A Deep Learning Approach for Predicting Process Behaviour at Runtime

Joerg Evermann[1], Jana-Rebecca Rehse[2,3], and Peter Fettke[2,3]

[1] Memorial University of Newfoundland
[2] German Research Center for Artificial Intelligence
[3] Saarland University

Abstract. Predicting the final state of a running process, the remaining time to completion or the next activity of a running process are important aspects of runtime process management. Runtime management requires the ability to identify processes that are at risk of not meeting certain criteria in order to offer case managers decision information for timely intervention. This in turn requires accurate prediction models for process outcomes and for the next process event, based on runtime information available at the prediction and decision point. In this paper, we describe an initial application of deep learning with recurrent neural networks to the problem of predicting the next process event. This is both a novel method in process prediction, which has previously relied on explicit process models in the form of Hidden Markov Models (HMM) or annotated transition systems, and also a novel application for deep learning methods.

Key words: Process management, Runtime support, Process prediction, Deep learning, Neural networks

1 Introduction

Managing processes at runtime has important business applications [1]. It allows customer service agents to respond to enquiries about the remaining time until a case is resolved or completed. It allows case managers to identify cases that are likely to be late or to terminate abnormally and to intervene early in order to mitigate business risk. Process prediction is an important aspect of runtime process management. In general, prediction concerns either the process outcome or the subsequent event(s) in a process. Examples of business relevant process outcomes include the final state (e.g. whether the final state is "accept client claim" or "reject client claim"), case data (e.g. whether the attribute "cost" is less than a certain amount) or LTL compliance formulas (e.g. whether "approve claim" has occurred prior to "issue cheque" and the activities have been performed by different resources). Predictions can be made from the sequence of the activities that have occurred in the running process, the case data that has been collected, the participating resources in the case activities, the execution times of those activities and any other available case or workflow data that is available at runtime.

Previous work on process prediction at runtime has focused on predicting process outcomes and primarily the remaining time to completion, whereas there exists limited work on predicting the next process event. Most prior work is based on an explicit representation of the process, e.g. mined from event logs, and augmented with probability tables and execution time information. In contrast, our approach does not rely on an explicit representation of the underlying process model, but is based on recent work in "deep learning". While "deep learning" has only recently become a popular research topic, it is essentially an application of neural networks and thus looks back on a long history of research [2]. Recent innovations both in algorithms, allowing novel architectures of neural networks, and computing hardware, especially access to GPU processing, have led to a resurgence in interest for neural networks and popularized the term "deep learning" [3].

This work is motivated by the application of deep learning to natural language process (NLP). Our idea is to treat an event trace as analogous to a natural language sentence, with the events analogous to words. In recent years, NLP research has moved from explicit representations of language models to statistical methods, specifically to recurrent neural networks (RNN) [4, 5, 6]. In this work, we apply a recurrent neural network to the problem of predicting the next event in a process from a sequence of observed events.

The aim of this paper is to explore the potential for applications of deep learning in business process management at runtime and to describe an initial application. Specifically, we are interested in whether methods that are not based on an explicit process model, such as neural networks where the process structure is only implicitly reflected in the model parameters, are as predictive as methods that are based on an explicit model, such as HMM. We emphasize that this is an initial exploration of the feasibility of this approach, and is intended more to open this line of inquiry than to provide final answers. *The contribution therefore lies not in the performance of this particular implementation but in the demonstration of the applicability of our approach and the potential for future work using deep learning in process management.*

The remainder of the paper is structured as follows. Sec. 2 presents related work on process prediction, especially prediction of next event in a process. Sec. 3 presents a brief introduction to deep learning. Sec. 4 describes our implementation of process prediction with RNNs, followed in Sec. 5 by an experimental evaluation. The paper closes with a discussion and outlook to future work in Sec. 6.

2 Related Work

Table 1 shows prior work in the area of process prediction for running cases. Most of the prediction methods address process outcomes rather than prediction of the next event in a process, as we do here. The most frequently examined outcome is the time remaining to completion of a case. Only five approaches are concerned with predicting the next event [7, 8, 9, 10, 11], many of which use an explicit

process model representation such as HMM (Hidden Markov Models) and PFA (Probabilistic Finite Automatons).

