# 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, St. John's, Canada
jevermann@mun.ca
[2] German Research Center for Artificial Intelligence, Saarbrücken, Germany
jana.rebecca.rehse@iwi.dfki.de
[3] Saarland University, Saarbrücken, Germany

**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.

**Keywords:** 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 (linear temporal logic) 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,


M. Dumas and M. Fantinato (Eds.): BPM 2016 Workshops, LNBIP 281, pp. 327–338, 2017.
DOI: 10.1007/978-3-319-58457-7_24

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 processing (NLP). In recent years, NLP research has moved from explicit representations of language models to statistical methods, specifically to recurrent neural networks (RNN) [4–6]. 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. Our idea is to treat an event trace as analogous to a natural language sentence, with the events analogous to words.

However, while there are many similarities between natural language and process traces, there are differences as well. First, the size of the vocabulary in process prediction (the number of event types) is much smaller than the size of a natural language vocabulary. Second, the length of a trace can far exceed the typical sentence length in natural language. Together, these two differences result in fewer possible prediction targets (vocabulary size or number of unique process event types) and more information to predict from (sentence or trace prefix length), suggesting that this approach may be able to achieve better results than, for example, word prediction in NLP. Third, process event sequences are determined or constrained by an internal process logic, typically determined by decision rules based on case data. However, natural language is also constrained, in the form of grammatical and morphological rules, for example the noun and verb agreement on plurals in English. Just as these linguistic rules and constraints are not explicitly captured in NLP deep-learning approaches [4–7] but are learned by the neural network when given sufficient training data, the process constraints and rules need not be explicitly represented in our approach either.

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

## 2 Related Work

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. Van Dongen et al. present a first approach using boosted regression on event frequencies, event times, and case data [8]. The approach by Pandey et al. applies a Hidden Markov Model (HMM) on event sequences and execution times [9]. It is based on an annotated transition system, as is the approach by Van der Aalst et al. [10]. Folino et al. present two contributions that use clustering trees and finite state machines (FSM) to predict the remaining time of a running process case [11,12]. Schwegmann et al. report on the development of a software tool that applies Complex Event Processing (CEP) to event sequences and is trained to predict their future behavior [13]. The two approaches by Rogge-Solti et al. use stochastic petri net simulation for the same purpose [14,15]. Bevacqua et al. present a prediction technique based on clustering and regression on case data [16]. Bolt et al. employ a clustering approach on partial and completed cases [17]. Finally, Polato et al. present two approaches that are based on annotated transition systems, as well as support vector regression and naive Bayes classifiers [18,19].

Another popular objective of process prediction is the binary assessment of its outcome, i.e. whether or not a process instance will fail. This was first addressed in the early 2000s by Castellanos et al. and Grigori et al., who use decision trees on time, resource, and case data [20–22]. Decision trees are also used in the approach by Conforti et al. [23]. Kang et al. present two different approaches to predicting process failures, one using a support vector machine (SVM) [24] and another one based on clustering and local outlier detection [25]. Maggi et al. employ decision trees to predict violation of LTL (Linear Temporal Logic) restrictions [26,27]. Leontjeva et al. use random forests for the same purpose [28]. These two techniques are combined in the approach by Francescomarino et al. [29]. Both Metzger et al. and Folino et al. are concerned with the binary outcome in terms of the completion past promise; while the former employs neural networks, constraint satisfaction, and Quality-of-Service aggregation [30], the latter relies on clustering and regression [31].

Only five approaches [32–36] are concerned with predicting the next event, many of which use an explicit process model representation such as HMM and PFA (Probabilistic Finite Automatons).

The MSA approach in [32] 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 [33,34] 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 [35] 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 [36] 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.

Our approach has the same objective, i.e. predictand, as these five previous works, but differs from the in terms of predictors, method, and process representation. Deep Learning in the form of an RNN is used to predict the next events, using event sequences and associated resources. Processes are only implicitly represented within the RNN. Overall, our method constitutes an innovative new approach to process prediction, described in the following sections.

## 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 $l$.

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. 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.

