# From Source to Target: Leveraging Transfer Learning for Predictive Process Monitoring in Organizations

**Sven Weinzierl, Sandra Zilker, Annina Liessmann, Martin Käppel, Weixin Wang, Martin Matzner**

Business & Information Systems Engineering (2026)

**Appendix (available online via http://link.springer.com)**

Online Appendix

# Appendix A. Prediction performance for complete model compared to model sub-variants

We conducted an additional evaluation in the form of an ablation analysis to assess the importance of two main components of our proposed model on prediction performance. These components include the second LSTM layer, which transforms the model into an encoder, and the dedicated parameter initialization, which is performed before model training. When the dedicated parameter initialization is deactivated, all model weights and biases are initialized uniformly. Both model components are designed to improve learning from the event data of the source context and facilitate the transfer of process knowledge to the target context.

Table 12 shows that the complete model of our proposed TL-based technique clearly outperforms its sub-variants in terms of prediction performance when it is trained on $L_S$ and tested on $L_T$ for the inter-organizational use case. This confirms the importance of using both components in the prediction model for an effective process knowledge transfer from source to target contexts.

Table 12: Prediction performance of complete model compared to its sub-variants for the inter-organizational use case (average and standard deviation over five runs)

| Scenario | Precision | Recall | F1-score | MCC | $AUC_{ROC}$ |
|---|---|---|---|---|---|
| COMPLETE MODEL | | | | | |
| **Model with two LSTM layers and dedicated parameter initialization** | | | | | |
| Train and test on $L_S$ | 0.735 (±.003) | 0.694 (±.003) | 0.708 (±.002) | 0.293 (±.007) | 0.730 (±.005) |
| Train and test on $L_T$ | 0.769 (±.012) | 0.769 (±.037) | 0.766 (±.024) | 0.330 (±.034) | 0.775 (±.018) |
| Train on $L_S$, test on $L_T$ | **0.739** (±.020) | **0.755** (±.050) | **0.724** (±.021) | **0.206** (±.029) | **0.711** (±.020) |
| SUB-VARIANTS OF MODEL | | | | | |
| **Model with two LSTM layers and without dedicated parameter initialization** | | | | | |
| Train and test on $L_S$ | 0.737 (±.005) | 0.691 (±.009) | 0.706 (±.006) | 0.296 (±.010) | 0.730 (±.005) |
| Train and test on $L_T$ | 0.770 (±.011) | 0.783 (±.015) | 0.775 (±.011) | 0.336 (±.027) | 0.767 (±.015) |
| Train on $L_S$, test on $L_T$ | 0.712 (±.037) | 0.701 (±.048) | 0.702 (±.035) | 0.167 (±.099) | 0.666 (±.063) |
| **Model with one LSTM layer and with dedicated parameter initialization** | | | | | |
| Train and test on $L_S$ | 0.738 (±.004) | 0.696 (±.009) | 0.710 (±.007) | 0.301 (±.007) | 0.734 (±.004) |
| Train and test on $L_T$ | 0.756 (±.021) | 0.743 (±.058) | 0.745 (±.043) | 0.292 (±.062) | 0.728 (±.030) |
| Train on $L_S$, test on $L_T$ | 0.719 (±.023) | 0.694 (±.046) | 0.693 (±.009) | 0.171 (±.058) | 0.701 (±.022) |
| **Model with one LSTM layer and without dedicated parameter initialization** | | | | | |
| Train and test on $L_S$ | 0.739 (±.005) | 0.686 (±.008) | 0.702 (±.006) | 0.299 (±.009) | 0.731 (±.006) |
| Train and test on $L_T$ | 0.764 (±.016) | 0.774 (±.025) | 0.767 (±.019) | 0.318 (±.045) | 0.741 (±.032) |
| Train on $L_S$, test on $L_T$ | 0.702 (±.026) | 0.688 (±.040) | 0.687 (±.017) | 0.133 (±.068) | 0.670 (±.044) |

*Note.* Best results for training on $L_S$ and testing on $L_T$ are marked in bold

## Appendix B. Prediction performance for different prefix encoding approaches

We conducted an additional evaluation to assess the effect of index-based prefix encoding used in our proposed TL-based PPM technique on the prediction performance compared to alternative prefix encoding approaches. The two alternative prefix encoding approaches used in this evaluation were last-3-state encoding and aggregation encoding. Both are common prefix encoding approaches in outcome-oriented PPM (Teinemaa et al. 2019). The first approach, last-3-state encoding, encodes each prefix only by the last three events, or fewer, if the complete trace includes fewer than three events. The second approach, aggregation encoding, encodes each prefix in an aggregated form by calculating the mean value for each attribute across all events in the prefix.

