Deep Learning Process Prediction with Discrete and Continuous Data Features

Stefan Schönig, Richard Jasinski, Lars Ackermann and Stefan Jablonski
Department of Computer Science, University of Bayreuth, Germany

Keywords: Process Prediction System, Neural Network, Deep Learning, Multi Perspective, Process Event Log.

Abstract: Process prediction is a well known method to support participants in performing business processes. These methods use event logs of executed cases as a knowledge base to make predictions for running instances. A range of such techniques have been proposed for different tasks, e.g., for predicting the next activity or the remaining time of a running instance. Neural networks with Long Short-Term Memory architectures have turned out to be highly customizable and precise in predicting the next activity in a running case. Current research, however, focuses on the prediction of future activities using activity labels and resource information while further event log information, in particular discrete and continuous event data is neglected. In this paper, we show how prediction accuracy can significantly be improved by incorporating event data attributes. We regard this extension of conventional algorithms as a substantial contribution to the field of activity prediction. The new approach has been validated with a recent real-life event log.

1 INTRODUCTION

Process prediction methods support process participants in performing running instances of a business process, e.g., software development processes (Van der Aalst et al., 2011). Therefore, these techniques use given event logs of the considered process, i.e., historical information of completed software development processes, as a knowledge base to make predictions about the evolution of running instances. An event log consists of traces, such that each trace corresponds to one execution of the process. Each event in a trace consists as a minimum of an event class (i.e., the activity to which the event corresponds) and generally a timestamp. In some cases, other information may be available such as the originator of the event (i.e., the performer of the activity) as well as data produced by the event in the form of attribute-value pairs (Schönig et al., 2016).

Several prediction techniques have been described in literature that focus on different tasks, e.g., predicting the next activity to be executed (Becker et al., 2014) or predicting the remaining runtime of the corresponding process instance (Polato et al., 2016). The results of process prediction approaches can be used to recommend next process steps to users or to support planning and resource allocation.

Contemporary process prediction approaches are tailored to specific prediction tasks and are not generally applicable. Furthermore, their prediction quality varies significantly depending on the knowledge base, i.e., the used input event log. As a result, a certain technique may produce good prediction results for one specific process but not for another one. In many cases, several techniques need to be used in parallel to ensure sufficient prediction quality (Metzger et al., 2015).

Latest research has shown that neural networks with Long Short-Term Memory (LSTM) architectures (Hochreiter and Schmidhuber, 1997) are highly customizable and precise in predicting the next activity in a running case. Here, LSTMs deliver highly accurate prediction results in a variety of different application. For example, the work in (Evermann et al., 2017) applied LSTMs to predict the next activity in a running instance. In (Tax et al., 2016) it is shown that LSTMs also achieve consistent and high quality results when being applied to further prediction problems like time-related properties, i.e., timestamp prediction of activities and the remaining instance runtime.

Current research, however, focuses on the prediction of future activities using activity labels and resource information while further event log information, in particular additional discrete and continuous *event data* is neglected. To the best of our knowl-

Schönig, S., Jasinski, R., Ackermann, L. and Jablonski, S.
Deep Learning Process Prediction with Discrete and Continuous Data Features.
DOI: 10.5220/0006772003140319
In *Proceedings of the 13th International Conference on Evaluation of Novel Approaches to Software Engineering (ENASE 2018)*, pages 314-319
ISBN: 978-989-758-300-1

edge there is currently no research work that examines the impact and effects of discrete and continuous event log data on prediction quality. In this paper, we show how prediction accuracy can be significantly improved by incorporating additional event data attributes in LSTM based process prediction. Therefore, our contribution forms the building blocks for a comprehensive multi perspective process prediction method. It substantially improves conventional activity prediction approaches by opening them to multi perspectives bearing valuable information for process prediction. The new approach has been validated with a recent real-life event log.

This paper is structured as follows: Section 2 describes fundamentals of neural networks. Section 3 gives an overview of related work. Section 4 introduces our approach to multi-perspective process prediction. Section 5 briefly describes how we implemented the technique. In Section 6 we describe the evaluation of our approach in detail and the paper is concluded in Section 7.

