Remaining Time Prediction in Outbound Warehouse Processes: A Case Study (Short Paper)

Erik Penther[1], Michael Grohs[1 [0000−0003−2658−8992]], and Jana-Rebecca Rehse[1 [0000−0001−5707−6944]]

University of Mannheim, Mannheim, Germany
{erik.penther@students., michael.grohs@, rehse@}uni-mannheim.de

Abstract. Predictive process monitoring is a sub-domain of process mining which aims to forecast the future of ongoing process executions. One common prediction target is the remaining time, meaning the time that will elapse until a process execution is completed. In this paper, we compare four different remaining time prediction approaches in a real-life outbound warehouse process of a logistics company in the aviation business. For this process, the company provided us with a novel and original event log with 169,523 traces, which we can make publicly available. Unsurprisingly, we find that deep learning models achieve the highest accuracy, but shallow methods like conventional boosting techniques achieve competitive accuracy and require significantly fewer computational resources.

Keywords: Predictive Process Monitoring · Remaining Time Prediction · Case Study.

1 Introduction

Process mining is a family of data analysis techniques that aims to provide insights into business processes within organizations [4]. To this end, process mining techniques analyze recorded process executions, so-called traces, which are captured and stored collectively in event logs [4]. A well-established sub-discipline of process mining is Predictive Process Monitoring (PPM), which focuses on predicting the future progression of ongoing traces [3]. One of the most common prediction targets is the remaining time until completion of traces [5]. Remaining time prediction can help to avoid deadline violations, improve operational efficiency, and provide estimates to customers [1]. In recent years, numerous approaches for remaining time prediction have been proposed, which aim to address the challenges of the task such as capturing long-range dependencies and harnessing process perspectives other than the control-flow [1]. Many approaches are based on deep learning models [9], but traditional machine learning methods, such as boosting, have also been employed [7].

Given this wide availability of approaches, companies who want to do remaining time prediction typically have to choose a well-suited approach for their application case, based on the characteristics of the process in question [2]. Relevant characteristics may include process complexity, the availability of static and dynamic data attributes, among other factors [3]. Often, multiple candidate approach may appear suitable for a given use case, making the selection of an appropriate technique a non-trivial task [2].

In this paper, we illustrate this selection of an appropriate remaining time prediction technique by comparing the performance of different approaches for a real-life outbound warehouse process. For this process, we obtained an event log from a company, which we can make publicly available. To this event log, we apply different state-of-the-art remaining time prediction approaches to determine the most suitable one. In particular, we apply three sophisticated deep learning techniques to the outbound warehouse process and assess how accurately they predict the remaining time of ongoing traces. In addition, we employ a rather simple baseline approach based on XGBoost. Our findings indicate that, unsurprisingly, deep learning approaches are more accurate than our simple baseline but also require substantial training effort. However, XGBoost performs competitively and even outperforms one deep learning approach. Consequently, it can be a viable alternative since it requires significantly fewer computational resources.

2 Our Case: Outbound Warehouse Process

The process analyzed in this case study originates from a company that provides logistics services for the aviation industry. The goal is to ensure the efficient supply of aircraft components, thereby focusing on smaller items such as spare parts. Since the shipment of these smaller items is standardized, the control-flow is relatively straightforward and linear. However, cycle times can vary due to factors such as item type or weight. This can cause difficulties in obtaining accurate forecasts for the delivery, but customers require such forecasts to plan the maintenance and repair of aircraft supplies.

For this process, we obtained an event log with 169,523 traces that we make publicly available in an anonymized form.[1] Based on this event log, the process managers seek for a reliable remaining time prediction approach to provide accurate time predictions for downstream tasks that use parts from the warehouse. Each trace in the log refers to one order that corresponds to one concrete order item. The desired control-flow consists of seven subsequent activities, of which one is optional. 24 attributes are recorded for each trace, which can be utilized in remaining time prediction.[2] From these 24 attributes, 20 are categorical (e.g., the items' type) and four are numerical (e.g., the shipment weight). We note that the log has been anonymized due to data privacy reasons. Concretely, activity names have been altered and trace attributes are enumerated and hashed. Nevertheless, the log contains real-life behavior that can be used independently of the fact that activities and attributes have no semantic meaning anymore.