We applied the index-based encoding, the last-3-state encoding, and the aggregation encoding to our proposed TL-based PPM technique for the inter-organizational use case. We also applied these three encoding approaches to two traditional PPM techniques, using LSTM and XGBoost models. We selected these two traditional PPM techniques because the first uses the same LSTM model as in our proposed technique but not in an TL-setting, and the second uses an XGBoost model, which performed relatively well in the first evaluation of our inter-case organizational use case (see Section 7.2.1).

Table 13 shows that our proposed technique with the index-based prefix encoding clearly outperforms all baseline approaches in terms of prediction performance on the test event data of the target context. This suggests that index-based encoding is an effective prefix encoding approach for our proposed TL-based PPM technique.

Table 13: Prediction performance of our proposed TL-based PPM technique and traditional PPM techniques with different prefix encoding approaches for the inter-organizational use case (average and standard deviation over five runs)

| ML approach | Precision | Recall | F1-score | MCC | $AUC_{ROC}$ |
|---|---|---|---|---|---|
| LSTM Model with Activity Embedding and Relative Cross-domain Timestamp Encoding | | | | | |
| **Index-based encoding** | | | | | |
| Train and test on $L_S$ | 0.735 (±.003) | 0.694 (±.003) | 0.708 (±.002) | 0.293 (±.007) | 0.730 (±.005) |
| Train and test on $L_T$ | 0.769 (±.012) | 0.769 (±.037) | 0.766 (±.024) | 0.330 (±.034) | 0.775 (±.018) |
| Train on $L_S$, test on $L_T$ | 0.739 (±.020) | 0.755 (±.050) | **0.724** (±.021) | 0.206 (±.029) | **0.711** (±.020) |
| **Last-3-state encoding** | | | | | |
| Train and test on $L_S$ | 0.745 (±.003) | 0.680 (±.009) | 0.698 (±.007) | 0.310 (±.003) | 0.745 (±.001) |
| Train and test on $L_T$ | 0.778 (±.020) | 0.792 (±.026) | 0.779 (±.017) | 0.351 (±.044) | 0.779 (±.013) |
| Train on $L_S$, test on $L_T$ | 0.725 (±.014) | 0.694 (±.054) | 0.699 (±.028) | 0.195 (±.039) | 0.709 (±.025) |
| **Aggregation encoding** | | | | | |
| Train and test on $L_S$ | 0.724 (±.016) | 0.697 (±.015) | 0.704 (±.003) | 0.265 (±.039) | 0.708 (±.025) |
| Train and test on $L_T$ | 0.731 (±.004) | 0.741 (±.018) | 0.734 (±.008) | 0.225 (±.015) | 0.706 (±.013) |
| Train on $L_S$, test on $L_T$ | 0.718 (±.023) | **0.756** (±.046) | 0.685 (±.007) | 0.074 (±.015) | 0.542 (±.027) |
| LSTM Model with One-hot Activity Encoding and Min-max time Feature Encoding | | | | | |
| **Index-based encoding** | | | | | |
| Train and test on $L_S$ | 0.737 (±.002) | 0.677 (±.004) | 0.695 (±.003) | 0.292 (±.005) | 0.721 (±.005) |
| Train and test on $L_T$ | 0.676 (±.047) | 0.775 (±.000) | 0.678 (±.001) | 0.011 (±.016) | 0.567 (±.002) |
| Train on $L_S$, test on $L_T$ | 0.050 (±.000) | 0.224 (±.000) | 0.082 (±.000) | -0.029 (±.000) | 0.430 (±.009) |
| **Last-3-state encoding** | | | | | |
| Train and test on $L_S$ | 0.745 (±.002) | 0.675 (±.005) | 0.694 (±.004) | 0.307 (±.002) | 0.741 (±.002) |
| Train and test on $L_T$ | 0.624 (±.050) | 0.776 (±.000) | 0.678 (±.000) | 0.004 (±.009) | 0.556 (±.004) |