2 BACKGROUND

In this section, we introduce the foundations of our approach. First, we introduce process event logs. Then, we briefly introduce neural networks and in particular long short-term architectures.

2.1 Event Logs for Multi-perspective Process Recommendation

Our prediction approach takes as input a process *event log*, i.e., a machine-recorded file that reports on the execution of tasks during the enactment of the instances of a given process. In an event log, every process instance corresponds to a sequence (*trace*) of recorded entries, namely, *events*. We require that events contain an explicit reference to the enacted task. This condition is commonly respected in real-world event logs (van der Aalst, 2011). For instance, the following excerpt of a business trip process event log encoded in the XES logging format (Verbeek et al., 2011a) shows the recorded information of the *start event* of activity *Apply for trip* performed by resource *ST*.

```
<event>
<string key="org:resource" value="ST"/>
<date key="time:timestamp" value="2013-08-06T14:58:00"/>
<string key="concept:name" value="Apply for trip"/>
<string key="lifecycle:transition" value="start"/>
</event>
```

2.2 Artificial Neural Networks

Artificial neural networks are gaining more and more importance in many applications, especially through advancements in the field of deep learning (Schmidhuber, 2015). A neural network consists of three types of layers: one input, multiple hidden and one output layer. These layers consist of neurons which are connected to neurons of the predecessor layer by trainable weights. Neurons of the hidden and output layer can be described by input, activation and output functions. In order to accumulate the output o_i of the predecessor layer neurons i with their weights w_{ij}, the input function of a neuron j is a weighted sum function:

$$net_i = \sum_i w_{ij} \cdot o_i \qquad (1)$$

Activation and output function can be chosen freely. In the case of the hidden layer, we use long short-term memory units that are described in Section 2.3 of the work at hand. The number of neurons in the input and output layers is limited by the input and output dimension of the neural network, respectively. The number of hidden layers and hidden layer neurons can be chosen independently to tune the neural network performance for a given application.

2.3 Long Short-term Memory Units

In this paper, we build on neural network architecture that uses so called long short-term memory (LSTM) units. LSTM networks have been introduced in (Hochreiter and Schmidhuber, 1997) and have proven to be powerful in learning long term dependencies, e.g., for speech recognition (Graves and Schmidhuber, 2005). Here, a unit can be described by following equations:

$$\begin{aligned}
i_t &= \sigma(W_{xi}x_t + W_{hi}h_{t-1} + W_{ci}c_{t-1} + b_i) && (2)\\
f_t &= \sigma(W_{xf}x_t + W_{hf}h_{t-1} + W_{cf}c_{t-1} + b_f) && (3)\\
c_t &= f_t c_{t-1} + i_t \tanh(W_{xc}x_t + W_{hc}h_{t-1} + b_c) && (4)\\
o_t &= \sigma(W_{xo}x_t + W_{ho}h_{t-1} + W_{co}c_t + b_o) && (5)\\
h_t &= o_t \tanh(c_t) && (6)
\end{aligned}$$

Through the cell state c_t it is possible to use contextual information for the prediction at the time t. This cell state is regulated by an input gate i_t, a forget gate f_t, and the current input x_t. h_t is the output of the unit, o_t is the output gate, σ is a sigmoid function chosen by the user. W are the weight matrices for the corresponding gates or vectors, e.g. W_{hf} the hidden-forget weight matrix. During training these weight matrices are optimized. Since the output of a unit depends on the past output, this neural network architecture is called recurrent neural network. This allows us to predict a next event by taking several past events into account.