3 Remaining Time Prediction

Event logs, traces, and prefixes. An event log is a collection of traces, where each trace corresponds to a process execution. A trace consists of a time-ordered sequence of events, each indicating the execution of a specific activity during the process. Traces and events can have numerical or categorical attributes. For example, a trace attribute

[1] http://figshare.com/articles/dataset/Warehouse_outbound_event_log/29500898

[2] The event log does not contain an event attribute for the executing resource due to privacy and data security reasons.

can be the value of shipped goods whereas the executing resource can be an event attribute. A running trace is referred to as a prefix, which is a sub-sequence of the trace starting from the initial activity. The prefix length indicates how many events from the trace are included. For example, a prefix of length 3 includes the first three events.

Remaining time prediction task. Remaining time prediction tries to forecast the time that will elapse until an ongoing trace is completed [3]. Typically, this is done by training a predictive model on a subset of traces, the so-called training set. The performance of the models is then evaluated on the remaining traces, the so-called test set [3]. Models can utilize all recorded attributes of the event data such as activities or timestamps [2].

4 Case Study Method

This section presents the steps we took to assess the suitability of different remaining time prediction approaches for the outbound warehouse process.

Pre-Processing. In the beginning, we pre-processed the event log to ensure that we work with accurate and reliable data. For that, we removed outliers to account for logging issues due to wrong or outdated master data or manual entries. First, we removed traces with logically impossible durations or traces taking over half a year. Second, we removed outliers with a weight of more than the 95th percentile since heavier values would suggest the shipment of a whole aircraft engine which is not part of this process. This reduced the number of traces to 130,835, with 1,043,555 events.

In addition, we selected a time frame that contains only the current process version to account for concept drifts, i.e., data from multiple versions of the same process. In particular, a concept drift occurred in the process behavior in May 2024, where some process variants were discontinued whereas others were executed more often. Therefore, we used only data from June 2024 onward. This reduced the number of traces to 41,927 consisting of 330,709 events (7.9 events on average). The average cycle time of traces was 24 hours, with a maximum of 192 hours, indicating quick cycle times.

Feature Selection. Next, we selected the features to be used for prediction. First, we removed uninformative features, striving to keep only informative features in a low-dimensional feature space. This is desired since high-dimensional feature spaces negatively impact learning algorithms [6]. We removed three categorical feature with several thousand realizations that likely possess only low predictive power.[3] Also, we removed two features with only one value, which, by definition, do not have any predictive power.

After that, we selected a subset of features that has potentially larger predictive power based on the Mutual Information (MI) shared between a feature and the remaining time of prefixes in the training set [8]. All features were ranked, and only those with MI scores above 1 were retained. This reduced the number of features to eleven.

Finally, we created additional features for which the managers of the process know that they have predictive power. These include attributes such as *time since trace started*, *time since last event*, and *day of the week*. Also, we calculated the number of concurrent traces open at the time of the event as a basic inter-case feature. This may help the algorithm to learn capacity utilization since traces share the same resources.

[3] The features cannot be explicated due to privacy reasons. They refer to the categorical features 16, 18, and 20 in the anonymized event log.

Dataset Partitioning. We split the event log into a training and test set using a 70-30 split. Also, we used the last 10% of the traces in the training set as the validation set.
Predictive Approach Selection. For finding the most suitable approach for remaining time prediction, we pre-selected four approaches to be tested, based on the most recent developments in remaining time prediction: (1) A data-aware LSTM approach [5] (2) A transformer-based approach (SuTraN) [9] (3) A graph transformer-based approach (PGTNet) [1] (4) An XGBoost-based approach [7]

