SOLMformer - Incorporating Sequence and Observation Level Metadata for Categorical Time Series Modeling

Yamini Ananth [1] **Gregory Benton** [1] **Jingxing Fang** [1] **Jerry Cheung** [1] **Xu Chu** [1] **Cong Yu** [1]

Abstract

Sequential modeling, such as time series forecasting or language generation, traditionally uses the set of previous observations to predict future outcomes. However, it is often insufficient to exclusively model the sequence of observations. Not only may the observations themselves be dependent on metadata about the sequence, but predicting future metadata itself may be of interest. To address the shortcomings of sequential modeling alone, we propose SOLMformer, a multi-task approach that incorporates Sequence and Observation Level metadata into the transformer architecture as both inputs and as multi-task outputs. We evaluate SOLMformer on real-world process mining datasets and where it outperforms state of the art deep learning methods.

1. Introduction

Sequential modeling has been one of the massive successes of machine learning in recent years. However, often just a sequence of observations is insufficient to capture a true understanding of the generative process, and we need to both use and predict metadata either about the sequence or about the observations themselves. If we are predicting the future locations of a shipment, the past states will not provide the full picture. In order to make accurate predictions, we need to know what is being shipped, how long it has been in transit, and ideally a host of other salient attributes.

To better use and predict Sequence and Observation Level Metadata we propose SOLMformer. SOLMformer accounts for sequence level metadata like information about the contents of the shipment, and observation level metadata like how long the shipment was at a given location. In Fig. 1 we give an overview of the SOLMformer architecture. SOLMformer includes trainable models for embedding metadata. Each observation is concatenated with its corresponding metadata embedding, and the entire sequence of embeddings is prepended with an embedding of the sequence level metadata. To our knowledge, this is the first modification to the transformer architecture to allow for the modeling and prediction of sequences, sequence level metadata, and observation level metadata using a single unified model.

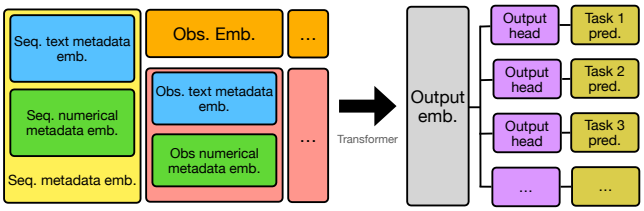

Figure 1. SOLMformer architecture - Sequence-level metadata embedding prepends composite embedding containing both an observation and its metadata. This sequence is passed through a Transformer, and the output embedding passes through task-specific output heads to produce multi-task predictions.

1.1. Process Prediction

In this work we focus SOLMformer on the task of predictive monitoring of *processes*. A process refers to a series of events that result in a desired output (van der Aalst, 2016). While process mining in general tends to look at large event logs to draw general insights into processes overall, predictive process monitoring aims to predict outcomes of each individual case in the event log (Di Francescomarino and Ghidini, 2022). In this setting an event log may consist of large quantities of distribution data for an organization, while an individual case may be a single shipment along with the sequence of activities it has gone through, such as being packed or stopping at intermediate locations. For individual cases), the event log may look different; however, mining a large event log can enable the predictive analysis of any individual case.

Predictive process monitoring poses unique challenges from a machine learning perspective. The sequence of observations that defines a case function as categorical time series,

[1]CeloAI, Celonis, New York. Correspondence to: Yamini Ananth <y.ananth@celonis.com>.

Accepted by the Structured Probabilistic Inference & Generative Modeling workshop of ICML 2024, Vienna, Austria.

with observations being unevenly spaced in time (van der Aalst et al., 2011). Since the time series defined by the observations is categorical we can also choose to view these sequences as traversals over a graph, as is typically done for process visualization (Dijkman et al., 2009; Reichert, 2013). In the graph view, nodes represent the possible activities, with edges being the connections between sequential activities.

Beyond just sequences of activities, each case is accompanied by its own metadata. For a shipment undergoing delivery, the metadata may be the contents of the shipment, the vendor, or the total time it will take to be delivered. Some metadata is known before the case begins and remains constant for the duration of the case (for example, the content of the shipment). Others, like the total time of the shipment, will not be known a priori and are of predictive interest. Together, these attributes are considered *sequence level metadata* as they apply to the sequence as a whole, rather than individual observations.