3 RELATED WORK

In this section, we give a short overview of related approaches to multi-perspective event log analysis and process prediction. Process event logs have been used for different purposes such as dynamic guidance features in process management systems (Günther et al., 2012), multi-perspective process mining (Sturm et al., 2017) or translation of process models from language to another one (Ackermann et al., 2016). Several approaches have been proposed that focus on time-related aspects of process prediction. The work in (Pika et al., 2012) describes a technique to predict deadline violations, (Metzger et al., 2015) focuses on predicting delays of activities and (Senderovich et al., 2014) predicts delays of instance executions by applying queue mining techniques. Furthermore, there are approaches that predict the remaining cycle time of running instances (Van der Aalst et al., 2011; van Dongen et al., 2008). Another branch of approaches focuses on the prediction of case outcomes, i.e., normal or deviant, based on the sequence of activities that have been executed (Maggi et al., 2014; Leontjeva et al., 2015). In order to predict the next activity to be executed several approaches have been proposed as well. While the authors in (Breuker et al., 2016) use probabilistic automatons for prediction, the work in (Evermann et al., 2017) and (Tax et al., 2016) are based on LSTMs. The LSTM based solutions have turned out to be more precise and more generalizable to further prediction tasks.

None of the mentioned approaches examine the impact and effects of incorporating discrete and continuous additional data attributes on prediction quality and accuracy, i.e. they neglect information available to improve prediction. Therefore, our approach opens a new field of prediction algorithms that has the potential to significantly improving prediction quality. The validation (Section 5) substantiates this thesis.

4 MULTI-PERSPECTIVE PROCESS PREDICTION SYSTEM

In this section, we describe how data of an event log is preprocessed and how the neural network is build dependent on the preprocessed data.

4.1 Data Preprocessing

A neural network is expecting any input to be in a vectorial shape so that each input neuron is responsible for one vector component. Therefore we will consider events and any contextual information as matrices where each row corresponds to one event:

$$\begin{pmatrix} event_1 & resource_1 & \dots & x_1 \\ \vdots & \vdots & \vdots & \vdots \\ event_n & resource_n & \dots & x_n \end{pmatrix} \tag{7}$$

In a most minimal case this will only contain n events of an event log. It is possible to add additional discrete features, e.g., the resource that performed the corresponding event, or continuous attributes (event log features) like credit score data. For creating a suitable training data set several processing steps have to be applied.

First, a column has to be added that represents the expected output. We distinguish between a single perspective output, which is ”only” $event_{t+1}$, and a multi perspective output, which consists of **at least** two features, e.g., the next activity to be performed as well as the performing resource $event_{t+1}\_resource_{t+1}$. Here, we also predict the organizational perspective. Afterwards, this data set is normalized to convert discrete data into numerical data and to be able to compare continuous attributes. In case of continuous attributes, we are using the min-max normalization:

$$y_{norm} = \frac{y_i - y_{min}}{y_{max} - y_{min}} \tag{8}$$

Discrete attributes are normalized with a one-hot encoding. For each value of a discrete feature a new column has to be introduced that takes the value 1 if the discrete feature was present in that row. The newly created matrix for a multi-perspective prediction of the next occurring event e and its resource r in consideration of a discrete or continuous data variable x can now be written as:

$$\begin{pmatrix} e_1 & \dots & e_n & r_1 & \dots & r_n & x & e_2r_2 & \dots \\ 1 & \dots & 0 & 1 & \dots & 0 & x_1 & 1 & \dots \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ 0 & \dots & 1 & 0 & \dots & 1 & x_n & 0 & \dots \end{pmatrix} \tag{9}$$

The first row was kept for better readability and is not present in the data set. With this method, an arbitrary number of discrete or continuous features can be used for predicting the next event under consideration of multiple process perspectives, i.e., resources or data values. This way, the process prediction approach can be adapted to different business processes and the corresponding event log knowledge bases.

