UNIVERSITY OF TARTU
Institute of Computer Science
Software Engineering Curriculum

Kaur Järvpõld

Leveraging Multi-Perspective A-priori Knowledge in Predictive Business Process Monitoring

Master's Thesis (30 ECTS)

Supervisors: Fabrizio Maria Maggi
Chiara Di Francescomarino
Chiara Ghidini

Tartu 2018

Leveraging Multi-Perspective A-priori Knowledge in Predictive Business Process Monitoring

Abstract:

Predictive business process monitoring is an area dedicated to exploiting past process execution data in order to predict the future unfolding of a currently executed business process instance. Most of the research done in this domain focuses on exploiting the past process execution data only, leaving neglected additional a-priori knowledge that might become available at runtime. Recently, an approach was proposed, which allows to leverage a-priori knowledge on the control-flow in the form of LTL-rules. However, cases exist in which more granular a-priori knowledge becomes available about perspectives that go beyond the pure control-flow like data, time and resources (multi-perspective a-priori knowledge). In this thesis, we propose a technique that enables to leverage multi-perspective a-priori knowledge when making predictions of complex sequences, i.e., sequences of events with a subset of the data attributes attached to them. The results, obtained by applying the proposed technique to 20 synthetic logs and 1 real-life log, show that the proposed technique is able to overcome state-of-the-art approaches by successfully leveraging multi-perspective a-priori knowledge.

Keywords:

Predictive Process Monitoring, Deep Learning, Recurrent Neural Networks, A-priori Knowledge, MP-Declare

CERCS: P170 – Computer Science, Numerical Analysis, Systems, Control

Pealkiri eesti keeles

Lühikokkuvõte:

Äriprotsesside ennestusseire on valdkond, mis on pühendunud käimasolevate äriprotsesside tuleviku ennustamiseks kasutades selleks minevikus sooritatud äriprotsesside kohta käivaid andmeid. Valdav osa uurimustööst selles valdkonnas keskendub ainult seda tüüpi andmetele, jättes tähelepanuta täiendavad teadmised (a-priori teadmised) protsessi teostumise kohta tulevikus. Hiljuti pakuti välja lähenemine, mis võimaldab a-priori teadmisi kasutada LTL-reeglite näol. Kuid tõsiasjana on antud tehnika limiteeritud äriprotsessi kontrollvoole, jättes välja võimaluse väljendada a-priori teadmisi, mis puudutavad lisaks kontrollvoole ka informatsiooni protsessis leiduvate atribuutide kohta (multiperspektiivsed a-priori teadmised). Me pakume välja lahenduse, mis võimaldab seda tüüpi teadmiste kasutuse, tehes multiperspektiivseid ennustusi käimasoleva äriprotsessi kohta. Tulemused, milleni jõuti rakendades väljapakutud tehnikat 20-le tehis-ärilogile ning ühele elulisele ärilogile, näitavad et meie lähenemine suudab pakkuda konkurentsivõimelisi ennustusi.

Võtmesõnad:

Äriprotsesside ennestusseire, Sügavõpe, Rekurrentsed närvivõrgud, A-priori teadmised, MP-Declare

CERCS: P170 – Arvutiteadus, Arvutusmeetodid, Süsteemid, Juhtimine

Table of Contents

1 Introduction

It is inevitable to notice that in the modern business world it is not enough to merely have a good product or service. The survival and success of an organization is highly dependent on the level of quality in which the product or service is being delivered. This means delivering it in a shorter amount of time, with lower costs and higher customer satisfaction. Organizations are constantly looking for ways to have a clearer overview of the strengths and weaknesses of their operations in order to improve the managerial side of their business. In this light, the field of Process Mining has evolved. Process mining is a family of techniques that exploits available information about past executions of a business process to analyse and improve it.

Exploiting knowledge about the past is possible thanks to the rapid evolution of advanced information systems which store detailed information about business activities in event logs. An event log is a dataset consisting of unique process execution instances called traces. A trace in turn consists of events which are executed and stored in the trace. Additional information about events, traces or the entire log, is held in data attributes. The existence of this valuable data has led to the possibility to predict the future behaviour of an organization. The idea of seeing the future has attracted tremendous attention in the process mining community and has led to a sub-domain of process mining called *Predictive Business Process Monitoring* (PBPM). More specifically, it aims to foresee how an ongoing process instance unfolds. Typical prediction scenarios include predicting the remaining time of an ongoing trace, the fulfilment of a *Service Level Agreement*, the next event or the full sequence of events until the end of the trace. The latter is the type of prediction we focus on in this thesis. The idea behind PBPM is simple – a prediction model is built based on the available past information, the model is then queried with an unfinished trace and the corresponding prediction is made. The latest trend in PBPM is the usage of deep learning which has become increasingly popular thanks to the availability of powerful computational hardware and innovations in algorithmics. The application of neural networks in PBPM has a lot of common grounds with *Natural Language Processing* (NLP). We can think of an event log as a block of text, the individual process instances as sentences and the events occurring as words. After drawing these parallels it is easy find motivation from NLP when applying deep learning to PBPM.

However, what motivates this work, is the fact that most of the research in PBPM distinctly focuses on using information abstracted from past event logs only, leaving neglected additional knowledge that might be available about the future unfolding of the process. Recently, an approach was proposed in [2], which leverages a-priori knowledge on top of the information extracted from the logs to enhance the prediction accuracy. The paper proves that a-priori knowledge can efficiently be used to increase the accuracy of predicting the suffixes of ongoing traces. Although this approach has proven to be effective, it has not reached its full potential. The current solution quickly hits its limits when multi-perspective a-priori knowledge is concerned. In particular, the proposed technique allows for specifying knowledge exclusively on the control-flow, without the possibility of specifying more granular knowledge including information about data attributes attached to the events (a.k.a. the payload). In many cases a-priori knowledge about the payload could be a valuable asset when the future of a running case is predicted. Let us consider an example about a treatment process in a hospital where treatment A can only be performed by Dr. Smith. If a patient comes to the hospital with the need for treatment A, but Dr. Smith is out of office due to illness, then treatment A can not be performed. Instead an alternative treatment B is performed which by Dr. Brown. This example illustrates how multi-perspective a-priori knowledge can give us insights about the future unfolding about the treatment process of

this particular patient. Therefore an approach for leveraging multi-perspective a-priori knowledge is necessary when making multi-perspective predictions. In the light of this motivation, and as a direct extension to the work done in [2], we propose an approach which enables to leverage a-priori knowledge not only on the control-flow, but also on the accompanying data payload.

2 Background

2.1 Process Mining

Process mining [47] can help organizations understand how their processes actually work in reality, measure the performance of the processes and find out to which extent the processes are following a predefined set of business rules. All of these questions are answered by analysing event logs which contain data about the activities carried out in an organization. Event logs are normally created by business process management systems and stored in XES[1] format. They contain data about individual process execution instances and the accompanying data payload. The data payload usually includes, but is not limited to, the timestamp of each event and the resource used to carry out the corresponding activity. The resource attribute is conjointly the data attribute we focus on in this thesis.

The tasks performed in the process mining domain are categorized into three main families. Namely *process discovery*, *conformance analysis* and *process enhancement.* Process discovery aims at learning structural patterns from an event log. The outcome is a process model giving insights about the real life operational patters of an organization. Conformance analysis takes an already existing process model and uses it to measure a process' performance, i.e. it checks if a process is following a predefined set of business rules. The results of the conformance analysis are then used during process enhancement to alter the existing process model with the aim of making it more efficient.

Process mining itself can be divided into two mechanisms – process controlling and process monitoring. Process controlling is an aggregated approach used to get insights about how a process has been performing during a certain period of time and is typically an offline analysis, i.e. the logs of already completed process executions are used. Process monitoring is an online analysis and takes as input currently running process executions. The aim is to monitor whether processes are executed conformingly and in the case of violations trigger counter measures.

2.2 Predictive Business Process Monitoring

A subfield of process mining called Predictive Business Process Monitoring (PBPM) is an area dedicated to predicting the future unfolding of currently running traces. As a descendant of process mining, PBPM also exploits event logs to gain information. Although in this case, the information is used to build predictive models which are purposed to predict an outcome of a case as early as possible. The predictive models are able to perform by taking as input the part of an ongoing trace executed so far. Some of the common prediction tasks include the remaining cycle time prediction, next activity prediction, the fulfilment of a *Service Level Agreement*.

Many mathematical models have been used to complete the prediction tasks, out of which, some of the most common methods are *regression* [32][41][49][50], *clustering* [49][41][30], *decision trees* [48][24][51][11] and *hidden Markov models* [24][51][37]. In this thesis we are following the latest trend in PBPM, *deep learning*, and use neural networks as a prediction model. The state-of-the-art work done in this domain can be found in [21][1][2].

[1] XEX (eXtensible Event Stream) is an xml-based standard for capturing systems behaviors (http://www.xes-standard.org/)

2.3 Event Log

An event log is a representation of the execution history of a certain business process. It consists of a set of traces, which represent specific executions of the business process. A trace, in turn, is a set of temporally ordered events which specify the control-flow of the process execution. Traces and events represent the structure of a log, but do not hold any informational knowledge themselves. Accompanying information of a log, trace or activity is held in its attributes. An attribute is a key-value pair, which can contain temporal, textual or numerical information. Attributes can be divided into two families – *static attributes*, which remain the same throughout the trace and dynamic attributes, which can change from event to event, e.g. a dynamic attribute can be the timestamp of an event (event level attribute) and a static attribute can be the gender of a medical patient (trace level attribute).

Definition 1. A trace $\sigma = \langle e_1, e_2, e_3, \dots, e_n \rangle \in E^*$ is a sequence of events where E^* is a multi set of events over a finite set of all possible events E.

A prefix $p_k(\sigma) = \langle e_1, e_2, e_3, \dots, e_k \rangle \in E^*$ is the subsequence of the first k events of a trace $\sigma = \langle e_1, e_2, e_3, \dots, e_n \rangle \in E^*$ where $k \leq n$. The suffix of the prefix $p_k(\sigma)$ is the remaining part of the trace σ defined as $s_k(\sigma) = \langle e_{k+1}, e_{k+2}, e_{k+3}, \dots, e_n \rangle \in E^*$. E.g the prefix $p_3(\sigma)$ of the trace $\sigma = \langle a, b, c, d, b, b, a, c \rangle$ is $\langle a, b, c \rangle$ and the suffix $s_3(\sigma)$ is $\langle d, b, b, a, c \rangle$.

2.4 Linear Temporal Logic

Linear Temporal Logic (LTL)[16], first described by A. Pnueli in 1977, is a modal logic which enables to express temporal knowledge about a sequential structure. A fraction of LTL dedicated to describing finite sequences is used in this thesis. LTL specifications are composed by defining rules on sequential units using boolean operators together with LTL operators dedicated to express temporal aspects. To understand the semantics of *MP-Declare* templates (see 2.5) the following knowledge about the LTL operators is necessary. The O and Y operators represent temporal knowledge about the past, namely $O\varphi_1$ indicates that φ_1 has to hold true at some point in the past, and $Y\varphi_1$ indicates that φ_1 has to hold true in the previous position of the sequence. The F, X, G and U operators represent temporal knowledge about the future, namely $F\varphi_1$ indicates that φ_1 has to hold true at some point in time in the future, $X\varphi_1$ indicates that φ_1 has to hold true in the next position of the sequence, $G\varphi_1$ indicates that φ_1 has to always hold true in the future, $\varphi_1 U \varphi_2$ indicates that φ_2 has to hold true at some point in time in the future, and until that point φ_1 has to hold true.

2.5 Multi-Perspective Declare

Declare [52] is a declarative process modelling language which aims at offering balance between flexibility and support when defining a business process. It is based on LTL semantics and enables to specify constraints on business behaviour creating an *open-world* assumption, i.e. there is no specification of allowed behaviour but rather a specification of illegal behaviour.