| Train on $L_S$, test on $L_T$ | 0.143 (±.208) | 0.225 (±.001) | 0.084 (±.003) | -0.007 (±.015) | 0.442 (±.014) |
| **Aggregation encoding** | | | | | |
| Train and test on $L_S$ | 0.725 (±.002) | 0.702 (±.005) | 0.711 (±.003) | 0.271 (±.004) | 0.713 (±.003) |
| Train and test on $L_T$ | 0.602 (±.000) | 0.776 (±.000) | 0.678 (±.000) | 0.000 (±.000) | 0.528 (±.003) |
| Train on $L_S$, test on $L_T$ | 0.050 (±.000) | 0.224 (±.000) | 0.082 (±.000) | -0.029 (±.000) | 0.450 (±.027) |
| XGBoost with One-hot Activity Encoding | | | | | |
| **Index-based encoding** | | | | | |
| Train and test on $L_S$ | 0.714 (±.002) | 0.743 (±.002) | 0.720 (±.001) | 0.234 (±.004) | 0.738 (±.002) |
| Train and test on $L_T$ | 0.793 (±.000) | 0.811 (±.000) | 0.793 (±.000) | 0.386 (±.000) | 0.759 (±.000) |
| Train on $L_S$, test on $L_T$ | 0.828 (±.003) | 0.257 (±.044) | 0.143 (±.080) | 0.086 (±.059) | 0.604 (±.004) |
| **Last-3-state encoding** | | | | | |
| Train and test on $L_S$ | 0.729 (±.000) | 0.750 (±.000) | 0.735 (±.000) | 0.279 (±.001) | 0.759 (±.000) |
| Train and test on $L_T$ | 0.782 (±.002) | 0.801 (±.002) | 0.785 (±.002) | 0.361 (±.006) | 0.763 (±.002) |
| Train on $L_S$, test on $L_T$ | 0.707 (±.003) | 0.541 (±.002) | 0.578 (±.002) | 0.128 (±.005) | 0.616 (±.002) |
| **Aggregation encoding** | | | | | |
| Train and test on $L_S$ | 0.723 (±.000) | 0.743 (±.001) | 0.730 (±.000) | 0.265 (±.001) | 0.749 (±.001) |
| Train and test on $L_T$ | 0.736 (±.002) | 0.759 (±.004) | 0.744 (±.002) | 0.236 (±.004) | 0.692 (±.003) |
| Train on $L_S$, test on $L_T$ | **0.831** (±.000) | 0.378 (±.011) | 0.351 (±.016) | **0.225** (±.009) | 0.614 (±.002) |

*Note.* Best results for training on $L_S$ and testing on $L_T$ are marked in bold

# Appendix C. Prediction performance for encoding activity and timestamp information of prefixes separately compared to encoding both jointly

We conducted an additional evaluation to test whether traditional, text-based contextualized embedding models can effectively encode activity and timestamp information about prefixes in a joint manner. For this evaluation, we constructed a baseline technique that jointly encodes the activity and timestamp information about prefixes using one of the four pre-trained contextualized embedding models, which we introduced in the setup of our evaluation. Specifically, the information about a prefix of the length $N$ is fed into the embedding models as "$a_1$; $t_1$; ...; $a_N$; $t_N$", where $a$ is an activity and $t$ is a timestamp in the format %d-%m-%Y %H:%M:%S.

The encoded prefixes are then mapped via an MLP model onto the values of the *in-time* prediction target. The MLP model has three fully connected blocks, each of which is followed by tanh activation, batch normalization, and dropout. In addition, the model incorporates residual connections after the second and third blocks using projection layers. The final output is passed through a sigmoid activation function. As with all other DL models used in our evaluation, the internal model parameters were optimized using the Adam optimizer (Kingma and Ba 2015).