4.2 Creating and Training the Neural Network

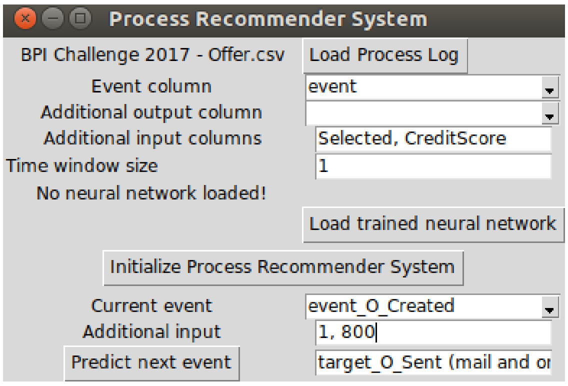
Based on the preprocessed data set, we can define the topology of the neural network. The number of input neurons is equal to the number of columns of the preprocessed data set minus the columns dedicated to the output, e.g., $\mathit{event}_1\_\mathit{resource}_1$. The number of neurons in the output layer is analogous equal to the number of columns in the data set representing a normalized output feature. Two hidden layers of LSTM units are used. Additionally, we choose a *softmax* activation function for the output neurons to model a probabilistic output with a cross-entropy loss function (Bishop, 1995). This logistic function limits the sum of the outputs of the output layer to 1. To avoid overfitting, *dropout* (Srivastava et al., 2014) was applied with a probability of 0.3, which means that in each training step 30% of the hidden neurons with their weights are randomly dropped. For each output feature the neural network will output a probability, so that the feature with highest probability will win and therefore will dictate the next event or in a multi-perspective setting the next event performed by a certain resource etc. This way, it is also possible to display other outputs with their probability to give the user additional feedback. This neural network is trained through *rmsprop* (Tieleman and Hinton, 2012). It is a improved stochastic gradient descent algorithm, which adapts the learning rate.

5 IMPLEMENTATION

To use neural networks on a high abstraction level, we implemented the neural network with the software *Keras* (Chollet, 2015). The user interface is visualized in Fig. 1. We assume that the event logs used as a data source for constructing the neural networks are given in the XES standard format (Verbeek et al., 2011b). After choosing a XES compatible event log it is subsequently transformed to a CSV file for accomplishing the matrix shape as described in Section 4.1.

- Add two columns representing traces and events.
- For each event in the .xes log add a new row and fill the event column with the event name and the trace column with the case name.
- For each additional event key, add a new column and fill it accordingly.

Keras differentiates between stateless and stateful long short-term memory units. In theory, units are always stateful, meaning their internal memory will be maintained over the whole training period. In Keras, the memory will be reset by default. This way a unit can only use the past n predictions for n unfolding steps. We use these stateless units to avoid influences between two traces. A graphical user interface was implemented with *tkinter*. It allows to load a CSV file, select input and output features, and to make process predictions based on the trained network.


Process Recommender System
Load Process Log
Load trained neural network
Initialize Process Recommender System
Predict next event

BPI Challenge 2017 - Offer.csv

Event column: event

Additional output column:

Additional input columns: Selected, CreditScore

Time window size: 1

No neural network loaded!

Current event: event_O_Created

Additional input: 1, 800

target_O_Sent (mail and or

Figure 1: Graphical user interface of the process recommender system.

6 EVALUATION

In this section, we provide an evaluation of our process prediction system approach w.r.t. a real-life event log. We first evaluate the prediction accuracy by considering also discrete and continuous data attributes. Afterwards, we highlight the impact of the number of unrolled steps on the prediction accuracy.

6.1 Prediction Accuracy

We evaluated the application of a LSTM network with different input configurations on a real-life event log (Van Dongen, 2017) since this offers a high variety of additional discrete and continuous data attributes besides the standard event data. The event log pertains to a loan application process of a Dutch financial institute. The data contains all offers made for an accepted application. In total, there are 1,202,267 events pertaining to 31,509 loan applications. For these applications, a total of 42,995 offers were created. The prediction target was for every configuration the next occurring event. We used different input configurations of the discrete input features *selected* and *accepted*, which are describing, whether the offer was selected and accepted by the customer. Furthermore, the continuous features *credit score* and *monthly cost* were considered as well. To evaluate possible overfitting, we applied stratified 5-fold cross-validation. The

Table 1: Prediction accuracy for different input combinations.