**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

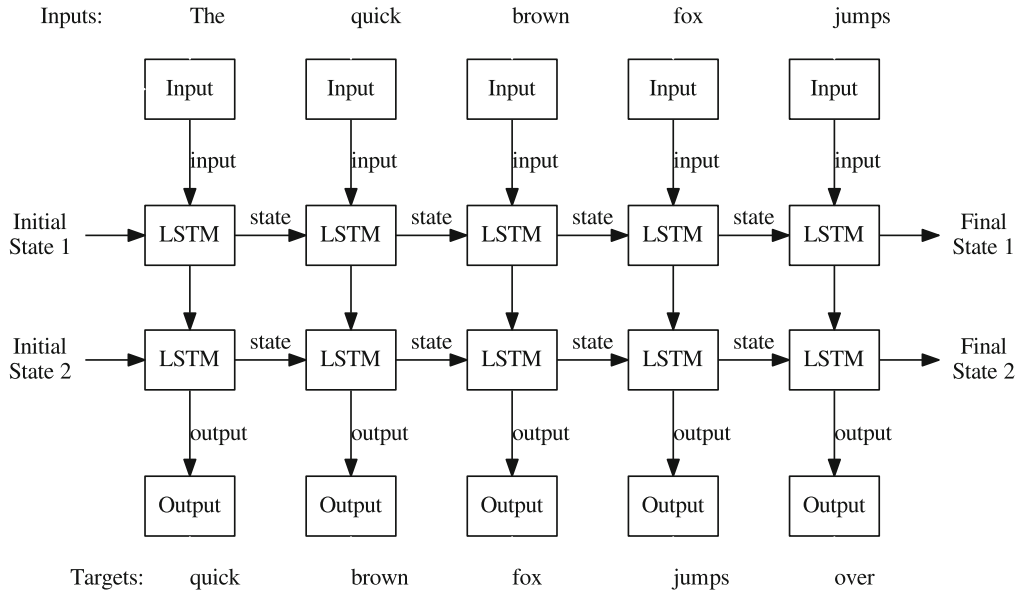


**Fig. 1.** RNN architecture with single hidden layer of LSTM cells, unrolled five steps

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).

**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 [37]. A basic LSTM cell is defined as follows, accepting $C_{t-1}$ and $h_{t-1}$ as state and input 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 state and output information to the subsequent unrolled cell and provides $h_t$ as output to the next layer; the various $W$ and $b$ are "trainable" parameters.

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

Intuitively, $f_t$ represents the "forget gate", allowing the LSTM cell to selectively suppress information from $C_{t-1}$. Similarly, $i_t$ represents the "input gate", allowing the LSTM cell to selectively add input to the state. Note how the new state $C_t$ is computed by first forgetting $f_t \times C_{t-1}$ and then adding $i_t \times \bar{C}_t$ where $\bar{C}_t$ is the "candidate state information" computed from the inputs $x_t$ and $h_{t-1}$. Similarly, $o_t$ represents the output gate, allowing the LSTM cell to selectively output part of the new state to the following cells.

**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 $P = \exp(H/n)$, where $n$ is the number of possible words (targets). Perplexity measures the "surprisedness" the network exhibits when encountering the next term. There are no absolute guidelines as to what an acceptable level of perplexity is, however, 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 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 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 Sect. 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 [36] used for their study. In addition to separating the BPI 2012 data set by sub-process as done in [36], we also used the combined dataset. While [36] 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 allows 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 1 summarizes

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

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

**Table 1.** Characteristics of datasets used in experimental evaluation. Event numbers for partial BPI2012 logs do not add up to that of corresponding complete log due to end-of-case marker events added to each trace.

| Dataset | Number of unique event types ("vocabulary size") | Number of unique event types ("vocabulary size") (with resource or org. group) | Number of events |
|---|---|---|---|
| BPI2013.Incidents | 14 | 3133 | 73087 |
| BPI2013.Problems | 8 | 64 | 11317 |
| BPI2012 (completion events) | 24 | 877 | 177593 |
| BPI2012 (all events) | 37 | 1349 | 275287 |
| BPI2012.W (completion events) | 7 | 264 | 82071 |
| BPI2012.A (completion events) | 11 | 302 | 73936 |
| BPI2012.O (completion events) | 8 | 313 | 36259 |
| BPI2012.W (all events) | 7 | 736 | 179765 |
| BPI2012.A (all events) | 11 | 302 | 73939 |
| BPI2012.O (all events) | 8 | 313 | 36259 |

**Table 2.** Results and comparison to [36]. Results are means of three runs. Precision is defined as the proportion of correct predictions among all predictions. Higher precision is better; lower perplexity is better.

| Dataset | Precision in [36] | 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 |

the characteristics of the datasets. We transformed the published XES datasets using XSL transformations to extract traces, events, and resource information in a suitable format.