Just as each sequence has metadata, so too does each observation (or activity). Observation level metadata can include if the task was automated, when it was completed, and the duration of the activity. Some components of the observation metadata are known in advance and can be used to enrich the embeddings of each observation, while others are only discovered post hoc, but can be predicted as part of the sequential forecast.

We show that SOLMformer is well suited to metadata laden prediction problems and can improve the performance of predictive process modeling, while unifying pipelines to a single model that is trained end to end. Specifically, our work is laid out as follows: in Sec. 2 we show the connections of SOLMformer to recent work from both the machine learning and process mining communities. In Sec. 3 we describe in detail how SOLMformer works including training and prediction procedures. In Sec. 4 we show comparisons with related methods on process prediction tasks, demonstrating a clear improvement over baselines. In Sec. 4.3 we include an ablation study to show the contribution of metadata and multi-task learning on SOLMformer's performance. Finally, in Sec. 5 we outline a future path for work in this direction in both process mining and beyond.

2. Related Work

The transformer was introduced by Vaswani et al. (2017) for neural machine translations; it is a sequential model that uses a self-attention mechanism to learn representations between input and output. Given transformers operate on sequences of embeddings, past works have used concatenations of embeddings representing information at different levels, such as Cohan et al. (2020). Shalaby et al. (2022) introduce a metadata-aware transformer for performing multiple tasks in session-based recommender systems. Several transformer-based multi-task models have shown improvements in performance through the incorporation of auxiliary tasks (Ye and Xu, 2023; Mo et al., 2023; Bhattacharjee et al., 2022). SOLMformer implements a multi-task approach with hard parameter sharing, as we share all relevant information and only add separate prediction layers at the very end (Vandenhende et al., 2020).

Transformers and other deep learning architectures have seen increasing use in process mining. Evermann et al. (2017) demonstrate an RNN-based framework for process prediction. Camargo et al. (2019) propose another approach using parameter sharing to design a business process prediction method using a generative LSTM, and additionally encoded some sequence- and observation-level attributes into the model. This was an early instance of a deep learning based multi-task model, as it predicted next activities and time remaining in a given sequence. Khan et al. (2018) demonstrate that memory-aware neural networks (MANNs), which provide additional memory to RNNs, are a useful tool for process analysis. Another approach transforms temporal event log data into an image-like structure in order to use convolutional neural networks to predict the next activity in a sequence (Pasquadibisceglie et al., 2019).

In ProcessTransformer (PT), the event log was treated as tokens and a separate transformer model was trained for each dataset and each predictive task (Bukhsh et al., 2021). PT set the precedent for the usage of pooling and dropout layers in the prediction head of the Transformer. SOLMformer leverages a similarly straightforward architecture, but improves on performance by including metadata. SOLMformer also uses one model for all three prediction tasks, significantly reducing the training and evaluation cost.

The HiP Transformer leverages the Hierarchical-Transformer architecture (Wu et al., 2021) towards process mining, using an Event-level transformer (with event self-attention) to compute the next activity and time remaining in a case (Ni et al., 2023). The concept drift detection algorithm uses OPTICS clustering to identify sub-sequences with stable internal distributions, which are then passed to two more sub-sequence and case-level transformers for next state prediction, essentially stacking 3 transformers in layers. SOLMformer is comparatively efficient, using a single transformer with only extra overhead being the end-to-end training of the lightweight embedding models.

3. Methods

In a standard setting, transformers operate on a sequence of token embeddings. We first embed tokens via a lookup

table, then pass this sequence of embeddings through the transformer itself in order to generate an embedding for the next token in the sequence (Vaswani et al., 2017). With this final embedding we use a classification head to generate logits over the vocabulary for the next token. Using these basic building blocks, SOLMformer allows us to re-use much of the existing architecture and training methods, while allowing for the inclusion of metadata as both inputs and outputs of the model.

In a standard transformer we have,

$$e_i = \text{Emb}(t_i); \quad e_{T+1} = f(e_1, \ldots, e_T; \Theta). \tag{1}$$

Where Equation 1 represents first the embedding operation, and second the generation of a new embedding by passing the sequence through a transformer f with parameters Θ.