The MSA approach in [7] considers each trace prefix as a state in a state-transition matrix. From the observed prefixes and their next tasks, a state transition matrix is built. When a running case has reached a state not contained in the state-transition matrix, its similarity to observed traces is computed using string edit distance. The prediction is made from the most similar observed case.

The approach described by [8, 9] consists of five steps. A process model is mined from existing logs. For each XOR split in the model, a decision tree is mined from case data. These trees are then used to compute the state transition probabilities for a Markov chain specific to the running case that is to be predicted, from the case data available at that point. This HMM is then used to predict the probabilities of the following events.

The approach described in [10] uses sequence mining to identify frequent trace prefixes. For each prefix, a regression model is trained to predict remaining time to completion and a classification model (decision trees) is trained to predict the next event. The algorithm identifies the appropriate prefix of the running case to chose the regression and classification model and uses these to predict remaining completion time and next event.

RegPFA [11] is also based on explicit process models but uses a probabilistic finite automaton (PFA) instead of an HMM because it allows the future hidden state to be a function of both the previous hidden state and the previous observed event (which itself is a probabilistic consequence of the previous hidden state). RegPFA uses an EM algorithm to estimate the model parameters of the PFA, similar to the Baum-Welch algorithm used for HMM.

3 Deep Learning

A neural network consists of a layer of input cells, multiple layers of "hidden" cells, and a layer of output cells. Cells in each layer are connected by weighted connections to cells in previous and following layers in various forms, allowing for different architectures (e.g. each cell connected to all others on the following layers, or other topologies). Each cell's output is a function of the weighted sum of its input. A typical network architecture is a fully connected network of cells using sigmoid activation functions:

$$a_j^l = \sigma\left(\sum_i w_i^{l-1,j} a_i^{l-1} + b_j^l\right) \qquad \text{where} \qquad \sigma(x) = \frac{1}{1 + exp(x)}$$

Here, a_j^l is the output ("activation") of cell j in layer l, $w_j^{l,i}$ is the weight of the connection from cell i on layer $l-1$ to cell j on layer l, a_i^{l-1} is the output of cell i on layer $l-1$ and b_j^l is the "bias" of cell j on layer l.

Year	*Predictand(s)*	*Predictor(s)*	*Method*	*Evaluation Method*	*Ref*
2001	Binary outcome	Time, resource, Case data	Decision trees	(1)	[12, 13, 14]
2008	Remaining time to completion	Event frequencies, event times, case data	Boosted regression	MSE	[15]
2011	Remaining time to completion	Event sequence and event execution times	Annotated transition system, HMM	(1)	[16]
2011	Binary outcome (process failure)	Case data	SVM	Sensitivity, specificity	[17]
2011	Remaining time to completion	Event sequence, execution times, case data	Annotated transition systems	RMSE	[18]
2012	Remaining time to completion	Event sequence and case data	Clustering tree and FSM	Accuracy, RMSE, MAE, MAPE	[19, 20]
2012	Binary outcome (process failure)	Case data	Clustering and local outlier detection	Precision, Recall	[21]
2012	Next event	Event sequence	HMM with sequence alignment	Accuracy	[7]
2013	Binary outcome (process failure)	Event sequence, resources, case data	Decision trees	(1)	[22]
2013	Remaining time to completion	Event sequence	?	MSE, Accuracy	[23]
2013	Remaining time to completion	Event sequence and execution times	Stochastic petri nets simulation	RMSE	[24, 25]
2013	Remaining time to completion	Case data	Clustering, regression	RMSE, MAE, MAPE	[26]
2013	Remaining time to completion	Event sequence and execution times	Trace similarity clustering	MAE, MAPE, RMSE	[27]

continued on next page

Year	*Predictand(s)*	*Predictor(s)*	*Method*	*Evaluation Method*	*Ref*
2014	Remaining time to completion	Event sequence and case data	Annotated transition system, support vector regression, naive Bayes classifier	MAPE, RMSE, RMSPE	[28, 29]
2014	Binary outcome (LTL violation)	Case data	Decision trees	Sensitivity, specificity	[30, 31]
2014	Next event, remaining time to completion	Event sequence and case data	Regression, decision trees	Accuracy, MAPE, RMSE	[10]
2014	Binary outcome (LTL violation)	Event sequence and case data	Random forests	Sensitivity, specificity, Area-Under-ROC Curve	[32]
2015	Binary outcome (completion past promise)	Execution times at checkpoints in process	NN, CSP, QoS	Sensitivity, specificity	[33]
2015	Binary outcome (completion past promise)	Case data	Clustering, regression	Sensitivity, specificity	[34]
2015	Next event	Event sequence and case data	HMM and Decision trees	(1)	[8, 9]
2016	Binary outcome (LTL violation)	Event sequence	Decision trees, random forests	Sensitivity, specificity	[35]
2016	Next event	Event sequence	Probabilistic Finite Automaton (PFA)	Cross entropy, Accuracy	[11]