Multi-Perspective Declare (MP-Declare) is an extension of *Declare* which allows to describe process behaviour also from the perspective of time and data. Similarly to Declare, it works by specifying constraints over traces, including constraints about additional information accompanied with the control-flow, i.e. the payload. An MP-Declare model consists of a set of constraints which are based on MP-Declare templates.

Table 1. Semantics of MP-Declare Templates (source: [13])

Template	MFOTL Semantics
Existence	$F_I(A \wedge \exists x.\, \varphi_a(x))$
Absence	$\neg F_I(A \wedge \exists x.\, \varphi_a(x))$
Choice	$F_I(A \wedge \exists x.\, \varphi_a(x)) \vee F_I(B \wedge \exists x.\, \varphi_a(x))$
Exclusive choice	$(F_I(A \wedge \exists x.\, \varphi_a(x)) \vee F_I(B \wedge \exists x.\, \varphi_a(x))) \wedge \neg(F_I(A \wedge \exists x.\, \varphi_a(x)) \wedge F_I(B \wedge \exists x.\, \varphi_a(x)))$
Responded existence	$G(\forall x.\, ((A \wedge \varphi_a(x)) \rightarrow (O_I(B \wedge \exists y.\, \varphi_c(x,y)) \vee F_I(B \wedge \exists y.\, \varphi_c(x,y))))))$
Not responded existence	$G(\forall x.\, ((A \wedge \varphi_a(x)) \rightarrow \neg(O_I(B \wedge \exists y.\, \varphi_c(x,y)) \vee F_I(B \wedge \exists y.\, \varphi_c(x,y)))))$
Response	$G(\forall x.\, ((A \wedge \varphi_a(x)) \rightarrow F_I(B \wedge \exists y.\, \varphi_c(x,y))))))$
Not response	$G(\forall x.\, ((A \wedge \varphi_a(x)) \rightarrow \neg F_I(B \wedge \exists y.\, \varphi_c(x,y))))$
Alternate response	$G(\forall x.\, ((A \wedge \varphi_a(x)) \rightarrow X_I(\neg(A \wedge \varphi_a(x)) U_I(B \wedge \exists y.\, \varphi_c(x,y)))))$
Chain response	$G(\forall x.\, ((A \wedge \varphi_a(x)) \rightarrow X_I(B \wedge \exists y.\, \varphi_c(x,y))))$
Not chain response	$G(\forall x.\, ((A \wedge \varphi_a(x)) \rightarrow \neg X_I(B \wedge \exists y.\, \varphi_c(x,y))))$
Precedence	$G(\forall x.\, ((B \wedge \varphi_a(x)) \rightarrow O_I(A \wedge \exists y.\, \varphi_c(x,y))))$
Not precedence	$G(\forall x.\, ((B \wedge \varphi_a(x)) \rightarrow \neg O_I(A \wedge \exists y.\, \varphi_c(x,y))))$
Alternate precedence	$G(\forall x.\, ((B \wedge \varphi_a(x)) \rightarrow Y_I(\neg(B \wedge \varphi_a(x)) S_I(A \wedge \exists y.\, \varphi_c(x,y)))))$
Chain precedence	$G(\forall x.\, ((B \wedge \varphi_a(x)) \rightarrow Y_I(A \wedge \exists y.\, \varphi_c(x,y))))$
Not chain precedence	$G(\forall x.\, ((B \wedge \varphi_a(x)) \rightarrow \neg Y_I(A \wedge \exists y.\, \varphi_c(x,y))))$

The templates are behavioural patterns which characterize the constraints. They consist of five parameters, namely the activation, the target, the activation condition, the correlation condition and the time condition. An activation of a constraint is an event whose occurrence causes requirements on another event (the target) in the same trace to be effective. An activation condition is a constraint on the payload of the activation event, which has to hold true in order for the constraint to be activated. Note that a constraint is not activated if the activation condition does not hold, even though the activation event has occurred. The correlation condition has to hold when the target event occurs in order for the constraint to be fulfilled. Finally, the time condition can be expressed in the form of an interval $I = [\tau_0, \tau_1)$ where τ_0 and τ_1 represent the minimum and the maximum temporal distances between the activation and the target events accordingly.

Following is the explanation of the MP-Declare templates presented in Table 1. Template *existence* demands that the activation event A has to happen at least once in the trace while the activation condition φ_a is fulfilled. Template *absence* demands the activation event A must not to take place in the trace if the activation condition φ_a is fulfilled. The *choice* template demands that activation event A or the activation event B has to happen at least

once in the trace while the activation condition φ_a is fulfilled. The occurrence of both activities specified in the choice template is allowed. *Exclusive choice* is identical to the choice template with the exception that only one of the specified activities is allowed to take place. Template *responded existence* demands that if the activation event A takes place while the activation condition φ_a is fulfilled then the target event B has to happen either before or after event A while the correlation condition φ_c is fulfilled. Template *response* demands that if activation event A takes place while the activation condition φ_a is fulfilled, then the target event B has to eventually happen after event A while the correlation condition φ_c is fulfilled. Template *alternate response* demands that if activation event A takes place while the activation condition φ_a is fulfilled then the target event B has to eventually happen after event A while the correlation condition φ_c is fulfilled. Activity A with the activation condition φ_a holding true is not allowed to take place again until activity B with correlation condition φ_c fulfilled has taken place. Template *chain response* demands that if activation event A takes place while the activation condition φ_a is fulfilled then the target event B has to happen straight after event A with the correlation condition φ_c fulfilled. Template *precedence* demands that the activation event B with the activation condition φ_a fulfilled can only take place if event A with the correlation condition φ_c fulfilled has taken place at some point in time before B. Template *alternate precedence* demands that the activation event B with the activation condition φ_a fulfilled can only take place if event A with the correlation condition φ_c fulfilled has taken place before B. Another occurrence of A in the time interval $[\tau_A, \tau_B)$ is not allowed. Template *chain precedence* demands that the activation event B with the activation condition φ_a fulfilled can only take place if event A with the correlation condition φ_c fulfilled has taken place straight before B. Template *not responded existence* demands that if activation event A takes place with the activation condition φ_a fulfilled then the target event B can not take place either before or after event A with the correlation condition φ_c fulfilled. Template *not response* demands that if activation event A takes place with the activation condition φ_a fulfilled then the target event B can not take place after event A with the correlation condition φ_c fulfilled. Template *not precedence* demands that the activation event B with the activation condition φ_a fulfilled can only take place if event A with the correlation condition φ_c fulfilled has not taken place before B. Template *not chain response* demands that if activation event A takes place with the activation condition φ_a fulfilled then the target event B can not take place straight after event A with the correlation condition φ_c fulfilled. Template *not chain precedence* demands that the activation event B with the activation condition φ_a fulfilled can only take place if event A with the correlation condition φ_c fulfilled has not taken place straight before B.

2.6 Artificial Neural Networks

An *Artificial Neural Network* (ANN) is a widely used deep learning model designed to mimic the human brain. It consists of building blocks called *neurons*, which act together to conduct complex classification tasks. The neurons operate by receiving standardized input values $\{x_1, x_2, .., x_n\}$ and calculating an output value y based on an activation function. The input values are always adjusted by weights $\{\omega_1, \omega_2, .., \omega_n\}$, which are parameters that the ANN learns during the training phase and which give the neural network the ability to learn from past experience. Inside the ANN all the neurons are connected to each other forming multiple units called the hidden layers. These units conjointly include two special layers called the input layer and the output layer. The input layer receives a feature vector X as an initial input, while the following hidden layers receive the input from the output values of their previous layers. This behaviour carries on until the output layer, which outputs the

probability of the initial feature vector X belonging to a class Y. Mathematically this can be expressed as follows:

$$p(Y|X) = f_{NN}(X;\theta) \quad (1)$$

Where f_{NN} is the aggregated function of all the neurons of the ANN. X is the feature vector given as input to the first hidden layer, Y is the output class and θ the parameters learned by the ANN. Although artificial neural networks have been proven to be efficient in many difficult classification tasks, the approach quickly hits its limits when making sequential predictions. Simple ANNs are only able to establish a long-term memory by adjusting the parameters θ mentioned earlier. Using only this long-term memory means that given the same input, the ANN will always give the same output no matter what point in time the prediction is made. The method fails at obtaining information from previous inputs, meaning that when making a prediction at time step t, the neural network is not able to leverage the information that became available at time step $t-1$. This problem is tackled by a subclass of ANNs called Recurrent Neural Networks (RNN).

2.7 Recurrent Neural Networks

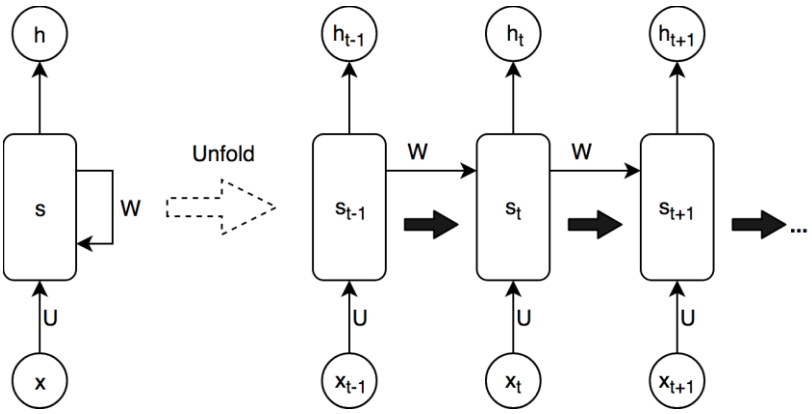

Figure 1 Recurrent Neural Network

RNNs differ from simple ANNs in the sense that their neurons are in addition to other neurons, also recursively connected to themselves, forming a directed loop. This kind of architecture enables the network to establish short-term memory by taking into consideration the new input x_t at time step t as well as the previous information received from past states $\{s_{t-n},..,s_{t-2},s_{t-1}\}$ at time steps $\{t-n,..,t-2,t-1\}$. RNNs have been successfully applied in sequential classification tasks where this kind of short-term memory is valuable, such as machine translation, image description, speech processing and recently also in predictive business process monitoring [1]. For illustration let us take a look at the following example involving machine translation: the sentence “the girl is glad” translates to Russian as “devushka rada” while the sentence “the boy is glad” translates to Russian as “mal'chik rad”. In the example given, the word “glad” translates to Russian differently depending on its preceding inputs. The word “boy” preceding “glad” indicates that the actor in the sentence is male and therefore the male version of the word “glad” should also be used and the same applies for the female version. The example shows that in some classification tasks it is important to know the context in which the classification task is executed. In mathematical terms RNNs can be expressed as follows:

$$p(Y|X_t,..,X_{t-k}) = f_{RNN}(X_t,..,X_{t-k};\theta) \quad (2)$$

Where $X^t, .., X^{t-k}$ is the input sequence of length k and f_{RNN} the aggregated function over all the neurons of the RNN. In more detail let us take a look at how a single unit inside an RNN computes its output values at time step t.

$$h_t = \sigma(WX_t + Uh_{t-1} + b) \quad (3)$$

In (3) h_t is the output value at time step t, σ a sigmoid function (or other non-linear function) applied to each element of the unit separately, X_t the input at timestep t, h_{t-1} the output value of the same unit at timestep $t-1$, W, U and b the weight matrices and bias learned by the RNN respectively.

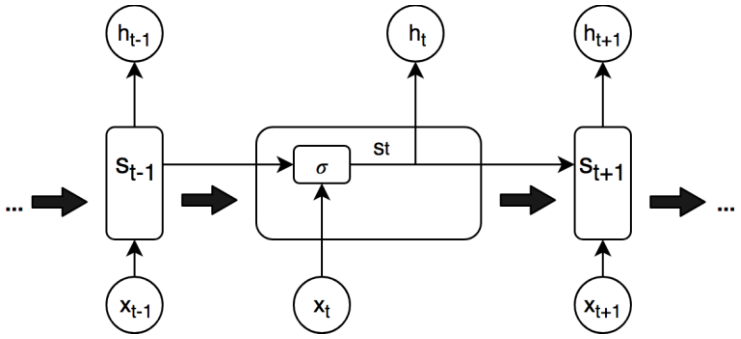

Figure 2 Standard RNN Unit

Unfortunately, the usage of Recurrent Neural Networks in turn quickly introduces a learning problem because of its recursive course of action. Namely, it gets more difficult to adjust the weights of the RNN, the more time steps it passes. The *vanishing gradient* or the *exploding gradient* problem causes the weights of the RNN to either vanish or explode the more it gets recursively multiplied. This means that regular RNNs are not capable of learning long-term dependencies which is a major drawback. These problems are discussed in detail in [4].