Table 2 shows our results and a comparison to the best result presented by [36], where available. While [36] report a cross-entropy $H$ in addition to accuracy, their definition of $H$ in their Fig. 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 [36] 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, 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 [36], 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 [36] 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 neural 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 [38], 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., Aalst, W.M.P., Krogstie, J.: BPM-in-the-large – towards a higher level of abstraction in business process management. In: Janssen, M., Lamersdorf, W., Pries-Heje, J., Rosemann, M. (eds.) EGES/GISP - 2010. IAICT, vol. 334, pp. 233–244. Springer, Heidelberg (2010). doi:10.1007/978-3-642-15346-4_19
2. Schmidhuber, J.: Deep learning in neural networks: an overview. Neural Netw. **61**, 85–117 (2015)
3. LeCun, Y., Bengio, Y., Hinton, G.: Deep learning. Nature **521**, 436–444 (2015)
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, pp. 1017–1024 (2011)
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. Graves, A.: Supervised Sequence Labelling with Recurrent Neural Networks. SCI, vol. 385. Springer, New York (2012)
8. Dongen, B.F., Crooy, R.A., Aalst, W.M.P.: Cycle time prediction: When will this case finally be finished? In: Meersman, R., Tari, Z. (eds.) OTM 2008. LNCS, vol. 5331, pp. 319–336. Springer, Heidelberg (2008). doi:10.1007/978-3-540-88871-0_22
9. 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, pp. 382–391 (2011)
10. van der Aalst, W.M.P., Schonenberg, M.H., Song, M.: Time prediction based on process mining. Inf. Syst. **36**(2), 450–475 (2011)
11. Folino, F., Guarascio, M., Pontieri, L.: Context-aware predictions on business processes: an ensemble-based solution. In: Appice, A., Ceci, M., Loglisci, C., Manco, G., Masciari, E., Ras, Z.W. (eds.) NFMCP 2012. LNCS (LNAI), vol. 7765, pp. 215–229. Springer, Heidelberg (2013). doi:10.1007/978-3-642-37382-4_15
12. Folino, F., Guarascio, M., Pontieri, L.: Discovering context-aware models for predicting business process performances. In: Meersman, R., et al. (eds.) OTM 2012. LNCS, vol. 7565, pp. 287–304. Springer, Heidelberg (2012). doi:10.1007/978-3-642-33606-5_18