Table 1: Prior work in business process prediction. Note 1: Not compared against ground truth values. MSE=Mean squared error, MAE=Mean absolute error, MAPE=Mean absolute percentage error, RMSE=Root mean square error, RMSPE=Root mean square percentage error

A neural network is a supervised learning technique where the output of the neural net is compared to a target by means of a loss function. Gradients of network parameters $(w_j^{l,i}, b_i^l)$ with respect to the loss function are computed using backpropagation and parameters are then adjusted using variants of gradient descent algorithms to minimize the loss function.

Recurrent Neural Networks (RNN) In a recurrent neural network, each cell also feeds back information into itself, allowing it to maintain "state" over time. In order to make this tractable within an acyclic computational graph and backpropagation, the recurrent network cells are "unrolled", that is, copies of it are produced for time $t, t-1, t-2, \ldots$. The state output of the RNN cell of time $t-1$ is state input to the cell for time t. In general, t can index any sequence, not only time. Depending on how long one wishes to maintain state, fewer or more cells are unrolled. Figure 1 shows an RNN architecture with an input layer, an output layer and two hidden layers that are unrolled five steps. Each layer (each box in Fig. 1) in turn consists of multiple input, output or hidden cells (not shown).

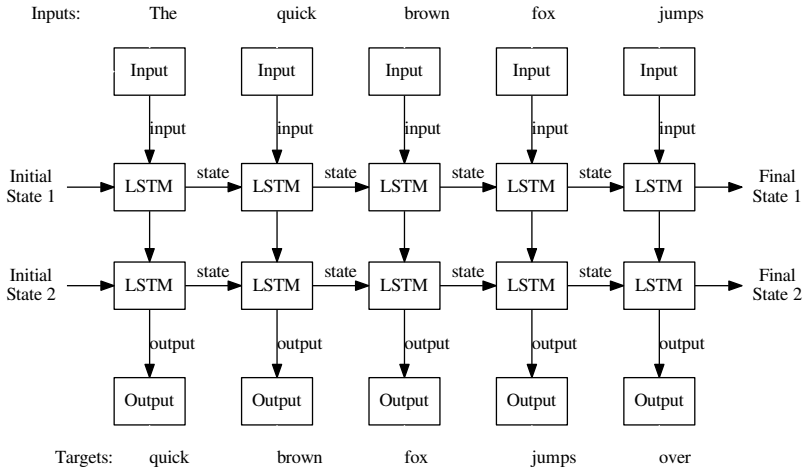

Fig. 1. RNN architecture with single hidden layer of LSTM cells, unrolled five steps

Long Short Term Memory (LSTM) RNN using sigmoid cells have been found to be unsatisfactory, leading to the development of long short term memory (LSTM) cells [36]. A basic LSTM cell is defined as follows, accepting C_{t-1} and h_{t-1} as state information from the prior unrolled cell on the same level, and accepting x_t as input from cells on the previous layers. In turn, it passes C_t and h_t as sate information to the subsequent unrolled cell and provides h_t as output to the next layer; the various W and b are "trainable" parameters.

$$f_t = \sigma\left(W_f \cdot [h_{t-1}, x_t] + b_f\right) \qquad i_t = \sigma\left(W_i \cdot [h_{t-1}, x_t] + b_i\right)$$
$$\bar{C}_t = \tanh\left(W_C \cdot [h_{t-1}, x_t] + b_C\right) \qquad C_t = f_t \times C_{t-1} + i_t \times \bar{C}_t$$
$$o_t = \sigma\left(W_o \cdot [h_{t-1}, x_t] + b_o\right) \qquad h_t = o_t \times \tanh(C_t)$$

Output Layers and Loss Functions The type of output layer cell and the loss function are often chosen jointly for their computational properties with respect to backpropagation. A typical combination is a softmax layer with a cross-entropy loss function:

$$y_i = softmax(x)_i = \frac{exp(x_i)}{\sum_j exp(x_j)} \qquad H_{y\prime}(y) = -\sum_i y_i \prime log(y_i)$$

Here, $y\prime$ are the target values and y are the network outputs, computed in turn from the outputs x_i of the next to last network layer.