2.8 Long Short Term Memory Cells

Long Short Term Memory model (LSTM) introduced in [3] alleviates this learning problem. LSTMs are a special type of RNNs which are able to establish long-term memory using a complex cell system illustrated in Figure 3. Instead of recurrent *neurons*, an LSTM uses recurrent *modules* containing a memory cell c_t which accumulates information from previous inputs. As seen in Figure 3, an LSTM unit consists of four gates f_t, i_t, o_t and $\check{c}_t$ which work as a controlling mechanism to remove or add information to the memory cell c_t. Looking at a layer of LSTM units we can group all the gates of same kind together and name them as follows:

- forget gate layer: f_t
- input gate layer: i_t
- output candidate layer: o_t
- cell state candidate layer: $\check{c}_t$

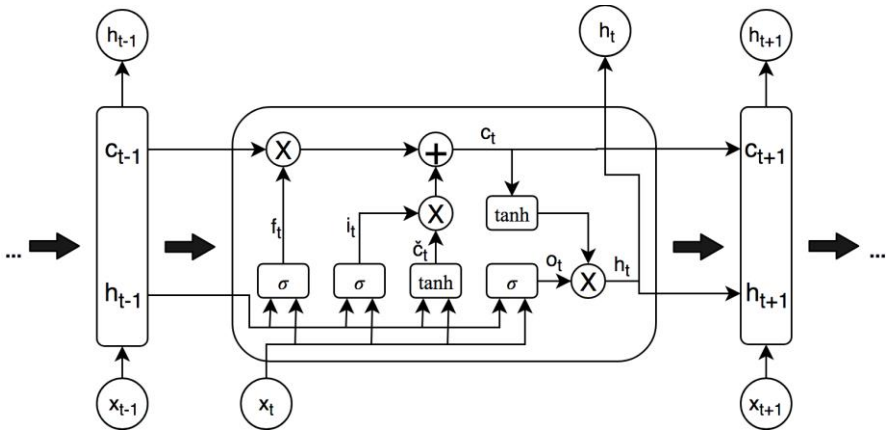

Figure 3 LSTM Unit (source [5])

The first layer in the LSTM is the forget gate layer which uses a sigmoid function to calculate a value between 0 and 1 where 0 means that all the previous knowledge from c_{t-1} should be forgotten and 1 that all the information from c_{t-1} should be remembered. Mathematically this can be expressed as follows:

$$f_t = \sigma(W_f \cdot [h_{t-1}, x_t] + b_f) \quad (4)$$

Where h_{t-1} is the output vector of the LSTM layer at time step $t-1$, x_t is the new input vector and W_f and b_f the weights and biases learned by the neural network respectively. The next step is to decide what new information should be added to the memory cell. This step consists of two parts where first, the input gate layer uses a sigmoid function to decide which are the values that should be updated. Then the cell state candidate layer uses a hyperbolic tangent function to create a vector of the new candidate values $\check{C}_t$ which are possibly added to the memory cell. Mathematically this can be expressed as follows:

$$i_t = \sigma(W_i \cdot [h_{t-1}, x_t] + b_i) \quad (5)$$

$$\check{C}_t = tanh(W_C \cdot [h_{t-1}, x_t] + b_C) \quad (6)$$

The third step is to update the memory cell according to the decisions made by f_t, i_t and $\check{C}_t$.

$$C_t = f_t * C_{t-1} + i_t * \check{C}_t \quad (7)$$

In (7) the old state of the memory cell C_{t-1} is multiplied by the forget gate layer f_t to get rid of any information that is unwanted. Then the new information stored in $\check{C}_t$ is added by first multiplying it with the input gate layer i_t. Finally, the new output h_t is calculated based on the output candidate layer o_t and the updated memory cell C_t.

$$o_t = \sigma(W_o[h_{t-1}, x_t] + b_o) \quad (8)$$

$$h_t = o_t * \tanh(C_t) \quad (9)$$

The described LSTM version is considered as the classical version and exists as only one of many. The state-of-the-art in LSTMs include many variations specifically designed for different tasks. In [6] the authors introduce a very popular version of LSTM where all the gates have *peephole connections* which allow the gates to inspect the current memory cell. This approach allows the LSTM to extract information from the size of the time lags. In other words it is able to count and leverage rhythmical patterns that occur in the time lags. Another increasingly popular version is introduced in [7]. The authors merge the input and update gates together forming a novel *update gate*. The approach is called the Gated Recurrent Unit (GRU) and the idea behind it is to have the input and output gates work closely together so

that the LSTM only updates something that is forgotten and only forgets something that is updated. The authors in [8] introduce Depth-Gated LSTMs which add a linear dependency to the memory cells of the LSTM at time steps $\{t-1, t, t+1\}$. A radically different approach to solve the classical RNN learning problem is introduced in [9]. In [9] the authors describe *ClockWork Recurrent Neural Networks* which modify the RNN so that subsets of the neurons compute at different speeds. A detailed comparison of different RNN architectures is provided in [10].

2.9 LSTMs in Predictive Business Process Monitoring

The state-of-the-art approaches [1][21] for predictive business process monitoring with LSTMs use one-hot encoding to convert the prefixes of traces into feature vector matrices which are used to train and later query the LSTM model. Each event in the prefix $p_k(\sigma)$ is converted into a vector $g_k = (g_{k1}, g_{k2}, .., g_{kj})$ where $j = |A|$ and A is the set of all possible activities (a.k.a. the activity alphabet). Every value in the vector gets set to 0 except for the value representing the event which occurred at that specific step of the suffix which is set to 1.

Given the set of all possible activities $A = \{a_1, a_2, .., a_n\}$ the ordering function $idx : A \rightarrow \{1, .., |A|\} \subseteq N$ determines the indexes of the activities which are used to navigate in the feature vector matrix. E.g. if the set of all possible activities in a trace is $A = \{a, b, c\}$ then $idx : A \rightarrow \{1,2,3\}$ and $idx(a) = 1$, $idx(b) = 2$, $idx(c) = 3$. For illustration let us examine the following example of one hot encoding of a prefix $p_k(\sigma)$, which is illustrated in Table 2. If $A = \{a, b, c, d, e\}$ and the prefix $p_3(\sigma) = \langle c, d, a \rangle$, then the one-hot encoding of this prefix would be a matrix of size $k * |A|$ where k represents the length of the prefix.

Table 2 Examples of One-Hot Encoding

	a	b	c	d	e
ε_1	0	0	1	0	0
ε_2	0	0	0	1	0
ε_3	1	0	0	0	0

During the training phase of the LSTM, the encoded traces are used to build the model. After training the LSTM, an inference algorithm is used to query the model and predict the suffix. Algorithm 1 describes the inference algorithm. The function takes as parameters the prefix $p_k(\sigma)$, the trained LSTM model $lstm$ and a variable max which represents the maximum number of iterations allowed, i.e. the maximum number of activities predicted . The max parameter is usually set to $maxLen + len(p_k(\sigma))$ where $maxLen$ represents the length of the longest trace in the log. Firstly the algorithm initializes a counter $i = 0$ which counts the iterations the algorithm has made or in other words how many activities it has already predicted (line 2). In line 3 the trace that will eventually be returned is initialized as $p_k(\sigma)$. In line 4 the algorithm starts a loop which predicts a new activity at every iteration until the end event is predicted or the maximum number of iterations is reached. Inside the loop, $p_k(\sigma)$ is one-hot encoded as described previously (line 5). The encoded trace is then used to query the LSTM model which outputs a vector of probabilities over the set of all possible activities A (line 6). In line 7 the most probable next symbol is chosen and then

concatenated to $trace$ (line 8). If the end symbol is predicted or the maximum number of allowed iterations is reached then the inference algorithm returns the full trace (line 11).

Algorithm 1: Inference algorithm for predicting the suffix of $p_k(\sigma)$

```
1:  function PredictSuffix(p_k(σ), lstm, max)
2:      i = 0
3:      trace = p_k(σ)
4:      do
5:          trace_encoded = encode(trace)
6:          next_symbol_probs = PredictNextSymbols(lstm, trace_encoded)
7:          next_symbol = GetSymbol(next_symbol_prob, trace_encoded)
8:          trace = trace · next_symbol
9:          i = i + 1
10:     while (not next_symbol == end_symbol) and (i < max)
11:     return trace
12: end function
```

Figure 4 Pseudo-code for Standard Inference Algorithm (source [2])

The results are evaluated by comparing the predicted suffixes to the actual suffixes. The most used evaluation method in this scope is the Damerau-Levenshtein similarity metric described in detail in [12]. Damerau-Levenshtein distance is a metric which is computed based on the minimum number of changes necessary to transform one trace into the other. There are four kinds of operations allowed which include deleting, adding or substituting an event and switching the positions of two consecutive events.

2.10 Complex Symbolic Encoding

Although the previously described encoding method works well in cases where the payload of the activities is arbitrary, it quickly hits its limits when considering the payload of activities is necessary. The authors of [14] propose two complex sequence encodings to alleviate this problem. The first encoding is index-based and is directly expanding the simple sequence encoding discussed earlier by mapping additional features to the feature vector. This approach divides the data associated with traces into static and dynamic data. Static data (a.k.a. trace attributes) is the same for every event in a trace. Dynamic data (a.k.a. event attributes) are dynamic and change from event to event. The resulting vector for a trace σ_k is the following:

$$g_k = (s_k^1, .., s_k^u, event_{k1}, event_{k2}, .., event_{km}, h_{k1}^1, h_{k2}^1, .. h_{km}^1, .., h_{k1}^r, h_{k2}^r, .., h_{km}^r) \quad (11)$$

(taken from [11])

Where s_k^u represents static features and h_k^r dynamic features. The second encoding proposed in [14] is based on Hidden Markov Models but is not discussed here because it is out of the scope of this thesis.

2.11 Breadth-First Beam-Search

Beam-Search is a heuristic based search algorithm designed to alleviate the problems which arise with large search spaces. A version of Beam-Search, the *Breadth-First Beam-Search* uses breadth-first[2] search to build its exploration graph and then prioritizes the nodes based on some heuristics. But unlike the simple breadth-first search, it only keeps a predefined

[2] Breadth-first means the search-space exploration is started from the root and all the neighboring (i.e. on the same level) nodes are explored first, before moving on to the next level.

number of nodes as candidates, making it a computationally much lighter approach. These characteristics are essential for PBPM as it aims at making predictions at run time.

Algorithm 2: Breath First Beam Search

```
1:  q_1 = PriorityQueue()
2:  q_2 = PriorityQueue()
3:  q_1.put(initial node)
4:  while solution is not found do
5:    while q_1 is not empty do
6:      node = q_1.get()
7:      if node is solution then
8:        return node
9:      else
10:     candidates = node.expand()
11:     q_2.put(beamSize best elements from candidates)
12:   end while
13:   q_1 = beamsize best elements from q_2
14:   q_2 = PriorityQueue()
15: end while
```

Figure 5 Pseudo-code for the Beam Search Algorithm (source [2])

Algorithm 2 starts by initializing two priority queues[3] q_1 and q_2 (lines 1 and 2). q_1 is then initialized with the initial node (line 3). In the case of process prediction the initial node would indicate a prefix $p_k(\sigma)$. The algorithm then starts an iteration over the nodes stored in q_1 by starting from the most promising[4] node (line 6). If the currently active node is the solution then it is returned (line 8), otherwise the node is expanded (line 10) and the elements are added to $candidates$ (line 10). After all the elements from q_1 are expanded, $beamSize$ most promising next nodes from $candidates$ are added to q_2 (line 11). The priority queue q_1 is then repopulated with the elements from q_2 (line 13) and q_2 is emptied (line 14). The process is then repeated until the solution is found.