Additional Data Attributes	**Training Accuracy**	**Validation Accuracy**
None	0.655	0.642
Credit score *(cont.)*	0.832	0.814
Credit score, monthly cost *(cont.)*	0.832	0.814
Accepted, selected *(disc.)*	0.908	0.886
Credit score, monthly cost, accepted and selected *(both)*	**0.917**	**0.895**

data set was split into 5 subsets with the same proportions of prediction targets. Each set was used as a validation data set, which was not used for training, the remaining sets are used as training data sets. The prediction accuracy is given in Table 1. The accuracy is calculated by dividing the number of correct predictions by the number of all predictions for the current validation set and afterwards averaged for the 5 different sets. Training took at maximum 100 epochs, so 100 iterations over the training data set, to achieve converging loss.

We can report an acceptable level of overfitting with 2%. Biggest impact on prediction accuracy was achieved by selecting *accepted* and *selected* as input features (accuracy increased from 0.642 to 0.886). This can be intuitively explained since these are sufficient conditions for an offer to be refused or canceled. Addition of *credit score* lead to an similiar increasement. *Monthly cost* had no impact on accuracy. To validate the impact of the continuous feature *credit score*, the discrete features *accepted* and *selected* were added as input features and input combinations were compared. Prediction accuracy could be increased by 0.9% in case of *credit score*, *monthly cost*, *accepted* and *selected* in comparison to this combination without using the continuous input parameters.

6.2 Impact of Number of Unrolled Steps

Since the number of unrolled steps is an important parameter as mentioned in Section 5, we examined whether this has impact on prediction accuracy. Traces should be viewed independent from each other. The number of unrolled steps was chosen after the minimum number of events per trace and the maximum number of events per trace. The results are given in Table 2. The work in (Evermann et al., 2017) shows that the number of unrolled steps has little to no impact on prediction accuracy. This is also the case if we add continuous parameters, since these parameters are static for an event trace. Therefore, no long-term dependencies can be verified. One can expect that this will change as soon as parameter are changing during a trace. Results are overall higher, because we skipped cross-validation, since no improvement of accuracy could be detected.

Table 2: Validation accuracy over number of unrolled steps.

Input Parameters	Number of Unrolled Steps		
	5	3	1
None	0.649	0.651	0.650
Credit score	0.870	0.870	0.876
Credit score and monthly cost	0.869	0.871	0.877
Accepted and selected	0.912	0.912	0.919
Credit score, monthly cost, accepted and selected	**0.924**	**0.925**	**0.931**

7 CONCLUSION AND FUTURE WORK

In this paper, we introduced a general approach towards a multi-perspective process prediction system. We used a deep learning architecture based on LSTM units with event logs containing contextual data features. We evaluated the architecture for a real-life event log. It showed to be an effective way to use an arbitrary number of additional input features besides the main event information. The prediction error could be decreased from 35.8% to 10.5% (Table 1) by using additional input features. In this paper, discrete text data and continuous data were discussed. For future work, we also want to consider hybrid neural network architectures that comprise additional input data like image data. This approach could be used for process automation by embedding a neural network into a software agent This way, accurate predictions by using different data sources can be made automatically executed. Other applications were introduced like (Bose et al., 2012) with the same goal, but these depend on explicitly modeling the process.

REFERENCES

Ackermann, L., Schönig, S., and Jablonski, S. (2016). Towards simulation- and mining-based translation of resource-aware process models. In *Business Process Management Workshops - BPM 2016 International Workshops, Rio de Janeiro, Brazil, September 19, 2016, Revised Papers*, pages 359–371.

Becker, J., Breuker, D., Delfmann, P., and Matzner, M. (2014). Designing and implementing a framework for event-based predictive modelling of business processes. In *EMISA*, pages 71–84.

Bishop, C. M. (1995). *Neural networks for pattern recognition*. Oxford university press.

Bose, S., Scimone, S., Sriraman, N., Duan, Z., Bernstein, A., Lewis, P., and Grosu, R. (2012). Business process automation. US Patent 8,332,864.

Breuker, D., Matzner, M., Delfmann, P., and Becker, J. (2016). Comprehensible predictive models for business processes. *MIS Quarterly*, 40(4):1009–1034.

Chollet, F. (2015). Keras. https://github.com/fchollet/keras.