NLP Applications of RNN A typical NLP application trains the RNN on sequences of input words to predict the next word, e.g. to provide word suggestions for user input. As shown in Fig 1, the target words are simply the input words shifted by one position, so that for each input word the following word is the target to be predicted. Words are mapped into an n-dimensional "embedding" space using an "embedding matrix", which is essentially a look-up matrix of dimensions $v \times m$ where v is the size of the "vocabulary" and m is the chosen dimensionality of the embedding space and the size of each LSTM hidden layer. While the dimensionality of the space can be chosen freely, larger dimensions allow better separation of words in that space, at the cost of computational performance. The input layer in Fig. 1 is an embedding lookup function that performs the embedding lookup of each input word. The output layer in Fig. 1 is typically a softmax layer that produces a probability over the vocabulary. Training performance is usually defined in terms of the per-word perplexity, defined as $P = \exp(H/n)$. Perplexity measures the "surprisedness" the network exhibits when encountering the next term. Thus, a network that predicts well, will show low perplexity.

4 Process Prediction using RNN

A number of software frameworks for deep-learning, such as Caffe, Torch, Singa, and Tensorflow, have become available recently[1]. We implemented our approach using Tensorflow as it provides a suitable level of abstraction, provides RNN specific functionality, and can be used on high-performance parallel, cluster, and GPU computing platform.

The network architecture features two hidden RNN layers, unrolled to 20 steps, using basic LSTM cells. We chose $m = 500$ for the dimensionality of the

[1] `http://caffe.berkelyvision.org`, `http://torch.ch`, `https://singa.incubator.apache.org`, `https://www.tensorflow.org`

embedding space and the size of the hidden layers. Thus, our network contained a total of $500 \times 20 \times 2 = 20000$ LSTM cells.

Trainable parameters are initialized using a uniform random distribution over $[-0.1, 0.1]$. Training proceeds in batches of size 20. For each batch, the backpropagation algorithm computes the mean gradients for all parameters. Training of the net proceeds is done in "epochs". Each epoch trains the net on the entire event log. Subsequent epochs maintain the weights W and biases b learned from the previous epoch but reinitialize the states for each layer and then train the net again on the entire event log. The learning rate is exponentially reduced from 1 by a factor of 0.75 after the 25th epoch. The net was trained for a maximum of 50 epochs or until maximum accuracy was reached. Because of the random initialization of parameters, we performed three runs for each dataset and report the mean results of the three runs (omitting clear outliers).

We ran our initial experiments on a single NVidia K1100M GPU. Training performance was approximately 1000 words per second. Code, data and results are available from the first author[2].

5 Experimental Results

Because only one of the related works discussed in Sec. 2 uses publicly available data, and to aid comparison of our approach to related work, we chose the same BPI Challenge 2012 and 2013 datasets that [11] used for their study. In addition to separating the BPI 2012 data set by sub-process as done in [11], we also used the combined dataset. While [11] use only activity completion events, we also tested our approach on all events (including the lifecycle transitions "start", "scheduled" and "completed"). Furthermore, we included an experimental condition where we not only extracted the event name, but combined this with the resource name (or "none" if not available). This creates a larger vocabulary which increases the prediction difficulty, but also provides more information to the training algorithm and allow prediction of next event and resource in a process. Because the number of distinct resources in the BPI 2013 Challenge dataset is very large, we combined the organizational group associated with each event, instead of the resource, with the event name.

Table 2 shows our results and a comparison to the best result presented by [11], where available. While [11] report a cross-entropy H in addition to accuracy, their definition of H in their Figure 8 appears to be the entropy, not the cross-entropy, and is therefore not comparable to the cross-entropy typically used in deep learning applications. Finally, we report the perplexity as a standard way of evaluating RNNs in the NLP context. High accuracy values close to 1 are preferable; low perplexity values close to 1 are preferable.