[3] In computer science, a priority queue is an abstract data structure where every element is associated with a priority. The priority queue serves elements starting with element associated with the highest priority.
[4] By most promising, we mean the node which has the highest priority.

3 Related Work

The following section covers the state-of-the-art research papers concerning predictive business process monitoring which are considered relevant in the scope of this thesis. The related work is categorized as follows: first a chronologically ordered overview of the early approaches in predicting business process outcomes is given. Secondly, we explore the work related to prediction of trace suffixes. Thirdly we describe the methods which use deep learning and finally the current state-of-the-art in leveraging a-priori knowledge during the prediction of trace outcomes is described in depth.

3.1 The Early Approaches

Most of the early works on predictive process monitoring focus on predicting binary outcomes. [26][27][28] use *decision trees* to predict whether an ongoing trace accommodates a certain *Service Level Agreement.* [29] uses *Support Vector Machines* to predict the final performance of an ongoing trace, i.e. whether or not the trace will be executed continually. [30] tackles the same prediction task with a clustering based approach and [31] with *decision trees*. The next distinguishable wave of research is predicting the remaining cycle time of an ongoing trace. [32] uses boosted regression, [33] proposes a prediction technique based on Hidden Markov Models, [34] uses annotated transition systems and [35][36] use predictive clustering trees. The remaining cycle time prediction has continually received a lot of attention in the research community; some of the later work include [38][39][40][41][42][43][44]. Also a noticeable sub-cluster of research is predicting binary outcomes in the form of LTL-violations – [45] uses *decision trees* while [11][48] use *random forests*.

3.2 Predicting Next Events

The work done in [37] kick-started another wave of predictive monitoring techniques – the prediction of next events. The authors of [37] propose an approach based on *Hidden Markov Models* with sequence alignment to predict a single next event of an ongoing trace. Other approaches quickly followed. In [23] a clustering based approach with a *sliding window model* is proposed for predicting the next activity and its attributes. A *predictive clustering tree* (PCT) is constructed similarly to standard decision trees by choosing a test at each internal node given an evaluation function. In order to predict multiple target variables at once, the clusters are identified based on both the descriptive space and the target space X, Y. In [24] the authors propose an approach based on *Markov chains* to estimate the probability of a potential future task happening in an ongoing trace. The paper also provides an analysis and discussion about the problem of dealing with loops and parallelism in a business process model.

The authors of [25] tackle a more ambitious prediction task of predicting the complete suffix on an ongoing trace. With the motivation of predicting the cash flow of a Dutch hospital, a method using a shortest path algorithm over a process graph is proposed. Their work is separated into four prediction tasks: a) a classification model is applied to a prefix $p_k(\sigma)$ to predict the case outcome Y b) a function $cost(Y)$ is applied to the outcome to determine its cost c) given a process graph G the shortest path from $\pi_A(e_k)$ to Y is mined – it is important to note that the shortest path is mined from the current activity, not from the whole prefix d) the mined suffix $s_k(\sigma)$ then determines the duration of the trace. The authors also address a problem of three kinds of noise in their data that were encountered during the experimental setup. Namely *sequence noise* which are deviances in the structure of the control-flow, *duration noise* i.e. errors in the timestamps of the activities, and *human noise* which are human

made errors e.g. incorrect diagnosis in a care taking process. Our approach differs from [25] in the sense that we predict next events incrementally until the end of the trace while [25] works the opposite way, starting the predictions from the end of the trace. The authors of [18] use Probabilistic Finite Automata (PFA)[19] to predict the next activity of an ongoing trace. The paper introduces a modification of the PFA based on Bayesian regularization which alleviates the overfitting problem characteristical of regular PFA. The Bayesian framework enables to take into account that extreme probabilities are unlikely to happen and ensures a smoother estimation of probabilities.

3.3 Predicting Next Events Using Deep Learning

The method introduced in [20] is the first in the field to use recurrent neural networks with LSTM cells for predicting the next event of a running trace. Most of the prior work before [20] is explicitly reliant on a process model in giving estimations about the future activities. The use of deep learning enables to leverage the actual execution history of the underlying business process. Although the results are not ground breaking, this paper gives a kick start for using deep learning in predictive monitoring and explores different neural network architectures suitable for this scope. The authors also investigate the inclusion of the resource attribute using a *simple symbolic encoding* both in the *predictors* and the *predictands*. The method used is the following: the activity alphabet size $|A|$ is expanded by concatenating every element in A with every element in the attribute vocabulary R (in this case the resource attribute). E.g. if $|A| = 5$ and $|R| = 3$ then the final alphabet size used for training the RNN would have size $5 * 3 = 15$. This enables to additionally predict the next resource of an ongoing trace and it showed that including the resource attribute in the predictor increases the precision of predicting the next event. This conclusion is also one of the underlying motivations for this master thesis. Although, differently from [20], we use *complex symbolic encoding* for including an additional attribute.

The work presented in [21] is a direct continuation of [20] where the authors explore various neural network architectures and parameters and take counter measures to prevent overfitting [22], a well known problem in the world of deep learning. The method is expanded to predict suffixes, and encoding of temporal information is explored. The paper also includes a method for visually representing what the neural network has learned. The surpassing results demonstrate the usefulness of using deep learning in predictive business process monitoring and motivates this thesis to also use deep learning.

The use of deep learning is further exploited in [1] where the authors investigate how LSTMs perform under different prediction tasks, event logs and prediction points. A method for predicting the next activity simultaneously with its timestamp is proposed. Two functions f_a^1 and f_t^1 are learned to predict the next most probable activity and its timestamp accordingly such that $f_a^1(p_k(\sigma)) = p_1(s_k(\pi_A(\sigma)))$ and $f_t^1(p_k(\sigma)) = p_1(s_k(\pi_T(\sigma)))$ where $\pi_A(e) = A$ is a function that assigns an activity to an event and $\pi_T(e) = T$ is a function that assigns a timestamp to an event. The neural network is trained and later queried by encoding the prefixes using complex symbolic encoding that allows to include also time-related features into the matrix. The authors of [1] describe an algorithm where every event of a trace is encoded into a vector with size $|A| + 3$ where the extra three slots represent the time increase since the last occurred activity, the time since midnight[5] and the time since the beginning of the week[6] respectively. This approach enables the prediction of time-related

[5] The time since midnight allows to differentiate business hours from non-business hours.
[6] The time since the beginning of the week allows to distinguish the weekday.

information. The generalized approach is then empirically compared against ad-hoc solutions at different prediction points. The experiments prove once again the usefulness of deep learning in predictive monitoring. We highlight the fact that differently from the approach proposed in this thesis, none of the previously described approaches takes into account a-priori knowledge while making predictions.

3.4 Predicting with A-priori Knowledge

To the best of our knowledge the only approach that leverages a-priori knowledge while making predictions is introduced in [2]. Therefore this work is considered as the baseline for this thesis and described in depth. The authors base their work on [1] and use *Recurrent Neural Networks* with LSTM cells as a prediction model for predicting the suffix of an ongoing trace. Two methods are proposed to increase the accuracy of the current state of the art. The first method called *NoCycle* tackles the problem of cyclic traces which cause the prediction of unnaturally long cyclic suffixes, i.e. the prediction algorithm getting caught in a loop. The second method called *A-Priori* is a further extension of the latter and allows to take into account a-priori knowledge in the form of LTL rules.

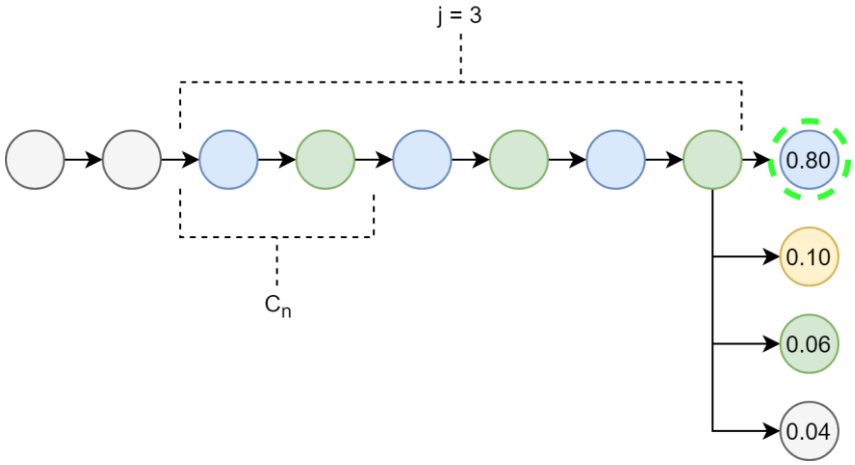

Figure 7 Probability Estimation with Cycles Present

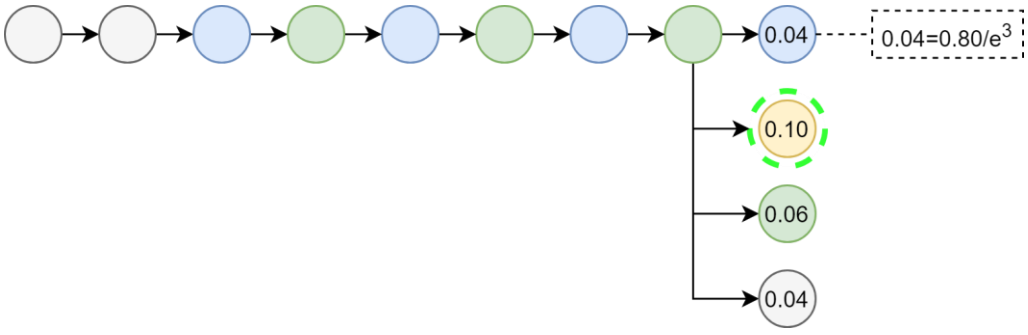

Figure 6 Probability Estimation with Cycles Present and NoCycle Algorithm Applied

The research conducted by the authors of [2] and [1] showed that the state of the art approaches work less efficiently on logs which contain traces with high number of cycles. The problem is traced down to the fact that frequent presence of cycles causes the RNN to deviate from the correct continuation of probability distributions, i.e. the RNN gets stuck in always predicting the first element of the cycle after the last element of the cycle. The *Nocycle* method offers remedy to this problem by weakening the back-loop probability. In the

light of this an extension is proposed to Algorithm 1 by the following means: for every prefix $p_k(\sigma)$ the algorithm checks if there are more than one consecutive cycles $C_n = \langle a_{c_1}, .., a_{c_n} \rangle$ present such that $a_{c_n} = a_k$. After the idendification of cycles, the probability of a_{c_1} occuring is decreased by $\frac{P(a_{c_1})}{e^j}$ where j is the number of consecutive occurences of C_n. Figure 6 visualizes how the next symbol is normally chosen in the presence of consecutive cycles while Figure 7 depicts how the *NoCycle* algorithm alters the decision making.

The pseudo-code of the *NoCycle* algorithm is presented in Algorithm 3. The proposed algorithm is identical to Algorithm 1 with the exception of line 7, where the weakening function is applied to the probabilities returned by the LSTM.