Table 14 shows that our TL-based PPM technique achieves a clearly higher prediction performance when trained on $L_S$ and tested on $L_T$ compared to the baseline techniques using the four different pre-trained contextualized embedding models. This suggests that it is ineffective to use traditional, text-based, contextualized embedding models to jointly encode activity and timestamp information about prefixes in our TL setting.

Table 14: Prediction performance for our TL-based PPM technique that encodes activity and timestamp information about a trace separately compared to a baseline technique that encodes activity and timestamp information about a trace jointly. The baseline technique was tested using the four pre-trained contextualized embedding models of our setup for the inter-organizational use case (average and standard deviation over five runs)

| Embedding model | Precision | Recall | F1-score | MCC | $AUC_{ROC}$ |
|---|---|---|---|---|---|
| SEPARATE ENCODING OF ACTIVITY AND TIMESTAMP INFORMATION | | | | | |
| **Proposed approach** | | | | | |
| Train and test on $L_S$ | 0.735 (±.003) | 0.694 (±.003) | 0.708 (±.002) | 0.293 (±.007) | 0.730 (±.005) |
| Train and test on $L_T$ | 0.769 (±.012) | 0.769 (±.037) | 0.766 (±.024) | 0.330 (±.034) | 0.775 (±.018) |
| Train on $L_S$, test on $L_T$ | **0.739** (±.020) | **0.755** (±.050) | **0.724** (±.021) | **0.206** (±.029) | **0.711** (±.020) |
| JOINED ENCODING OF ACTIVITY AND TIMESTAMP INFORMATION | | | | | |
| **bert-base-cased** | | | | | |
| Train and test on $L_S$ | 0.725 (±.006) | 0.591 (±.031) | 0.616 (±.031) | 0.233 (±.008) | 0.680 (±.005) |
| Train and test on $L_T$ | 0.602 (±.000) | 0.776 (±.000) | 0.678 (±.000) | 0.000 (±.000) | 0.517 (±.020) |
| Train on $L_S$, test on $L_T$ | 0.688 (±.006) | 0.466 (±.035) | 0.499 (±.038) | 0.075 (±.016) | 0.566 (±.007) |
| **bert-base-uncased** | | | | | |
| Train and test on $L_S$ | 0.714 (±.008) | 0.666 (±.010) | 0.682 (±.006) | 0.238 (±.016) | 0.682 (±.006) |
| Train and test on $L_T$ | 0.602 (±.000) | 0.776 (±.000) | 0.678 (±.000) | 0.000 (±.000) | 0.526 (±.013) |
| Train on $L_S$, test on $L_T$ | 0.727 (±.025) | 0.397 (±.069) | 0.393 (±.101) | 0.109 (±.015) | 0.569 (±.004) |
| **all-MiniLM-L12-v2** | | | | | |
| Train and test on $L_S$ | 0.649 (±.086) | 0.668 (±.078) | 0.638 (±.029) | 0.131 (±.120) | 0.598 (±.096) |
| Train and test on $L_T$ | 0.620 (±.041) | 0.750 (±.057) | 0.676 (±.005) | 0.023 (±.052) | 0.537 (±.055) |
| Train on $L_S$, test on $L_T$ | 0.645 (±.040) | 0.713 (±.076) | 0.670 (±.028) | 0.036 (±.035) | 0.540 (±.044) |
| **all-mpnet-base-v2** | | | | | |
| Train and test on $L_S$ | 0.651 (±.087) | 0.688 (±.054) | 0.657 (±.021) | 0.140 (±.131) | 0.609 (±.102) |
| Train and test on $L_T$ | 0.656 (±.031) | 0.763 (±.014) | 0.684 (±.006) | 0.021 (±.021) | 0.594 (±.009) |
| Train on $L_S$, test on $L_T$ | 0.636 (±.033) | 0.772 (±.003) | 0.679 (±.002) | 0.006 (±.012) | 0.527 (±.036) |

*Note.* Best results for training on $L_S$ and testing on $L_T$ are marked in bold