Comparing our initial results to those of [11] shows that they are close to the state-of-the-art on many datasets, significantly lagging only on the BPIC 2013 Problems dataset. Given that this is an initial application and evaluation,

[2] `http://joerg.evermann.ca/software.html`

Dataset	Precision in [11]	Precision	Perplexity
BPI2012.W (complete events)	.719	.623	3.128
BPI2012.A (complete events)	.801	.778	1.649
BPI2012.O (complete events)	.811	.789	1.624
BPI2012.W (complete events, resource)		.836	1.733
BPI2012.A (complete events, resource)		.941	1.226
BPI2012.O (complete events, resource)		.992	1.040
BPI2012.W (all events)		.840	1.531
BPI2012.A (all events)		.775	1.673
BPI2012.O (all events)		.793	1.591
BPI2012.W (all events, resource)		.820	1.745
BPI2012.A (all events, resource)		.941	1.225
BPI2012.O (all events, resource)		.992	1.040
BPI2012 (all events)		.852	1.462
BPI2012 (complete events)		.768	1.822
BPI2012 (all events, resource)		.724	2.368
BPI2012 (complete events, resource)		.802	1.966
BPI2013.Incidents	.714	.699	2.346
BPI2013.Problems	.690	.451	6.151
BPI2013.Incidents (with group)		.939	1.236
BPI2013.Problems (with group)		.954	1.174

Table 2. Results and comparison to [11]. Results are means of three runs.

these results are encouraging. Table 2 also show many results with accuracies in excess of 90%. While we have no comparison to the state-of-the-art on these datasets available in [11], this level of precision is encouraging for practical applications. Comparing the performance of including resource or organizational groups, which dramatically increases the size of the vocabulary, shows that the predictive accuracy improves in all cases. More importantly, the improved accuracy is higher than the best results reported by [11] for the corresponding datasets without resource or group information.

6 Discussion and Conclusion

This paper presents a novel approach to predicting the behaviour of running processes. Using analogies to natural language processing, we applied deep learning, specifically recurrent networks with LSTM cells, to the problem of predicting the next event in a running process. Our results, close to the state-of-the-art on two real datasets and with accuracies in excess of 90% on many problems, demonstrate the feasibility of this approach and should encourage further work in this direction.

As this research is early work with the deep learning approach, we recognize the limitations of this study and the need for further work. Our immediate plans are to explore different network architectures and the parameter space. For example, more advanced LSTM cells are available [37], one can introduce

additional RNN layers (currently 2), one can adjust the sequence of "unrolled" RNN cells (currently 20, which is shorter than the mean trace length for some datasets), one can adjust the dimension of the space into which terms are embedded (currently 500, which is larger than the vocabulary of all datasets), one can adjust the learning rate to be more or less "aggressive", one can adjust the clipping of gradients to allow faster, but possibly sub-optimal, convergence, and one can adjust the random initialization of network parameters. While we believe that the results we have presented were achieved using typical parameters, more work is clearly required to identify optimal architectures and sets of parameter values for the datasets considered in this work. Furthermore, additional replications and cross-validation is required.

Another area of inquiry is to add additional information into the predictors and/or the predictands. In our experiments, we have added resource information to both the predictors as well as the predictands, allowing us to predict also the next resource (as well as the next event). However, case attribute information could be added to each predictor but not the predictands, increasing the information available for prediction but not the number of possible prediction targets, and may therefore lead to better prediction accuracy.

Finally, the deep learning approach can be applied to the prediction of process outcomes. Process outcomes, such as remaining time to completion or violation of an LTL compliance expression, are continuous or categorical, but not in the form of sequences. Both are suitable for neural networks, but do not require the recurrent neural network architecture used here and more "traditional" architectures need to be explored and evaluated.