Algorithm 3: NoCycle Extension for Algorithm 1

```
1:  function PredictSuffixNoCycle(p_k(σ), lstm, max)
2:      i = 0
3:      trace = p_k(σ)
4:      do
5:          trace_encoded = encode(trace)
6:          next_symbol_prob = PredictNextSymbols(lstm, trace_encoded )
7:          weak_next_symbol_prob = weakenProb(trace, next_symbol_prob)
8:          next_symbol = GetSymbol(next_symbol_prob, trace_encoded )
9:          trace = trace · next_symbol
10:         i = i + 1
11:     while (not next_symbol == end_symbol) and (i < max)
12:     return trace
13: end function
```

Figure 8 Pseudo-code for the NoCycle Algorithm (source [2])

The *A-Priori* algorithm is based on Algorithm 3 to estimate the probabilities of next activities, but instead of predicting just the most probable suffix, it explores a set of possible suffixes and chooses the most probable suffix that is compliant with the a-priori knowledge available, represented in the from of LTL constraints. In order to avoid computational over flow by exploring all the possible suffixes, the *beam search* algorithm is applied. This allows to prune suffixes which are heuristically not promising. Algorithm 4 depicts the inner-workings of the *A-Priori* algorithm. It takes as parameters the prefix $p_k(\sigma)$, the trained LSTM model $lstm$, an integer value max representing the maximum number of iterations allowed, beam size $bSize$ which determines how many possible next symbols are returned at each explored node, $maxSize$ representing the maximum number of branches explored at the same time, and the available a-priori knowledge $K(\sigma)$. On line 2 a counter is initialized to determine the number of iterations conducted. On line 3 a priority queue for storing $maxSize$ unexplored branches is initialized with the initial prefix $p_k(\sigma)$. The algorithm then starts to iterate over the priority queue until a conformant trace is found or the maximum number of iterations is reached (line 4). For each trace in $prefixes$, $bSize$ next possible activities are predicted and stored in $candidate\_next$ as a concatenation of the prefix and the predicted activity (line 5). This function is carried out by Algorithm 3, where the $nocycle$ algorithm is used to prevent the problems caused by cyclic traces. $canditate\_next$ is a $|prefixes| * bSize$ sized priority queue out of which the $maxSize$ most probable traces are obtained (line 6). The algorithm then iterates over $top\_candidates$ (line 7). If the last symbol of $candidate$ is not the end symbol then it is added to the $prefixes$ priority queue (lines 8-9) and in the case the last symbol is end_symbol, a check is conducted on $candidate$ to see whether or not it is compliant with the a-priori knowledge $K(\sigma)$ (line 11).

If that is the case then $candidate$ is returned (line 12). If no trace is returned the iterations counter i is increased (line 16) and the next $maxSize$ paths are explored.

Algorithm 4: A-Priori Algorithm

```
1:  function Apriori(p_k(σ), lstm, max, bSize, maxSize, K(σ))
2:      i = 0
3:      prefixes = {p_k(σ)}
4:      while k ≤ max and not empty(prefixes) do
5:          candidate_next = predictPrefNextSymbols(lstm, prefixes, bSize)
6:          top_candidates = topRank(candidates_next, maxSize)
7:          for all candidate in top_candidates do
8:              if last_symbol(candidate) <> end_symbol then
9:                  push(candidate, prefixes)
10:             else
11:                 if check(candidate, K(σ)) then
12:                     return candidate
13:                 end if
14:             end if
15:         end for
16:         i = i + 1
17:     end while
18: end function
```

Figure 9 Pseudo-code for the A-Priori Algorithm (source [2])

4 The Problem

Predicting the future unfolding of an ongoing trace has attracted a lot of attention amongst the PBPM community in the past decade. The reason behind it is the emergence of accurate system tracking software and computers capable of high-level data processing. This has created the conditions to generate valuable insights about the future of organizations. Having a solid understanding of the future unfolding of a company business flow can help its managers with budget planning, resource allocation and taking timely counter measures to a possible mishap in the business flow. Taking into consideration that deviations inevitably happen in small and large organizations, we can assume that some conditions will often emerge that could alter the normal execution. For example a hospital could have a technical issue which leads to one or many surgery rooms to be out of order. This could cause significant delays in the treatment process of a patient or have an effect on the control-flow of parallel executed traces. One might also think of a situation where new regulatory requirements have been suddenly forced to take effect, e.g. an additional analysis has been deemed obligatory before a certain treatment. This may result in substantial deviations in the upcoming control-flow. Unprepared, the organization under question may end up in a "bottleneck" situation where the current financial or resource planning is not viable. This could mean that the organization is not able to fulfil its operational goals or obligations. In the case of a hospital this could mean a patient not receiving a treatment in time, although, when leveraging the freshly emerged knowledge in the course of predictive monitoring, it would be possible to foresee the possible unwanted culminations. Realizing the fact that often times this type of knowledge emerges at runtime means that an already trained predictive model would not be able to adapt to the new situation. That is why a solution for injecting a-priori knowledge to the predictive model at runtime is necessary. The solution provided in [2] is a great kick start to offer remedy to this problem. The proposed approach is able to leverage a-priori knowledge on the control-flow of an ongoing business case and shows the potential of this type of techniques.

Now let us consider another example in the hospital scenario. A hospital worker (a *resource* from the perspective of a business process) suddenly being sick would mean a temporary unavailability of the resource. This means that the activities usually carried out by this worker have to be carried out by another worker. This could have a major impact on the future resource allocation and possibly affect the control-flow causing unexpected situations to occur. By ruling out the usage of the prediction of the unavailable resource it would be possible to forecast these situations. The problem lies in the fact that this kind of knowledge is not reflected in the control-flow and therefore can not be leveraged by using the approach introduced in [2]. Currently there exists no solution which would allow to leverage multi-perspective knowledge while making predictions. This leads us to research question 1.

RQ1: How does *leveraging multi-perspective a-priori knowledge* affect the prediction accuracy of multi-perspective sequences?

In the course of answering research question 1, another unexplored area revealed itself. Although experiments conducted in [20] explore how the inclusion of the resource attribute in the predictor affects the prediction accuracy, they do so by concatenating the resource with the activity, though still remaining in the scope of simple symbolic sequence encoding. The work conducted in this paper explores the same area using complex symbolic encoding which leads us to research question 2.

RQ2: How does the inclusion of an additional categorical data attribute to the predictor using complex symbolic sequence encoding affect the prediction accuracy?

5 Conceptual Framework

Table 3 Conceptual Framework

Predictor	
Category	**Explanation**
Control-Flow	
Control-Flow	Sequence of events
Payload	
Timestamps	Timestamps attached to individual activities
Case Attributes	Static attributes attached to a trace which remain the same throughout a case
Event Attributes	Dynamic attributes attached to individual events
Derived values	A sequence of values derived from existing attributes
Intercase	
Intercase Features	Features related to more than one concurrent trace, e.g. the number of cases running concurrently
Predictand	
Category	**Explanation**
Control-Flow	
Next Activity	A single next activity of the ongoing trace
Sequence of Next Activities	The entire sequence of next events until the end of the trace
Binary Case Outcome	The binary end result of a case based on the control-flow, e.g. whether or not a loan is granted
Categorical Case Outcome	The categorical end result of a case based on the control-flow
Payload	
Categorical Sequence	A sequence of categorical event attributes
Continuous Sequence	A sequence of continuous event attributes
Binary Case Outcome	The binary end result of a case based on an attribute or a combination of attributes. i.e. this can be any yes or no question which can be answered based on the payload
Specific Categorical Value	A categorical value of a specific attribute belonging to a specific event, e.g. what is the diagnosis of a particular patient on a particular analysis going to be
Specific Continuous Value	A numerical value of a specific attribute belonging to a specific event
Next Timestamp	A single timestamp of the next event
Remaining cycle time	The remaining cycle time until the end of the trace
Sequence of timestamps	The sequence of timestamps corresponding to the predicted event sequence
A-priori	
Category	**Explanation**
Control-Flow	
LTL Constraints	Constraints over the control-flow expressed with LTL rules, e.g. activity A has to be followed by activity B
Intercase Constraints	Constraints spanning over more than one trace
Payload	
Timestamp Constraints	Constraints over timestamps, e.g. activity B has to happen within 600 seconds after activity A
Case Attribute Constraints	Constraints over case attributes, e.g. a patient who is older than 70 years has to undergo analysis C
Event Attribute Constraints	Constraints over event attributes, e.g. activity A has to be performed by resource C
Aggregate Intercase Constraint	Constraints spanning over the complete event log, e.g. the waiting time between two events should be less than 25 minutes
Concurrent Trace Constraints	Constraints on a data value over concurrent traces, e.g. a resource can not be allocated to multiple cases simultaneously
Multi-perspective	
MP-Declare Constraints	MP-Declare constraints are a combination of control-flow and payload constraints.

When talking about predictive process monitoring with a-priori knowledge, we can identify, besides the type of information we want to leverage for learning the predictive model, and what we want to predict, also the type of a-priori knowledge we want to consider. Moving into the multi-perspective domain, taking into account not only the control-flow, but also the payload, further increases the number of possible scenarios that can be considered.

In this section we provide a conceptual framework including all these different scenarios which unbundles the different possible conditions and is explained in Table 3. Namely, we have defined 3 dimensions – the *predictor*, which is the type of data used for building the prediction model, the *predictand*, which is the type of prediction being made and *a-priori*, which represents the type of a-priori knowledge applied. Combinations of the first two dimensions are used for different prediction tasks. Usually, the predictor type matches the predictand type, e.g. when predicting the sequence of next activities then work flow is used as the predictor, when predicting the next timestamp, timestamps are used as predictors, and so on. Although it is difficult to predict a predictand without the corresponding predictor, often times a combination of predictors are used to predict a single predictand, e.g. adding case attributes to the control-flow predictor could significantly boost the prediction accuracy when predicting the sequence of next activities. On top of the possible combinations originating from the *predictor* and *predictand* dimensions, we can apply the different types of a-priori knowledge defined in Table 3. In this light, we can imagine the whole conceptual framework as a three-dimensional *hypercube*. In this thesis we will focus on a specific point of this *hypercube* where we use:

1. as predictors, the *control-flow* together with an *event attribute*,
2. as predictands, the *sequence of next activities* and a *categorical sequence* associated to these activities,
3. and as a-priori, *MP-Declare constraints* spanning over the predictands.

We also compare our work against the work done in [2] which can be placed in the *hypercube* as follows:

1. as predictors, the *control-flow* is used,
2. as predictans, the *sequence of next activities* is used,
3. and as a-priori, *LTL constraints* are used.

6 Architecture of the Solution Proposed

In this section a solution is proposed which enables to leverage multi-perspective a-priori knowledge on top of an already trained neural network. This is a direct extension for the work done in [2], that leverages a-priori knowledge on the control-flow in the form of LTL-rules. It is important to stress again the fact that similarly to [2] we propose a solution that enables to leverage knowledge on top of an already trained neural network. This is necessary due the fact that a-priori knowledge is dynamic and could change from case to case. Re-training the neural network for each prediction would quickly lead to scalability issues.

6.1 Predicting an Additional Categorical Attribute

Since we are dealing with sequential predictions, the first milestone for leveraging multi-perspective a-priori knowledge is to additionally predict the data attribute we wish to apply a-priori knowledge to. In this thesis we focus on the *resource* attribute. Complex symbolic encoding (see 2.10) is used to map the additional attribute to the event feature vector used for training the LSTM and later querying it. We use an LSTM architecture with one shared layer and two single task layers as visualized in Figure 10. The proposed LSTM architecture outputs two probability distributions for the activity and resource alphabets respectively.

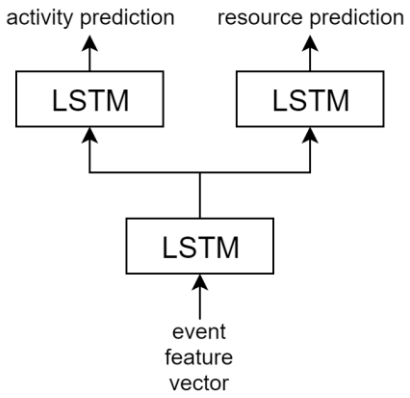

Figure 10 LSTM Architecture of the Proposed Solution

The prediction of an additional categorical attribute is also possible with an LSTM architecture that outputs only one sequence. Namely the activity and data attribute alphabets can be concatenated, creating a unified alphabet. Though, a high number of categorical values could rapidly result in a state space explosion. In theory, having two separate sequences, sets the foundation for a more scalable solution.