The former three can be considered state-of-the-art approaches for remaining time prediction, which represent distinct and promising paradigms: LSTMs as one of the first deep learning cells to process sequential data, SuTraN as a rather novel encoder-decoder transformer, and PGTNet with its innovative graph-based event log representations. All three potentially suit our process at hand. The last approach XGBoost serves as a less sophisticated baseline to show how deep learning relates to other ML approaches.
Experimental Setting. The experiment was conducted using an Apple M1 Pro with 32 GB shared memory in a Python 3.10 environment. The implementations of the deep learning algorithms of LSTM and SuTraN [9] as well as PGTNet [1] are based on PyTorch and initially utilize NVIDIA CUDA cores. Since these were not available in our setting, the implementations were altered to use available GPU resources.
Metrics. We apply the commonly used Mean Absolute Error (MAE) as evaluation metric for remaining time prediction as it provides a clear accuracy measure [7].
Hyperparameter Optimization. We optimized hyperparameters of all approaches with a grid search, depending on the approach. Concretely, we optimized these parameters:
(1) LSTM: hidden layer size, no. of shared layers, no. of dedicated layers, batch size, batch interval, dropout
(2) SuTraN: hidden layer size, no. of prefix encoder layers, no. of decoder layers, no. of heads, batch size, batch interval, dropout
(3) PGTNet: positional encoding dimensions, positional encoding times, no. of layers, no. of heads, dropout
(4) XGBoost: no. of estimators, learning rate, subsample, colsample, maximum depth

We selected the best performing hyperparameters for each approach based on the validation set. The next section presents the results of the approaches.

5 Results

In this section, we present the results of our case study. Our evaluation pipeline including the original and pre-processed event log can be found online.[4]
Overall performance. Table 1 presents the evaluation results of the four different approaches in form of MAE as well as training and inference time. Among the approaches, SuTraN achieves the lowest MAE with 554 minutes, suggesting that it delivers the most accurate predictions. LSTM follows closely with a MAE of 568 minutes, performing only slightly worse than SuTraN. XGBoost has an MAE of 613 minutes, making it the third most accurate approach. In contrast, PGTNet shows a significantly higher MAE of 1390 minutes, likely due to overfitting on training data.

[4] https://github.com/ultrawaffle/ppm_remaining_time

With respect to required time, XGBoost is the fastest to train, requiring only 2 minutes, and has the shortest inference time of 0.10 ms. This suggests that it is highly efficient for quick retraining and real-time predictions. In comparison, LSTM requires a much longer training time of 1.26 hours, but its inference time remains relatively low at 0.63 ms. SuTraN, despite achieving the best predictive performance, has the longest training time of 4.65 hours and a relatively higher inference time of 3.17 ms, indicating that its accuracy comes at the cost of more computational requirements. PGTNet, on the other hand, takes 0.8 hours to train, which is moderate compared to the other models, but its inference time is high at 95.39 ms, making it the least efficient.

Table 1: Average Results of the Models Performances

Approach	**MAE (min)**	**Training time**	**Inference time**
LSTM	568	1.26 h	0.63 ms
SuTraN	**554**	4.65 h	3.17 ms
PGTNet	1390	0.8 h	95.39 ms
XGBoost	613	**2 min**	**0.10 ms**

Discussion. We found that some models demonstrate advantages in specific areas but also unraveled challenges that impact their applicability.

LSTM achieves better accuracy than XGBoost. However, LSTMs require longer training times and demand more computational resources. Also, static trace attributes cannot be learned dynamically. Additionally, in this specific experiment, LSTMs tended to underestimate the remaining time, which could be a bigger problem than overestimation as customers might already need the shipped parts when promised, whereas overestimated times are likely to cause no shortages at the customers' sites.

SuTraN, a transformer-based model, leverages attention mechanisms to capture long-term dependencies. Among all tested models, SuTraN achieved the best performance in this use case. However, it comes with significantly more computational demands than the other tested models, leading to the longest training time.