References

1. Houy, C., Fettke, P., Loos, P., van der Aalst, W.M.P., Krogstie, J.: Bpm-in-the-large - towards a higher level of abstraction in business process management. In: Joint IFIP TC 8 and TC 6 International Conferences, EGES 2010 and GISP 2010, Held as Part of WCC 2010, Brisbane, Australia, September 20-23, 2010. Proceedings. (2010) 233–244
2. Schmidhuber, J.: Deep learning in neural networks: An overview. Neural Networks **61** (2015) 85–117
3. LeCun, Y., Bengio, Y., Hinton, G.: Deep learning. Nature **521** (2015) 436–444
4. Sutskever, I., Martens, J., Hinton, G.E.: Generating text with recurrent neural networks. In: Proceedings of the 28th International Conference on Machine Learning, ICML 2011, Bellevue, Washington, USA, June 28 - July 2, 2011. (2011) 1017–1024
5. Graves, A.: Generating sequences with recurrent neural networks. CoRR **abs/1308.0850** (2013)
6. Zaremba, W., Sutskever, I., Vinyals, O.: Recurrent neural network regularization. CoRR **abs/1409.2329** (2014)
7. Le, M., Gabrys, B., Nauck, D.: A hybrid model for business process event prediction. In: Proceedings of AI-2012, The Thirty-second SGAI International Conference on Innovative Techniques and Applications of Artificial Intelligence, Cambridge, England, UK, December 11-13, 2012. (2012) 179–192

8. Lakshmanan, G.T., Shamsi, D., Doganata, Y.N., Unuvar, M., Khalaf, R.: A markov prediction model for data-driven semi-structured business processes. Knowl. Inf. Syst. **42**(1) (2015) 97–126
9. Unuvar, M., Lakshmanan, G.T., Doganata, Y.N.: Leveraging path information to generate predictions for parallel business processes. Knowl. Inf. Syst. **47**(2) (2016) 433–461
10. Ceci, M., Lanotte, P.F., Fumarola, F., Cavallo, D.P., Malerba, D.: Completion time and next activity prediction of processes using sequential pattern mining. In: Discovery Science - 17th International Conference, DS 2014, Bled, Slovenia, October 8-10, 2014. Proceedings. (2014) 49–61
11. Breuker, D., Matzner, M., Delfmann, P., Becker, J.: Comprehensible predictive models for business processes. MIS Quarterly (2016)
12. Castellanos, M., Salazar, N., Casati, F., Dayal, U., Shan, M.: Predictive business operations management. In: Databases in Networked Information Systems, 4th International Workshop, DNIS 2005, Aizu-Wakamatsu, Japan, March 28-30, 2005, Proceedings. (2005) 1–14
13. Grigori, D., Casati, F., Castellanos, M., Dayal, U., Sayal, M., Shan, M.: Business process intelligence. Computers in Industry **53**(3) (2004) 321–343
14. Grigori, D., Casati, F., Dayal, U., Shan, M.: Improving business process quality through exception understanding, prediction, and prevention. In: VLDB 2001, Proceedings of 27th International Conference on Very Large Data Bases, September 11-14, 2001, Roma, Italy. (2001) 159–168
15. van Dongen, B.F., Crooy, R.A., van der Aalst, W.M.P.: Cycle time prediction: When will this case finally be finished? In: On the Move OTM Confederated International Conferences, Monterrey, Mexico, November 9-14, 2008, Proceedings, Part I. (2008) 319–336
16. Pandey, S., Nepal, S., Chen, S.: A test-bed for the evaluation of business process prediction techniques. In: 7th International Conference on Collaborative Computing: Networking, Applications and Worksharing, CollaborateCom 2011, Orlando, FL, USA, 15-18 October, 2011. (2011) 382–391
17. Kang, B., Kim, D., Kang, S.: Periodic performance prediction for real-time business process monitoring. Industrial Management and Data Systems **112**(1) (2011) 4–23
18. van der Aalst, W.M.P., Schonenberg, M.H., Song, M.: Time prediction based on process mining. Inf. Syst. **36**(2) (2011) 450–475
19. Folino, F., Guarascio, M., Pontieri, L.: Context-aware predictions on business processes: An ensemble-based solution. In: New Frontiers in Mining Complex Patterns - First International Workshop, NFMCP 2012, Held in Conjunction with ECML/PKDD 2012, Bristol, UK, September 24, 2012, Revised Selected Papers. (2012) 215–229
20. Folino, F., Guarascio, M., Pontieri, L.: Discovering context-aware models for predicting business process performances. In: On the Move OTM Confederated International Conferences, Rome, Italy, September 10-14, 2012. Proceedings, Part I. (2012) 287–304
21. Kang, B., Kim, D., Kang, S.: Real-time business process monitoring method for prediction of abnormal termination using knni-based LOF prediction. Expert Syst. Appl. **39**(5) (2012) 6061–6068
22. Conforti, R., de Leoni, M., Rosa, M.L., van der Aalst, W.M.P.: Supporting risk-informed decisions during business process execution. In: Advanced Information Systems Engineering - 25th International Conference, CAiSE 2013, Valencia, Spain, June 17-21, 2013. Proceedings. (2013) 116–132