6.2 Beam-Search with Data Payload

Having two separate probability distributions emerges as a problem when applying the Beam search algorithm for exploring the prediction graph. In a simple scenario the algorithm always explores the element with the highest probability first, then the element with the second highest probability and so on. Following the same linear pattern with two separate probability distributions would leave neglected the different possible combinations of the two alphabets. To tackle this problem, we create a probability matrix with size $|A| * |R|$, where A and R are the activity and resource alphabets respectively. The values in the matrix

represent the sum of the natural logarithms of the probabilities[7]. Mathematically this can be expressed as follows:

$$P(a_i r_j) = \ln(P(a_i)) + \ln(P(r_j)) \quad (12)$$

The beam size number of elements of the probability matrix are then explored decreasingly starting from the most probable activity-resource combination. Table 4 depicts a probability matrix of size $3 * 5 = 15$ where probabilities are ordered decreasingly starting from the upper left corner. The grey cells represent the $beam\ size = 3$ elements which are explored by the beam search algorithm. We can see how this approach removes the linearity and explores the search space in a more sophisticated way. The same approach can easily be expanded to multiple data attributes by expanding the probability space with the product of all alphabet sizes under consideration.

Table 4 Probability Matrix for Beam-Search

	$\boldsymbol{P(a_1)}$	$\boldsymbol{P(a_2)}$	$\boldsymbol{P(a_3)}$	$\boldsymbol{P(a_4)}$	$\boldsymbol{P(a_5)}$
$\boldsymbol{P(r_1)}$	$P(a_1 r_1)$	$P(a_2 r_1)$	$P(a_3 r_1)$	$P(a_4 r_1)$	$P(a_5 r_1)$
$\boldsymbol{P(r_2)}$	$P(a_1 r_2)$	$P(a_2 r_2)$	$P(a_3 r_2)$	$P(a_4 r_2)$	$P(a_5 r_2)$
$\boldsymbol{P(r_3)}$	$P(a_1 r_3)$	$P(a_2 r_3)$	$P(a_3 r_3)$	$P(a_4 r_3)$	$P(a_5 r_3)$

A further visualization of how the two separate probability distributions are combined can be found in Figure 11. The blue circles represent the activity sequence of a prefix $p_k(\sigma)$ and the green circles represent the resource sequence of the same prefix.

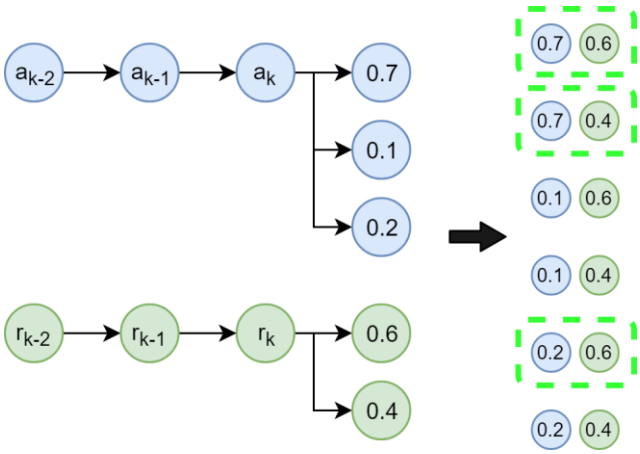

Figure 11 Node Expansion with Data Payload

6.3 Multi-perspective Knowledge Expression

For expressing multi-perspective a-priori knowledge, we propose the usage of MP-Declare. Like mentioned earlier (see 2.5) it is a declarative process modelling language which enables to specify constraints over traces including constraints about data attributes. The language is suitable to represent a-priori knowledge taking into consideration control-flow and payloads. It indeed does not create a "closed world" effect where the prediction model can only choose between the specified behaviour. MP-Declare merely states what behaviour is not

[7] Instead of multiplying the simple probabilities, we calculate the formula in the logarithmic space to avoid computational over-flow due to the recursive dynamics of the inference algorithm.

allowed and guides the prediction model to more accurate predictions by rejecting non compliant predictions. Note that MP-Declare can still be used for specifying only simple knowledge about the control-flow. The language also supports time interval specifications, which enables to set specific constraints on the timestamps of activities. These characteristics make MP-Declare a scalable solution for expressing complex types of a-priori knowledge. Leveraging a-priori knowledge about timestamps is motivation for future work.

6.4 Leveraging Multi-Perspective A-priori Knowledge

Algorithm 5: MP A-Priori algorithm

```
1:  function Apriori(p_k(σ), lstm, max, bSize, maxSize, K_MP(σ))
2:    i = 0
3:    prefixes = {p_k(σ)}
4:    while k ≤ max and not empty(prefixes) do
5:      candidate_next = predictPrefNextSymbolsMP(lstm, prefixes, bSize)
6:      top_candidates = topRank(candidates_next, maxSize)
7:      for all candidate in top_candidates do
8:        if last_symbol(candidate) <> end_symbol then
9:          push(candidate, prefixes)
10:       else
11:         if check(candidate, K_MP(σ)) then
12:           return candidate
13:         end if
14:       end if
15:     end for
16:     i = i + 1
17:   end while
18: end function
```

Figure 12 Pseudo-code for Multi-perspective Declare Algorithm (source [2])

We extend Algorithm 4 by replacing the LTL conformance checker module with the MP-Declare conformance checker module. Algorithm 5 depicts the pseudo-code for the Multi-Perspective A-priori algorithm. Instead of the simple a-priori knowledge, Algorithm 5 takes as a parameter the more granular knowledge $K_{MP}(\sigma)$ in the form of MP-Declare constraints. Similarly to Algorithm 4, a priority queue $prefixes$ is initialized (line 3) with the initial prefix $p_k(\sigma)$ given as input. Line 5 then expands all the traces from $prefixes$ with $bSize$ nodes by using the beam search algorithm adapted to multi-perspective predictions (see 6.2). The $maxSize$ most probable traces are then obtained (line 6). The $top\_candidates$ are then iterated over, to check whether an end event symbol has been predicted (line 8). If the last symbol of the trace in $top\_candidates$ is not the end event symbol then it is added to the $prefixes$ priority queue. If the last symbol is the end symbol, then a conformance check is conducted with the MP-Declare knowledge $K_{MP}(\sigma)$ (line 11). The trace is returned as the final prediction if the check gives a positive response (line 12). Otherwise the algorithm continues to explore the search space until it exhausts $prefixes$ or reaches the maximum number of iterations allowed max.

7 Implementation Details

A prototype of the proposed solution has been implemented in Python 2.7. We use the Keras[8] library together with the Tensorflow[9] framework for building the recurrent neural networks. Tensorflow allows parallel GPU computing and offers RNN specific functionalities, making it a suitable solution for online predictions where time efficiency is important. We train the RNNs with batch size of 20 and with the maximum number of epochs of 100.

The conformance checker module is a separate Java application and is accessed by the Py4J[10] library which is a gateway for accessing Java objects from a Python program. For checking the conformance of the a-priori knowledge, we use a lightweight adaptation of the Declare Analyzer [13] plugin implemented for the ProM toolkit. Originally the plugin takes as input an entire event log with a Declare model[11] and provides an aggregated report together with a detailed analysis about the activations, fulfilments and violations of each individual Declare rule with respect to each individual trace in the event log. It also has a visual component for visualizing the event logs' conformance with respect to the Declare model. The plugin is adapted according to the needs of Algorithm 5, i.e. the functionality is reduced down so it takes as input a single trace with a declare model and returns a boolean value representing whether or not the trace is compliant with the a-priori knowledge specified. A trace is compliant if there are no violations of the a-priori knowledge. The complete source code of the implementation is available in github[12].

[8] https://keras.io/
[9] https://www.tensorflow.org/
[10] https://www.py4j.org/
[11] A declare model is a xml-based document for specifying declare rules.
[12] https://github.com/kaurjvpld/Incremental-Predictive-Monitoring-of-Business-Processes-with-A-priori-knowledge

8 Evaluation

In this section we provide a comparison of the proposed approach leveraging multi perspective a-priori knowledge against two baseline methods. The first baseline method is proposed in [1] and uses LSTM neural networks for predicting the suffix of an ongoing trace without using any a-priori knowledge. The second baseline method is [2] which uses a-priori knowledge only on the control-flow. Both the baseline methods are direct predecessors to our approach as we extend the work done in [2] and the work done in [2] is, in turn, based on [1]. We thoroughly examine how the three approaches perform with logs with different characteristics and also explore how the inclusion of an additional data attribute by training the RNN with complex symbolic sequences affects the prediction accuracy compared to only using the activity sequences. By doing this, we answer the research questions defined in Section 4 and reviewed here:

RQ1: How does *leveraging multi-perspective a-priori knowledge* affect the prediction accuracy of multi-perspective sequences?

RQ2: How does the inclusion of an additional categorical data attribute to the predictor using complex symbolic sequence encoding affect the prediction accuracy?

A similar study to the latter has been made in [20] with the difference of encoding the resource attribute in a simple symbolic sequence (see 4). But as stated by the authors of [20] the feasibility of this approach is limited as the number of resources in large organizations is quite high. Therefore the alphabet would quickly become a computational burden. Our approach treats both the activities and resources as separate sequences avoiding the explosion of the alphabet size.

8.1 Experimental Framework

To test our approach, we defined an experimentation framework based on the three dimensions of the conceptual framework defined in Section 5. The first dimension is the information which is used for training the prediction model, i.e. the *predictor*. Both the baseline methods are being tested with a prediction model that is trained only with the control-flow and with a model that is trained using the control-flow together with the corresponding resources; the proposed approach, instead, is only being tested with a model trained using both control-flow and resource information, as, by construction, it needs to be trained with complex symbolic sequences. The second dimension is the *predictor*, for which we follow the same pattern as for the first dimension. For the baseline methods we predict a) only the sequence of activities and b) the control-flow together with the resource attribute. As for the proposed approach we predict the control-flow together with the resources. The third dimension is the presence of a-priori knowledge in the scope of which, the effect of MP-Declare rules with different complexity and strength is explored.

As a further aspect, we explore the effect of the activity and alphabet sizes characterizing the logs on the three approaches. We hypothesize that the alphabet size of the attributes has an effect on the performance of the multi-perspective A-priori algorithm. The idea is that the bigger the alphabet size the more difficult it is for the algorithm to find compliant traces as the search space increases.

8.2 Event logs

Basing our decision on the experimental framework, we use synthetic logs in our experimentation, as it gives us the possibility to test our approach on logs with systematic characteristics. Motivated by alphabet sizes and a-priori injection, we created 20 synthetic logs

with the characteristics defined in Table 5. The horizontal header represents the size of the activity and resource alphabets respectively, e.g. the column header $10x5$ indicates a log with activity alphabet of size 10 and resource alphabet of size 5. The vertical header represents the a-priori knowledge that is injected to the log, where W represents a weak rule (see 8.3) and S represents a strong rule (see 8.3), e.g. the row header $1W$ indicates a log where

Table 5 Log Characteristics

	10x2	**10x5**	**10x20**	**5x5**	**50x5**
1W	Log 1	Log 5	Log 9	Log 13	Log 17
3W	Log 2	Log 6	Log 10	Log 14	Log 18
1S	Log 3	Log 7	Log 11	Log 15	Log 19
3S	Log 4	Log 8	Log 12	Log 16	Log 20

1 weak rule is injected.

The logs were generated with a prototypical tool[13] which enables to generate synthetic logs based on multi-perspective declarative process models by generating a specified amount of random traces compliant with the process model. Each of the synthetic logs consists of 2000 traces and the trace lengths vary from 9 to 17. We stress the fact that declarative models have an open-world assumption which means that the traces are not necessarily following a strict structure and have a certain level of unpredictability in them, i.e. we do not know for sure what trends an RNN is able to pick up from the logs.

We additionally tested our approach on the BPI 2017[14] dataset to prove that the proposed approach is able to perform well also on a real life log. We used a small fraction of the log which contains 3000 traces (approximately 10% of the complete log). The activity alphabet size of the fraction of traces we use is 25 and the resource alphabet size is 80.

