauck, D., Gabrys, B., and Martin, T. (2013b). Sequential approaches for predicting business process outcome and process failure warning. In *3rd SIMPDA International Symposium on Data Driven Process Discovery and Analysis, Workshop of the International Conference Very Large Databases (VLDB)*, pp. 1–15.

Le, M., Nauck, D., Gabrys, B., and Martin, T. (2014). Sequential clustering for event sequences and its impact next process step prediction. In *15th IPMU International Conference on Information Processing and Management of Uncertainty in Knowledge-Based Systems*, pp. –.

LeCun, Y., Bottou, L., and Bengio, Y. (1997). Reading checks with multilayer graph transformer networks. In *IEEE International Conference on Acoustics Speech and Signal Processing*, pp. 151–154.

Leleu, M., Ligotti, C., Boulicaut, J., and Euvrard, G. (2003). Go-spade: Mining sequential patterns over datasets with consecutive repetitions. In *Conference on Machine Learning and Data Mining in Pattern Recognition*, pp. 293–306.

Lemmens, A. and Croux, C. (2006). Bagging and boosting classification trees to predict churn. *Journal of Marketing Research*, **43**(2), 276–286.

Li, C. and Biswas, G. (1999). Clustering sequence data using hidden markov model representation. In *Proceedings of the SPIE99 Conference on Data Mining and Knowledge Discovery: Theory*, pp. 14–21.

Li, T., Ma, S., and Ogihara, M. (2004). Entropy-based criterion in categorical clustering. In *21st International Conference on Machine Learning*, pp. 68–76.

MacKay, D. (2003). *Information theory, Inference, and Learning Algorithms*. Cambridge Univ Press, Cambridge.

McKibben, J. and Pacatte, L. (2003). Business process analysis/modelling for defining gis applications and uses. In *ESRI Library, http://gis.esri.com/library/userconf/proc03/p0537.pdf*.

Mika, K., Heikki, M., Pirjo, R., Hannu, T., and Inkeri, V. (1994). Finding interesting rules from large sets of discovered association rules. In *CIKM '94 Proceedings of the Third International Conference on Information and Knowledge Management*, pp. 401–407.

Milligan, M. (2001). A sliding window technique for calculating system lolp contributions of wind power plants. In *AWEA's Windpower 2001 Conference, Washington, DC, June, 2001*.

Mingers, J. (1989). An emperical comparison of prunning methods for decision tree induction. *Machine Learning*, **4**, 227–243.

Motoi, S., Nakada, Y., Misu, T., Yazaki, T., Matsumoto, T., and Yagi, N. (2008). A hierarchical bayesian hidden markov model for multi-dimensional discrete data. In *AIA '08 Proceedings of the 26th IASTED International Conference on Artificial Intelligence and Applications*, pp. 52–57.

Neal, R. (1996). *Bayesian Learning for Neural Networks*. Springer Verlag, New York.

Needleman, S. and Wunsch, C. (1970). A general method applicable to the search for similarities in the amino acid sequence of two proteins. *Journal of Molecular Biology*, **48**, 443–453.

Netzer, O., Lattin, J., and Srinivasan, V. (2007). A hidden markov model of customer relationship dynamics. *Marketing Science*, **27**, 185–204.

Papoulis, A. and Pillai, S. (1991). *Probability, Random Variables and Stochastic Processes*. McGraw-Hill, New York.

Pham, D., Dimov, S., and Nguyen, C. (2005). Selection of k in k-means clustering. *I MECH E Part C Journal of Mechanical Engineering Science*, **219**(1), 103119.

Pimentel, M., Clifton, D., and Tarassenko, L. (2013). Gaussian process clustering for the functional characterisation of vital-sign trajectories. In *2013 IEEE International Workshop on Machine Learning for Signal Processing, September 2013, Southampton, UK*, pp. 1–6.

Pitkow, J. and Pirolli, P. (1999). Mining longest repeating subsequences to predict world wide web surfing. In *The 2nd USENIX Symposium on Internet Technologies & Systems*, pp. 139–150.

Porikli, F. (2004). Clustering variable length sequences by eigenvector decomposition using hmm. In *In Lecture Notes in Computer Science*, pp. 352–360. Springer-Verlag.

Quinlan, J. (1986). Induction of decision trees. *Machine learning*, **1**, 81–106.

Raedt, L andBlockeel, H. (1997). Using logical decision trees for clustering. In *Lecture Notes in Computer Science*, vol. 1297, p. 133140. Springer, Berlin.

Rajaraman, A. and Ullman, J. (2011). *Mining of Massive Datasets*. Cambridge University

Press, Cambridge.