Evermann, J., Rehse, J.-R., and Fettke, P. (2017). Predicting process behaviour using deep learning. *Decision Support Systems*.

Graves, A. and Schmidhuber, J. (2005). Framewise phoneme classification with bidirectional lstm and other neural network architectures. *Neural Networks*, 18(5):602–610.

Günther, C., Schönig, S., and Jablonski, S. (2012). Dynamic guidance enhancement in workflow management systems. In *Proceedings of the ACM Symposium on Applied Computing, SAC 2012, Riva, Trento, Italy, March 26-30, 2012*, pages 1717–1719.

Hochreiter, S. and Schmidhuber, J. (1997). Long short-term memory. *Neural computation*, 9(8):1735–1780.

Leontjeva, A., Conforti, R., Di Francescomarino, C., Dumas, M., and Maggi, F. M. (2015). Complex symbolic sequence encodings for predictive monitoring of business processes. In *BPM*, pages 297–313.

Maggi, F. M., Di Francescomarino, C., Dumas, M., and Ghidini, C. (2014). Predictive monitoring of business processes. In *CAISE*, pages 457–472. Springer.

Metzger, A., Leitner, P., Ivanović, D., Schmieders, E., Franklin, R., Carro, M., Dustdar, S., and Pohl, K. (2015). Comparing and combining predictive business process monitoring techniques. *Transactions on Systems, Man, and Cybernetics: Systems*, 45(2):276–290.

Pika, A., van der Aalst, W. M., Fidge, C. J., ter Hofstede, A. H., and Wynn, M. T. (2012). Predicting deadline tansgressions using event logs. *BPM*.

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

Schmidhuber, J. (2015). Deep learning in neural networks: An overview. *Neural Networks*, 61:85 – 117.

Schönig, S., Di Ciccio, C., Maggi, F. M., and Mendling, J. (2016). Discovery of multi-perspective declarative process models. In *Service-Oriented Computing - 14th International Conference, ICSOC 2016, Banff, AB, Canada, October 10-13, 2016, Proceedings*, pages 87–103.

Senderovich, A., Weidlich, M., Gal, A., and Mandelbaum, A. (2014). Queue mining–predicting delays in service processes. In *CAISE*.

Srivastava, N., Hinton, G. E., Krizhevsky, A., Sutskever, I., and Salakhutdinov, R. (2014). Dropout: a simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research*, 15(1):1929–1958.

Sturm, C., Schönig, S., and Di Ciccio, C. (2017). Distributed multi-perspective declare discovery. In *Proceedings of the BPM Demo Track and BPM Dissertation Award co-located with 15th International Conference on Business Process Modeling (BPM 2017), Barcelona, Spain, September 13, 2017*.

Tax, N., Verenich, I., La Rosa, M., and Dumas, M. (2016). Predictive business process monitoring with lstm neural networks. *arXiv preprint arXiv:1612.02130*.

Tieleman, T. and Hinton, G. (2012). Lecture 6.5-rmsprop: Divide the gradient by a running average of its recent magnitude. *COURSERA: Neural networks for machine learning*, 4(2).

van der Aalst, W. (2011). *Process mining: discovery, conformance and enhancement of business processes*. Springer-Verlag Berlin Heidelberg.

Van der Aalst, W. M., Schonenberg, M. H., and Song, M. (2011). Time prediction based on process mining. *Information Systems*, 36(2):450–475.

Van Dongen, B. (2017). Bpi challenge 2017 - offer log.

van Dongen, B., Crooy, R., and Van der Aalst, W. (2008). Cycle time prediction: When will this case finally be finished? *CoopIS*, pages 319–336.

Verbeek, E., Buijs, J., van Dongen, B., and van der Aalst, W. (2011a). XES, xESame, and ProM 6. In *Information Systems Evolution*, volume 72, pages 60–75.

Verbeek, E., Buijs, J., van Dongen, B., and van der Aalst, W. (2011b). XES, xESame, and ProM 6. In *Information Systems Evolution*, pages 60–75.