PGTNet has demonstrated strong performance in other studies where the dataset is sufficiently complex [7]. However, in our use case, PGTNet struggled with overfitting, even after extensive hyperparameter tuning. This suggests that its architecture may be too complex for the given event log. Thus, PGTNet was ultimately deemed unsuitable.

XGBoost is an efficient model, offering advantages in both training and inference speed. It is the quickest out of the tested approaches. However, XGBoost does not inherently model temporal dependencies and we have to use specific encoding strategies such as aggregation or index encoding [3], potentially leading to information loss. This might explain the worse accuracy in contrast to LSTM and SuTraN.

6 Conclusion

In this paper, we evaluated the accuracy of four models in predicting the remaining time in process instances of a warehouse outbound process. Our work has both practical and scientific implications. For practitioners, our study shows that there are alternative predictive approaches among which should be chosen carefully. Especially if periodic re-

training is required, shallow approaches like XGBoost might be suited better. Thus, the model choice has to be adapted to the process at hand. For researchers, our study shows that there is still room for improvement in predictive approaches. In particular, even the state-of-the-art approaches are not able to reduce the MAE below about 9 hours, a large error relative to an average trace duration of only 27 hours. Further, our findings suggest simple architectures suit shorter processes, while other studies found that larger ones handle complex tasks better, indicating potential benefits of hybrid strategies.

Our study is suspect to some limitations. First, we acknowledge that there are factors that impact processing times that we were not able to quantify. For instance, executing resources were not recorded in the event log. Second, the hyperparameter optimization and model testing can always be further refined. Although an extensive tuning process was conducted, the models are highly sensitive to hyperparameters, and more exhaustive tuning could lead to further improvements. Finally, we only applied a subset of all remaining time prediction approaches. Nevertheless, we believe that the four applied models cover a wide range of recent works and a less sophisticated baseline.

Last, we want to stress the fact that we provide a novel public dataset based on a real-life process in the repository referenced above. Although the log is anonymized due to privacy reasons, the recorded behavior is still a detailed depiction of reality. We highly encourage fellow researchers to use this dataset for their own research and, for example, try to discover and characterize the underlying concept drift in the event log. Given that the availability of public event logs is of utmost important for the validity of experimental evaluations, we believe that the publication of the event log can help researchers in the field of process mining when conducting their experiments.

References

1. Amiri Elyasi, K., van der Aa, H., Stuckenschmidt, H.: PGTNet: A Process Graph Transformer Network for Remaining Time Prediction of Business Process Instances. In: CAiSE. pp. 124–140 (2024)
2. Di Francescomarino, C., Dumas, M., Federici, M., Ghidini, C., Maggi, F.M., Rizzi, W., Simonetto, L.: Genetic algorithms for hyperparameter optimization in predictive business process monitoring. Inf Syst **74**, 67–83 (2018)
3. Di Francescomarino, C., Ghidini, C.: Predictive Process Monitoring. In: Process Mining Handbook, pp. 320–346. Springer (2022)
4. Dumas, M., La Rosa, M., Mendling, J., Reijers, H.A.: Fundamentals of Business Process Management. Springer (2018)
5. Gunnarsson, B.R., Broucke, S.v., De Weerdt, J.: A Direct Data Aware LSTM Neural Network Architecture for Complete Remaining Trace and Runtime Prediction. IEEE Trans Serv Comput **16**(4), 2330–2342 (2023)
6. Maimon, O., Rokach, L.: Data Mining and Knowledge Discovery Handbook. Springer (2010)
7. Roider, J., Nguyen, A., Zanca, D., M.Eskofier, B.: Assessing the Performance of Remaining Time Prediction Methods for Business Processes. IEEE Access pp. 130583–130601 (2024)
8. Vergara, J.R., Estévez, P.A.: A review of feature selection methods based on mutual information. Neural Comput Appl **24**(1), 175–186 (2014)
9. Wuyts, B., Vanden Broucke, S., De Weerdt, J.: SuTraN: an Encoder-Decoder Transformer for Full-Context-Aware Suffix Prediction of Business Processes. In: ICPM. pp. 17–24 (2024)