To generate the next embedding, e_{T+1}, SOLMformer modifies both the sequence itself, as well as each of the individual embeddings e_i. Equation 1 then becomes:

$$\mathbf{s} = \text{SequenceMetadataEmb}(\mathcal{S}; \Theta_{\mathcal{S}}) \tag{2}$$
$$\mathbf{o}_i = \text{ObservationMetadataEmb}(\mathcal{O}_i; \Theta_{\mathcal{O}}) \tag{3}$$
$$\tilde{e}_i = \text{Concat}\,(e_i, \mathbf{o}_i) \tag{4}$$
$$\tilde{e}_{T+1} = f(\mathbf{s}, \tilde{e}_1, \ldots, \tilde{e}_T; \Theta). \tag{5}$$

Equation 2 is the embedding of the sequence level metadata $\mathcal{S}$, and Equation 3 is the embedding of the observation level metadata $\mathcal{O}_i$. The details on these embedding functions are given below in Section 3.1. To emphasize that these are trainable embedding models we include their parameters $\Theta_{\mathcal{S}}$ and $\Theta_{\mathcal{O}}$. In Equation 5 we pass the complete set of information including all activities, their metadata, and the sequence level metadata to the transformer model, f.

By prepending the sequence with $\mathbf{s}$ we are mirroring the behavior of in-context learning through prompting (Dong et al., 2022). Since metadata is often typically semi-structured data (a mixture of text and numeric features) we manage the embedding procedure as a trainable component of the input and consider the first embedding in the sequence as a "metadata prompt" used to generate predictions.

3.1. Metadata Embedding Functions

Sequence and observation metadata take various modes, including high-dimensional categorical data, unstructured text, and ordinal data. To form the embedding functions for both sequence and observation metadata, we split metadata into text and numerical attributes. We handle these separately before re-joining them to form a final metadata embedding.

Numerical features are extracted from the metadata and passed through a small multi-layer perceptron (MLP) to be transformed into an embedding. Textual attributes are extracted and embedded using a lightweight sentence embedding model (Jiao et al., 2019). We take this text embedding and concatenate with the embedding of the numerical features to form our final metadata embedding vector. The generation of the embedding $\mathbf{s}$ from the metadata $\mathcal{S}$ is as follows:

$$\mathbf{s} = \text{Concat}(\text{MLP}(\mathcal{S}_{num}, \Theta_{\mathcal{S}}), \text{BERT}(\mathcal{S}_{text}, \Theta_{\mathcal{S}})). \tag{6}$$

With the same procedure being followed for producing $\mathbf{o}_i$ from $\mathcal{O}_i$.

Thus, from a set of text-based and numerical metadata, we can construct the metadata embeddings $\mathbf{s}$ and $\mathbf{o}_i$. Both the MLP for embedding numerical attributes and the sentence embedding model are included in the set of trainable parameters for the metadata embedding functions $\Theta_{\mathcal{S}}$ and $\Theta_{\mathcal{O}}$. These parameters are trained alongside the main transformer architecture end-to-end.

3.2. Constructing Input Sequences

A given observation t has an associated embedding $e_i \in \mathbb{R}^k$ constructed using a hash table embedding layer as is standard in tokenized language problems. For predictive process monitoring the tokens represent observed activities in the sequence rather than tokens produced through a tokenizer.

We use the procedure in Sec. 3.1 to produce both the sequence metadata embedding $\mathbf{s}$ and the observation metadata embeddings $\mathbf{o}_i$. The sequence metadata vector $\mathbf{s}$ has dimensionality $\|e_i\| + \|\mathbf{o}_i\|$ so it can be concatenated along the sequence axis as the first "prompting" token (see Fig. 1).

3.3. Predictions & Training

The output embedding e_{T+1} (Equation 5), is passed through various output heads to produce a set of predictions $\hat{y}$.

$$\hat{y} = \{\text{OutputHead}_i(\tilde{e}_{T+1}) \mid i = 1, 2, \ldots, Ntasks\} \tag{7}$$

Output heads are determined on a per-task basis. For these tasks, we use a single linear layer, although a two layer MLP did also provide similar results for next observation duration and time remaining in case predictions. The dimensionality of each prediction varies depending on the task (for instance, to predict the next activity we generate a vector of probabilities across the dimension of possible activities, whereas to predict the time remaining in the case we generate a real-valued scalar).