Rasmussen, C. and Williams, C. (2006). *Gaussian Processes in Machine Learning*. MIT Press, Cambridge, MA, USA.

Rokach, L. and Maimon, O. (2008). *Data Mining with Decision Trees: Theory and Applications*. World Scientific Publishing, Singapore.

Ruta, D. and Majeed, B. (2011). Business process forecasting in telecommunication industry. In *GCC Conference and Exhibition (GCC), 2011 IEEE*, pp. 389–392.

Ruta, D., Nauck, D., and Azvine, B. (2006). K nearest sequence method and its application to churn prediction. In *Intelligent Data Engineering and Automated Learning IDEAL 2006* (Corchado, E., Yin, H., Botti, V., and Fyfe, C., eds.), vol. 4224 of *Lecture Notes in Computer Science*, pp. 207–215. Springer, Berlin.

Safavian, R. and Landgrebe, D. (1991). A survey of decision tree classifier methodology. *IEEE Transactions of System, Man, and Cybernetics*, **21**(3), 660674.

Smith, T. and Waterman, M. (1981). Identification of common molecular subsequences. *Journal of Molecular Biology*, **147**, 195–197.

Smyth, P. (1997). Clustering sequences with hidden markov models. In *Advances in Neural Information Processing Systems*, pp. 648–654. MIT Press.

Snelson, E. and Ghahramani, Z. (2006). Sparse gaussian processes using pseudo-inputs. *Advances in Neural Information Processing Systems, Proceeding Track*, **16**, 524–531.

Song, M. and van der Aalst, W. (2007). Supporting procesmining by showing events at a glance. In *Advances in Neural Information Processing Systems*, pp. 139–145.

Srikant, R. and Agrawal, R. (1996). Mining sequential patterns: Generalizations and performance improvements. In *5th International Conference on Extending Database Technology: Advances in Database Technology*, pp. 3–17.

Steinbach, M., Karypis, G., and Kumar, V. (2000). A comparison of document clustering techniques. In *KDD Workshop in Text Mining*.

Taylor, P., Leida, M., and Majeed, B. (2012). Case study in process mining in a multinational enterprise. In *Lecture Notes in Business Information Processing*. Springer, Berlin.

Thede, S. and Happer, M. (1999). A second-order hidden markov model for part-of-speech tagging. In *The 37th Annual Meeting of the ACL*, pp. 175–182.

Trcka, N. and Pechenizkiy, M. (2009). From local patterns to global models: Towards domain driven educational process mining. In *9th International Conference on Intelligent Systems Design and Applications, ISDA 2009, Pisa, Italy , November 30-December 2, 2009*, pp. 1114–1119.

Tsui, K., Chen, V., Jiang, W., and Aslandogan, Y. (2005). Data mining methods and applications. In *Handbook of Engineering Statistics* (Pham, H., ed.), pp. 651–669. Springer-Verlag London.

van der Aalst, W. (2011). *Process mining: Discovery, Conformance and Enhancement of Business Processes*. Springer-Verlag, New York Icn.

van der Aaslt, W. (2004). Business process management: A personal view. *Business Process Management Journal*, **10**(2), 135–139.

van der Aaslt, W. and Weijters, A. (2004). Process mining: Research agenda. *Computers* *in Industry*, **53**(3), 231–244.

van der Aaslt, W., Schonenberg, M., and Song, M. (2011). Time prediction based on process mining. *Information Systems*, **2**, 450–475.

van der Aaslt et al, W. (2011). Process mining manifesto. In *Business Process Management Workshops 2011* (Daniel, F., Barkaoui, K., and Dustdar, S., eds.), vol. 99 of *Lecture Note in Business Information Processing*. Springer, Berlin.

Wagstaff, K., Cardie, C., Rogers, S., and Schrodl, S. (2001). Constrained k-means clustering with background knowledge. In *18th International Conference on Machine Learning*, pp. 577–584.

Wallach, H. M. (2004). Condition random fields: An introduction. Tech. rep.

Waterman, M. (1994). Estimating statistical significance of sequence alignments. *Philosophical Transactions of the Royal Society of London, Series B: Biological Sciences*, **344**, 383–390.

Weijters, A. and van der Aalst, W. (2001). Process mining discovering workflow models from event-based data. In *ECAI Workshop on Knowledge Discovery and Spatial Data*, pp. 283–290.

Widmer, G. and Kubat, M. (1996). Learning in the presence of concept drift and hidden contexts. *Machine Learning*, **23**(1), 69–101.

Zaki, M., Peters, M., Assent, I., and Seidl, T. (2007). Clicks: An effective algorithm for mining subspace clusters in categorical datasets. *Data Knowl. Eng.*, **60**(1), 51–70.