23. Schwegmann, B., Matzner, M., Janiesch, C.: precep: Facilitating predictive event-driven process analytics. In: Design Science at the Intersection of Physical and Virtual Design - 8th International Conference, DESRIST 2013, Helsinki, Finland, June 11-12, 2013. Proceedings. (2013) 448–455
24. Rogge-Solti, A., Weske, M.: Prediction of remaining service execution time using stochastic petri nets with arbitrary firing delays. In: Service-Oriented Computing - 11th International Conference, ICSOC 2013, Berlin, Germany, December 2-5, 2013, Proceedings. (2013) 389–403
25. Rogge-Solti, A., Weske, M.: Prediction of business process durations using non-markovian stochastic petri nets. Inf. Syst. **54** (2015) 1–14
26. Bevacqua, A., Carnuccio, M., Folino, F., Guarascio, M., Pontieri, L.: A data-driven prediction framework for analyzing and monitoring business process performances. In: Enterprise Information Systems - 15th International Conference, ICEIS 2013, Angers, France, July 4-7, 2013, Revised Selected Papers. (2013) 100–117
27. Bolt, A., Sepúlveda, M.: Process remaining time prediction using query catalogs. In: Business Process Management Workshops - BPM 2013 International Workshops, Beijing, China, August 26, 2013, Revised Papers. (2013) 54–65
28. Polato, M., Sperduti, A., Burattin, A., de Leoni, M.: Data-aware remaining time prediction of business process instances. In: 2014 International Joint Conference on Neural Networks, IJCNN 2014, Beijing, China, July 6-11, 2014. (2014) 816–823
29. Polato, M., Sperduti, A., Burattin, A., de Leoni, M.: Time and activity sequence prediction of business process instances. CoRR **abs/1602.07566** (2016)
30. Maggi, F.M., Francescomarino, C.D., Dumas, M., Ghidini, C.: Predictive monitoring of business processes. CoRR **abs/1312.4874** (2013)
31. Maggi, F.M., Francescomarino, C.D., Dumas, M., Ghidini, C.: Predictive monitoring of business processes. In: Advanced Information Systems Engineering - 26th International Conference, CAiSE 2014, Thessaloniki, Greece, June 16-20, 2014. Proceedings. (2014) 457–472
32. Leontjeva, A., Conforti, R., Francescomarino, C.D., Dumas, M., Maggi, F.M.: Complex symbolic sequence encodings for predictive monitoring of business processes. In: Business Process Management - 13th International Conference, BPM 2015, Innsbruck, Austria, August 31 - September 3, 2015, Proceedings. (2015) 297–313
33. Metzger, A., Leitner, P., Ivanovic, D., Schmieders, E., Franklin, R., Carro, M., Dustdar, S., Pohl, K.: Comparing and combining predictive business process monitoring techniques. IEEE Trans. Systems, Man, and Cybernetics: Systems **45**(2) (2015) 276–290
34. Folino, F., Guarascio, M., Pontieri, L.: A prediction framework for proactively monitoring aggregate process-performance indicators. In: 19th IEEE International Enterprise Distributed Object Computing Conference, EDOC 2015, Adelaide, Australia, September 21-25, 2015. (2015) 128–133
35. Di Francescomarino, C., Dumas, M., Federici, M., Ghidini, C., Maggi, F.M., Rizzi, W.: Predictive business process monitoring framework with hyperparameter optimization. In: Proceedings of CAiSE. (2016) 361–376
36. Hochreiter, S., Schmidhuber, J.: Long short-term memory. Neural Computation **9**(8) (1997) 1735–1780
37. Sak, H., Senior, A.W., Beaufays, F.: Long short-term memory recurrent neural network architectures for large scale acoustic modeling. In: INTERSPEECH 2014, 15th Annual Conference of the International Speech Communication Association, Singapore, September 14-18, 2014. (2014) 338–342