8.3 Knowledge injection

We aim at testing a-priori knowledge with different levels of strengths, and therefore distinguish two strength levels of a-priori knowledge – *weak a-priori knowledge* and *strong a-priori knowledge.* In terms of MFOTL we define *weak a-priori knowledge* as of type $F_I(A \wedge \exists x.\, \varphi_a(x))$ which represents the MP-Declare template *existence* and the *strong a-priori knowledge* as of type $(G(\forall x.\, ((A \wedge \varphi_a(x)) \rightarrow F_I(B \wedge \exists y.\, \varphi_c(x,y)))))) \wedge (F_I(A \wedge \exists x.\, \varphi_a(x)))$ which represents the *response* template together with an *existence* template imposed on the activation A with φ_a holding true.

During the log generation we specified two declarative process models for each log:

1. The first model characterizes the normal behaviour of the process and is used for generating 80% of the traces of the log
2. The second model is identical to the first one, with the exception that some additional constraint(s) are specified which are forced to take effect in every trace. This model is used to generate the remaining 20% of the cases.

We then concatenated the two sets of traces and shuffled the traces to form the log. For every newly formed log, we consider the additional constraints enforced in the second model as the a-priori knowledge.

[13] https://github.com/darksoullock/MPDeclareLogGenerator
[14] https://data.4tu.nl/repository/uuid:5f3067df-f10b-45da-b98b-86ae4c7a310b

8.4 Rule Mining

When it comes to the BPI2017 log, we did the opposite, i.e. instead of injecting knowledge, we mined already existing rules with a prototypical tool[15]. The tool enables to mine specific MP-Declare rules with respect to the specified data attributes. In our case we mined *response* (see 2.5) rules. To find a rule comparable to the knowledge injected in the synthetic logs, we imposed *existence* (see 2.5) on the activation of the *response* rule to create *strong a-priori knowledge*. We then used the *DeclareAnalyzer* plugin to analyse the knowledge with the aim of identifying a rule which has a support of ~0.20, i.e. the rule is activated and fulfilled in approximately 20% of the traces. We took the first rule that met the requirements and used it as *strong a-priori knowledge* and the *existence* of the activation of that rule, as *weak a-priori knowledge*.

8.5 Experimental Procedure

Every log was separated into training and testing sets with sizes of 67% and 33% respectively. During the testing phase we selected, out of the testing set, only the traces which are compliant with the injected a-priori knowledge and used the same testing set for each inference algorithm in the course of one log. This procedure aims at simulating an effect where a-priori knowledge has emerged during a process execution. We then predicted suffixes at four different prediction points, i.e. we used prefixes with four different lengths. The prefix lengths were determined by the median trace length and were in the range of $\{k-2;\ k-1; k; k+1\}$, where k is the median trace length divided by 2. For baseline 2 we set the beam size equal to 3 and for the *MP A-priori* we set the beam size equal to 5. The bigger beam size for the *MP A-priori* is justified by the fact that the multi-perspective a-priori knowledge is more constraining than simple a-priori knowledge, hence the beam search needs to explore a bigger amount of nodes before finding a compliant trace.

[15] https://github.com/volodymyrLeno/CorrelationMinerForDeclare

8.6 Results and Discussion

Table 6 Results on Synthetic Logs

	Baseline 1				Baseline 2				MP A-Priori	
Predictor →	A		A + R		A		A + R		A + R	
Predictand →	A	R	A	R	A	R	A	R	A	R
10x2 | 1W	0.693	N/A	0.643	0.674	0.649	N/A	0.665	0.640	0.646	0.649
10x2 | 3W	0.742	N/A	0.806	0.682	0.742	N/A	0.806	0.682	0.803	0.679
10x2 | 1S	0.511	N/A	0.258	0.803	0.682	N/A	0.536	0.760	0.695	0.763
10x2 | 3S	0.658	N/A	0.629	0.793	0.682	N/A	0.662	0.826	0.667	0.833
10x5 | 1W	0.804	N/A	0.810	0.800	0.803	N/A	0.813	0.800	0.780	0.790
10x5 | 3W	0.729	N/A	0.723	0.605	0.719	N/A	0.723	0.605	0.831	0.824
10x5 | 1S	0.390	N/A	0.695	0.700	0.479	N/A	0.720	0.717	0.802	0.792
10x5 | 3S	0.765	N/A	0.831	0.741	0.769	N/A	0.846	0.761	0.849	0.761
10x20 | 1W	0.727	N/A	0.864	0.579	0.909	N/A	0.855	0.579	0.864	0.575
10x20 | 3W	0.784	N/A	0.761	0.583	0.784	N/A	0.723	0.571	0.838	0.738
10x20 | 1S	0.827	N/A	0.784	0.608	0.884	N/A	0.884	0.627	0.884	0.704
10x20 | 3S	0.849	N/A	0.774	0.566	0.556	N/A	0.620	0.449	N/A	N/A
5x5 | 1W	0.618	N/A	0.645	0.677	0.616	N/A	0.639	0.684	0.601	0.785
5x5 | 3W	0.729	N/A	0.749	0.574	0.729	N/A	0.750	0.574	0.754	0.758
5x5 | 1S	0.727	N/A	0.655	0.642	0.726	N/A	0.691	0.650	0.718	0.763
5x5 | 3S	0.409	N/A	0.806	0.644	0.413	N/A	0.806	0.643	0.846	0.679
50x5 | 1W	0.553	N/A	0.735	0.642	0.746	N/A	0.707	0.561	0.697	0.687
50x5 | 3W	0.314	N/A	0.506	0.308	0.674	N/A	0.819	0.514	0.735	0.742
50x5 | 1S	0.231	N/A	0.466	0.377	0.874	N/A	0.825	0.623	0.870	0.875
50x5 | 3S	0.623	N/A	0.802	0.803	0.732	N/A	0.789	0.799	0.811	0.815

We use the *Damerau-Levenshtein* similarity metric (see last paragraph of 2.9) as a comparison measure to showcase the difference between the predicted traces and the actual traces. A higher value indicates a more accurate performance. In Table 6 and 7, the grey cells indicate the best activity and resource accuracy for each log respectively, and to better understand the two tables we define *A* as sequence of activities and *R* as sequence of resources. We indicate the most accurate activity-resource combination for each log with bold cell borders.

Table 6 reports the results of the three inference algorithms' performances on the synthetic logs described in Table 5. The overall performance of the multi-perspective a-priori algorithm is positive and the results match our expectations – the dominant algorithm is *MP A-priori*. In terms of predicting the sequence of next activities, it managed to outperform the two baseline methods in 45% of the times and landed close to the most accurate result in the remaining 55% of the times. In terms of predicting the sequence of next resources, the *MP A-priori* algorithm managed to outperform the baseline methods in 75% of the times and followed the most accurate result closely in the remaining 25% of the times. We also compared the combined accuracies (a.k.a. multi-perspective predictions) of the activity and resource sequences where it was applicable. In this setting, the *MP-A-priori* outperformed the baseline methods in 75% of the cases (RQ1).

Table 6 does not indicate any noticeable trends in the sense of log alphabet sizes. The proposed approach shows steady performance throughout all logs, with the exception of Log 12, where the algorithm was not able find compliant traces with the given beam size. With further investigation we can notice that Log 12 (see Table 5) belongs to the group of logs with the biggest alphabet sizes and the most constraining a-priori knowledge, i.e. it contains 3 strong a-priori rules. The large search space combined with strong constraints explains the inability for the algorithm to perform well. Although Table 6 reports the performance of the algorithm with beam size 5, we also experimented the same test with bigger beam sizes, namely 10, 20 and 50. We discovered that when increasing the beam size extremely high

with respect to the alphabet size, the algorithm may exhaust the search space and forcefully find a compliant trace in the early iterations. By saying that, we mean that during the first iterations, the algorithm tries out a large fraction of all the possible activity-resource combinations and finds a compliant trace with very low probability which, most likely, is very inaccurate. This outlier result points out a vulnerability of the proposed approach. As future work we aim at developing an algorithm, which similarly to the *NoCycle* algorithm, is able to intelligently manipulate the probability distributions in order to locate the compliant trace with a smaller beam size without exhausting the search space, e.g. if the algorithm has predicted the activation for a *response* rule, then we can increase the probability of the target to take place.

The inclusion of the resource attribute to the predictor showed an increased accuracy in activity prediction in 60% of the cases, which is not a strong trend as we expected (RQ2). Although, this can be explained by the fact that the RNN outputs two separate probability distributions which are treated as two independent values. The RNN predicts the most probable next activity given the prefix $p_k(\sigma)$, and does the same for the next resource, given the prefix $p_k(\sigma)$. This means that the activity and resource are not treated as a single unit and eventually two sequences are accummulated which can be asynchronous. We aim at finding remedy to this problem as future work. We suggest that the RNN architecture should be accommodated as follows: (i) first, the most probable next activity a_{k+1} is predicted given the prefix $p_k(\sigma)$, (ii) then, the most probable next resource r_{k+1} should then be predicted given the prefix $p_k(\sigma)$ together with the already predicted next activity a_{k+1}. This approach would treat the activity and resource as a single unit and would possibly lead to more accurate predictions.

During the experimentation, we also noticed an interesting phenomenon; the detection of a fulfilled prediction constraint causes the RNN to predict the end event symbol of a trace prematurely. We see remedy to this problem in taking into consideration the average trace length (*intercase a-priori knowledge*) when making predictions, which has the potential to assist the RNN in predicting traces whose length is closer to the ground truth.

Table 7 Results on BPI 2017 dataset

	Baseline 1		Baseline 2		MP A-Priori	
Predictor	A + R		A + R		A + R	
Predictand	A	R	A	R	A	R
WEAK	0.700	0.266	0.695	0.269	0.636	0.400
STRONG	0.694	0.366	0.687	0.346	0.695	0.394

Table 7 reports the performances of the three inference algorithms applied to the BPI 2017 dataset. In this setting we used only one type of predictor, namely the control-flow combined with the resource. When applying weak a-priori knowledge, baseline 1 managed to outperform the other methods at predicting the activity sequence. However, we can notice a significant dominance of the *MP A-Priori* when predicting the resource sequence; and in the combination of activity and resource predictions, the proposed approach manages to outperform the two baseline methods. When applying strong a-priori knowledge, the proposed approach manages to outperform the baseline methods in both the activity and resource sequence predictions, confirming the positive results from Table 6 and proving that the proposed approach is able to successfully perform on a real life log (RQ1).

Based on the results we answer **RQ1** by assessing that leveraging multi-perspective a-priori knowledge improves the accuracy of multi-perspective predictions. Concerning **RQ2**, we

found weak evidence that the inclusion of an additional categorical data attribute to the predictor using complex symbolic sequence encoding improves the prediction accuracy. Although, we deem further research necessary to strengthen this statement.

9 Conclusion

The main contribution of this thesis is a technique for leveraging multi-perspective knowledge on top of an already trained RNN while making predictions of complex symbolic sequences. We proved that the proposed approach is able to provide state-of-the-art results and successfully expands the work done in [2]. We systematically proved that the approach is effective on synthetic logs with different characteristics and confirmed that the approach is able to perform well also on a real life log. The proposed approach can easily be extended to categorical attributes other than resources and it is scalable to leveraging a-priori knowledge on multiple categorical attributes simultaneously, including constraints on timestamps. The second contribution is a conclusion that the activity and resource predictions should not be treated as two separate output sequences. Rather they should be treated as a single prediction unit.

For future work we plan to address the problems discussed in Section 8.6. Firstly, we aim at finding a more suitable RNN architecture, which treats the control-flow and the corresponding attributes as a single unit. Secondly, we aim at finding a solution to the search space exhaustion problem with a more sophisticated search space exploration algorithm which enables to give advantage to possible traces which naturally accommodate the a-priori knowledge. Thirdly, we plan to explore the effects of leveraging intercase features, namely the average trace length time for traces with similar structures, to prevent premature end-event predictions. We also see great potential in using time-related a-priori knowledge when making predictions about timestamps and remaining cycle time.