13. Schwegmann, B., Matzner, M., Janiesch, C.: preCEP: facilitating predictive event-driven process analytics. In: Brocke, J., Hekkala, R., Ram, S., Rossi, M. (eds.) DESRIST 2013. LNCS, vol. 7939, pp. 448–455. Springer, Heidelberg (2013). doi:10.1007/978-3-642-38827-9_36
14. Rogge-Solti, A., Weske, M.: Prediction of remaining service execution time using stochastic petri nets with arbitrary firing delays. In: Basu, S., Pautasso, C., Zhang, L., Fu, X. (eds.) ICSOC 2013. LNCS, vol. 8274, pp. 389–403. Springer, Heidelberg (2013). doi:10.1007/978-3-642-45005-1_27
15. Rogge-Solti, A., Weske, M.: Prediction of business process durations using non-markovian stochastic petri nets. Inf. Syst. **54**, 1–14 (2015)
16. Bevacqua, A., Carnuccio, M., Folino, F., Guarascio, M., Pontieri, L.: A data-driven prediction framework for analyzing and monitoring business process performances. In: Hammoudi, S., Cordeiro, J., Maciaszek, L.A., Filipe, J. (eds.) ICEIS 2013. LNBIP, vol. 190, pp. 100–117. Springer, Cham (2014). doi:10.1007/978-3-319-09492-2_7
17. Bolt, A., Sepúlveda, M.: Process remaining time prediction using query catalogs. In: Lohmann, N., Song, M., Wohed, P. (eds.) BPM 2013. LNBIP, vol. 171, pp. 54–65. Springer, Cham (2014). doi:10.1007/978-3-319-06257-0_5
18. 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, pp. 816–823 (2014)
19. Polato, M., Sperduti, A., Burattin, A., de Leoni, M.: Time and activity sequence prediction of business process instances. CoRR abs/1602.07566 (2016)
20. Castellanos, M., Salazar, N., Casati, F., Dayal, U., Shan, M.-C.: Predictive business operations management. In: Bhalla, S. (ed.) DNIS 2005. LNCS, vol. 3433, pp. 1–14. Springer, Heidelberg (2005). doi:10.1007/978-3-540-31970-2_1
21. Grigori, D., Casati, F., Castellanos, M., Dayal, U., Sayal, M., Shan, M.: Business process intelligence. Comput. Ind. **53**(3), 321–343 (2004)
22. 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, pp. 159–168 (2001)
23. Conforti, R., Leoni, M., Rosa, M., Aalst, W.M.P.: Supporting risk-informed decisions during business process execution. In: Salinesi, C., Norrie, M.C., Pastor, Ó. (eds.) CAiSE 2013. LNCS, vol. 7908, pp. 116–132. Springer, Heidelberg (2013). doi:10.1007/978-3-642-38709-8_8
24. Kang, B., Kim, D., Kang, S.: Periodic performance prediction for real-time business process monitoring. Ind. Manage. Data Syst. **112**(1), 4–23 (2011)
25. 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), 6061–6068 (2012)
26. Maggi, F.M., Francescomarino, C.D., Dumas, M., Ghidini, C.: Predictive monitoring of business processes. CoRR abs/1312.4874 (2013)
27. Maggi, F.M., Francescomarino, C., Dumas, M., Ghidini, C.: Predictive monitoring of business processes. In: Jarke, M., Mylopoulos, J., Quix, C., Rolland, C., Manolopoulos, Y., Mouratidis, H., Horkoff, J. (eds.) CAiSE 2014. LNCS, vol. 8484, pp. 457–472. Springer, Cham (2014). doi:10.1007/978-3-319-07881-6_31

28. Leontjeva, A., Conforti, R., Francescomarino, C., Dumas, M., Maggi, F.M.: Complex symbolic sequence encodings for predictive monitoring of business processes. In: Motahari-Nezhad, H.R., Recker, J., Weidlich, M. (eds.) BPM 2015. LNCS, vol. 9253, pp. 297–313. Springer, Cham (2015). doi:10.1007/978-3-319-23063-4_21
29. Francescomarino, C., Dumas, M., Federici, M., Ghidini, C., Maggi, F.M., Rizzi, W.: Predictive business process monitoring framework with hyperparameter optimization. In: Nurcan, S., Soffer, P., Bajec, M., Eder, J. (eds.) CAiSE 2016. LNCS, vol. 9694, pp. 361–376. Springer, Cham (2016). doi:10.1007/978-3-319-39696-5_22
30. 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. Syst. Man Cybern. Syst. **45**(2), 276–290 (2015)
31. 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, pp. 128–133 (2015)
32. 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, pp. 179–192 (2012)
33. 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), 97–126 (2015)
34. Unuvar, M., Lakshmanan, G.T., Doganata, Y.N.: Leveraging path information to generate predictions for parallel business processes. Knowl. Inf. Syst. **47**(2), 433–461 (2016)
35. 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: Džeroski, S., Panov, P., Kocev, D., Todorovski, L. (eds.) DS 2014. LNCS (LNAI), vol. 8777, pp. 49–61. Springer, Cham (2014). doi:10.1007/978-3-319-11812-3_5
36. Breuker, D., Matzner, M., Delfmann, P., Becker, J.: Comprehensible predictive models for business processes. MIS Q. **40**, 1009–1034 (2016)
37. Hochreiter, S., Schmidhuber, J.: Long short-term memory. Neural Comput. **9**(8), 1735–1780 (1997)
38. 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, pp. 338–342 (2014)