10 References

[1] Tax, N., Verenich, I., La Rosa, M., & Dumas, M. (2017, June). Predictive business process monitoring with LSTM neural networks. In *International Conference on Advanced Information Systems Engineering* (pp. 477-492). Springer, Cham.

[2] Di Francescomarino, C., Ghidini, C., Maggi, F. M., Petrucci, G., & Yeshchenko, A. (2017, September). An Eye into the Future: Leveraging A-priori Knowledge in Predictive Business Process Monitoring. In *International Conference on Business Process Management* (pp. 252-268). Springer, Cham.

[3] Hochreiter, S., & Schmidhuber, J. (1997). Long short-term memory. *Neural computation*, *9*(8), 1735-1780.

[4] Bengio, Y., Simard, P., & Frasconi, P. (1994). Learning long-term dependencies with gradient descent is difficult. *IEEE transactions on neural networks*, *5*(2), 157-166.

[5] Navarin, N., Vincenzi, B., Polato, M., & Sperduti, A. (2017). LSTM Networks for Data-Aware Remaining Time Prediction of Business Process Instances. *arXiv preprint arXiv:1711.03822*.

[6] Gers F., Schmidhuber J. (2000) Recurrent nets that time and count. Proceedings of the International Joint Conference on Neural Networks

[7] Cho, K., Van Merriënboer, B., Gulcehre, C., Bahdanau, D., Bougares, F., Schwenk, H., & Bengio, Y. (2014). Learning phrase representations using RNN encoder-decoder for statistical machine translation. *arXiv preprint arXiv:1406.1078*.

[8] Yao, K., Cohn, T., Vylomova, K., Duh, K., & Dyer, C. (2015). Depth-gated recurrent neural networks. *arXiv preprint*.

[9] Koutnik, J., Greff, K., Gomez, F., & Schmidhuber, J. (2014). A clockwork rnn. *arXiv preprint arXiv:1402.3511*.

[10] Greff, K., Srivastava, R. K., Koutník, J., Steunebrink, B. R., & Schmidhuber, J. (2017). LSTM: A search space odyssey. *IEEE transactions on neural networks and learning systems*, *28*(10), 2222-2232.

[11] Leontjeva, A., Conforti, R., Di Francescomarino, C., Dumas, M., & Maggi, F. M. (2015, August). Complex symbolic sequence encodings for predictive monitoring of business processes. In *International Conference on Business Process Management* (pp. 297-313). Springer, Cham.

[12] Damerau, F. J. (1964). A technique for computer detection and correction of spelling errors. *Communications of the ACM*, *7*(3), 171-176.

[13] Burattin, A., Maggi, F. M., & Sperduti, A. (2016). Conformance checking based on multi-perspective declarative process models. *Expert Systems with Applications*, *65*, 194-211.

[14] Murata, T. (1989). Petri nets: Properties, analysis and applications. *Proceedings of the IEEE*, *77*(4), 541-580.

[15] Van der Aalst, W. M. (1998). The application of Petri nets to workflow management. *Journal of circuits, systems, and computers*, *8*(01), 21-66.

[16] Pnueli, A. (1977, October). The temporal logic of programs. In *Foundations of Computer Science, 1977., 18th Annual Symposium on* (pp. 46-57). IEEE.

[17] Petri, C. A(1962)., Kommunikation mit Automaten, Ph.D. thesis, Technische Hochschule Darmstadt

[18] Breuker, D., Matzner, M., Delfmann, P., & Becker, J. (2016). Comprehensible Predictive Models for Business Processes. *MIS Quarterly*, *40*(4), 1009-1034.

[19] Rabin, M. O. (1963). Probabilistic automata. *Information and control*, *6*(3), 230-245.

[20] Evermann, J., Rehse, J. R., & Fettke, P. (2016, September). A deep learning approach for predicting process behaviour at runtime. In *International Conference on Business Process Management* (pp. 327-338). Springer, Cham.

[21] Evermann, J., Rehse, J. R., & Fettke, P. (2017). Predicting process behaviour using deep learning. *Decision Support Systems*, *100*, 129-140.

[22] Geman, S., Bienenstock, E., & Doursat, R. (1992). Neural networks and the bias/variance dilemma. *Neural computation*, *4*(1), 1-58.

[23] Pravilovic, S., Appice, A., & Malerba, D. (2013, September). Process mining to forecast the future of running cases. In *International Workshop on New Frontiers in Mining Complex Patterns* (pp. 67-81). Springer, Cham.

[24] Lakshmanan, G. T., Shamsi, D., Doganata, Y. N., Unuvar, M., & Khalaf, R. (2015). A markov prediction model for data-driven semi-structured business processes. *Knowledge and Information Systems*, *42*(1), 97-126.

[25] Van Der Spoel, S., Van Keulen, M., & Amrit, C. (2012, June). Process prediction in noisy data sets: a case study in a dutch hospital. In *International Symposium on Data-Driven Process Discovery and Analysis* (pp. 60-83). Springer, Berlin, Heidelberg.

[26] Castellanos, M., Salazar, N., Casati, F., Dayal, U., & Shan, M. C. (2005, March). Predictive business operations management. In *International Workshop on Databases in Networked Information Systems* (pp. 1-14). Springer, Berlin, Heidelberg.

[27] Grigori, D., Casati, F., Castellanos, M., Dayal, U., Sayal, M., & Shan, M. C. (2004). Business process intelligence. *Computers in industry*, *53*(3), 321-343.

[28] Grigori, D., Casati, F., Dayal, U., & Shan, M. C. (2001, September). Improving business process quality through exception understanding, prediction, and prevention. In *VLDB*(Vol. 1, pp. 159-168).

[29] Kang, B., Kim, D., & Kang, S. H. (2012). Periodic performance prediction for real-time business process monitoring. *Industrial Management & Data Systems*, *112*(1), 4-23.

[30] Kang, B., Kim, D., & Kang, S. H. (2012). Real-time business process monitoring method for prediction of abnormal termination using KNNI-based LOF prediction. *Expert Systems with Applications*, *39*(5), 6061-6068.

[31] Conforti, R., De Leoni, M., La Rosa, M., & Van Der Aalst, W. M. (2013, June). Supporting risk-informed decisions during business process execution. In *International Conference on Advanced Information Systems Engineering* (pp. 116-132). Springer, Berlin, Heidelberg.

[32] van Dongen, B. F., Crooy, R. A., & van der Aalst, W. M. (2008, November). Cycle time prediction: When will this case finally be finished?. In *OTM Confederated International Conferences" On the Move to Meaningful Internet Systems"*(pp. 319-336). Springer, Berlin, Heidelberg.

[33] Pandey, S., Nepal, S., & Chen, S. (2011, October). A test-bed for the evaluation of business process prediction techniques. In *Collaborative Computing: Networking, Applications and Worksharing (CollaborateCom), 2011 7th International Conference on* (pp. 382-391). IEEE.

[34] Van der Aalst, W. M., Schonenberg, M. H., & Song, M. (2011). Time prediction based on process mining. *Information systems*, *36*(2), 450-475.

[35] Folino, F., Guarascio, M., & Pontieri, L. (2012, September). Context-aware predictions on business processes: an ensemble-based solution. In *International Workshop on New Frontiers in Mining Complex Patterns* (pp. 215-229). Springer, Berlin, Heidelberg.

[36] Folino, F., Guarascio, M., & Pontieri, L. (2012, September). Discovering context-aware models for predicting business process performances. In *OTM Confederated International Conferences" On the Move to Meaningful Internet Systems"*(pp. 287-304). Springer, Berlin, Heidelberg.

[37] Le, M., Gabrys, B., & Nauck, D. (2012). A hybrid model for business process event prediction. In *Research and Development in Intelligent Systems XXIX* (pp. 179-192). Springer, London.

[38] Schwegmann, B., Matzner, M., & Janiesch, C. (2013, June). preCEP: facilitating predictive event-driven process analytics. In *International Conference on Design Science Research in Information Systems* (pp. 448-455). Springer, Berlin, Heidelberg.

[39] Rogge-Solti, A., & Weske, M. (2013, December). Prediction of remaining service execution time using stochastic petri nets with arbitrary firing delays. In *International Conference on Service-Oriented Computing* (pp. 389-403). Springer, Berlin, Heidelberg.

[40] Rogge-Solti, A., & Weske, M. (2015). Prediction of business process durations using non-Markovian stochastic Petri nets. *Information Systems*, *54*, 1-14.

[41] Bevacqua, A., Carnuccio, M., Folino, F., Guarascio, M., & Pontieri, L. (2013, July). A data-driven prediction framework for analyzing and monitoring business process performances. In *International Conference on Enterprise Information Systems*(pp. 100-117). Springer, Cham.

[42] Bolt, A., & Sepúlveda, M. (2013, August). Process remaining time prediction using query catalogs. In *International Conference on Business Process Management* (pp. 54-65). Springer, Cham.

[43] Polato, M., Sperduti, A., Burattin, A., & de Leoni, M. (2014, July). Data-aware remaining time prediction of business process instances. In *Neural Networks (IJCNN), 2014 International Joint Conference on* (pp. 816-823). IEEE.

[44] Polato, M., Sperduti, A., Burattin, A., & de Leoni, M. (2016). Time and activity sequence prediction of business process instances. *arXiv preprint arXiv:1602.07566*.

[45] Maggi, F. M., Di Francescomarino, C., Dumas, M., & Ghidini, C. (2014, June). Predictive monitoring of business processes. In *International Conference on Advanced Information Systems Engineering* (pp. 457-472). Springer, Cham.

[46] Chomicki, J. (1995). Efficient checking of temporal integrity constraints using bounded history encoding. *ACM Transactions on Database Systems (TODS), 20*(2), 149-186.

[47] Dumas, M., La Rosa, M., Mendling, J., & Reijers, H. A. (2013). *Fundamentals of business process management* (Vol. 1, p. 2). Heidelberg: Springer.

[48] Di Francescomarino, C., Dumas, M., Federici, M., Ghidini, C., Maggi, F. M., & Rizzi, W. (2016, June). Predictive business process monitoring framework with hyperparameter optimization. In *International Conference on Advanced Information Systems Engineering* (pp. 361-376). Springer, Cham.

[49] Folino, F., Guarascio, M., & Pontieri, L. (2015, September). A prediction framework for proactively monitoring aggregate process-performance indicators. In *Enterprise Distributed Object Computing Conference (EDOC), 2015 IEEE 19th International* (pp. 128-133). IEEE.

[50] Ceci, M., Lanotte, P. F., Fumarola, F., Cavallo, D. P., & Malerba, D. (2014, October). Completion time and next activity prediction of processes using sequential pattern mining. In *International Conference on Discovery Science* (pp. 49-61). Springer, Cham.

[51] Unuvar, M., Lakshmanan, G. T., & Doganata, Y. N. (2016). Leveraging path information to generate predictions for parallel business processes. *Knowledge and Information Systems*, *47*(2), 433-461.

[52] van Der Aalst, W. M., Pesic, M., & Schonenberg, H. (2009). Declarative workflows: Balancing between flexibility and support. *Computer Science-Research and Development*, *23*(2), 99-113.

I. License

Non-exclusive licence to reproduce thesis and make thesis public

I, **Kaur Järvpõld**,

1. herewith grant the University of Tartu a free permit (non-exclusive licence) to:

   1.1. reproduce, for the purpose of preservation and making available to the public, including for addition to the DSpace digital archives until expiry of the term of validity of the copyright, and

   1.2. make available to the public via the web environment of the University of Tartu, including via the DSpace digital archives until expiry of the term of validity of the copyright,

of my thesis

Leveraging Multi-Perspective A-priori Knowledge in Predictive Business Process Monitoring,

supervised by Fabrizio Maria Maggi, Chiara Di Francescomarino and Chiara Ghidini

2. I am aware of the fact that the author retains these rights.

3. I certify that granting the non-exclusive licence does not infringe the intellectual property rights or rights arising from the Personal Data Protection Act.

Tartu, **24.05.2018**