For a given model output $\hat{y}$, we also have the true value for each task, y. Then, to train i tasks simultaneously, we use a composite loss function (Equation 8) where the loss weights $\hat{w}$ are selected empirically for each specific dataset and the loss functions $\mathcal{L}_i$ are selected to suit each task. In this case, categorical cross-entropy loss is used for next activity prediction and L1 loss is used for the next observation duration

and time remaining in sequence.

$$\mathcal{L}(\hat{y}, y; \Theta) = \sum w_i \mathcal{L}_i(\hat{y}_i, y_i) \tag{8}$$

Manging loss is an ongoing challenge in multi-task learning. One scheme, random loss weighting, presented by Lin et al. (2022), had no significant effect on SOLMformer performance. It is possible that other dynamic gradient balancing techniques, such as GradNorm or Conflict-Averse gradient descent may lead to further improvement, and this is an open area for future study (Chen et al., 2018; Liu et al., 2024).

Details of training and setup are included in Appendix B. For the focus use case of predictive process monitoring, auxiliary numerical features are added to the observation-level numerical metadata as described in Appendix A.2.

4. Results

4.1. Datasets, Baselines, & Metrics

We selected 5 open-source datasets from the Business Process Intelligence Challenge (BPIC) that contain event logs rich with metadata to demonstrate the performance of SOLMformer. Information describing the datasets including their metadata can be found in Appendix A.1. We include 6 baselines across the tasks, prioritizing deep learning and multi-task models.

We include LSTM, GRU, MANN, and CNN-based models to highlight multiple deep learning architectures and their performance on these tasks (Tax et al., 2017; Khan et al., 2018; Hinkka et al., 2018; Pasquadibisceglie et al., 2019). We also select ProcessTransformer as it is the landmark implementation of the transformer architecture for process mining (Bukhsh et al., 2021). Also, the HiP Transformer is the only improvement to the architecture specifically aimed at process mining use cases to our knowledge (Ni et al., 2023).

For models with open-source implementations we make best efforts to train as specified by the authors (Tax et al., 2017; Bukhsh et al., 2021). Where we were unable to retrieve the baselines, we use results as benchmarked by Rama-Maneiro et al. (2020).

We select the evaluation metric of accuracy for next activity prediction, a classic classification problem. Weighted F1 score and Mathews Correlation Coefficient are sometimes also reported, but are typically aligned with accuracy and provide limited additional information in the predictive process modeling context (Matthews, 1975; Rama-Maneiro et al., 2020). We select mean absolute error (MAE) as the evaluation metric for next observation duration and time remaining in case predictions. This does not overpenalize variable observations, since time between two observations in a sequence can be quite large (Willmott and Matsuura, 2005).

4.2. SOLMformer on predictive tasks

As shown in Figure 2 (exact values in Appendix 5) we find SOLMformer improves performance in 4 out of 5 datasets on next activity prediction, with other transformer-based implementations slightly outperforming on the BPI2013 dataset. Notably, BPI2013 contains no sequence metadata or numerical observation metadata. All other datasets contain at least one sequence metadata attribute.

For the next observation duration, we are predicting the expected time until the next observation occurs. Over a sequence, the duration of each activity is not monotonic and varies significantly based on activity type and context. From sequence to sequence, the same activity can vary in duration depending on outside circumstances which are not documented in the sequence or its metadata.

SOLMformer outperforms baselines on 4 datasets in predicting the next activity duration. For BPI2012, it performs similarly to ProcessTransformer, an implementation from Transformer that does not incorporate case-level metadata or non-numerical observational metadata (Bukhsh et al., 2021). SOLMformer outperforms several baselines in predicting the next observation duration and time remaining in a sequence. However, ProcessTransformer lags behind in small datasets or datasets with minimal data (BPI2012/2013). For the time remaining in the case, the HiP Transformer slightly outperforms SOLMformer for BPI2013 and SOLM performs squarely between ProcessTransformer and HiP Transformer for BPI2012.

4.3. Breaking apart SOLMformer

We include the results of an ablation study in this section to illustrate the impact of including case- and observation-level metadata and multi-task learning on multiple prediction tasks. We present a baseline method and two variants of SOLMformer (architectures illustrated in Figures 3, 4, 5). Architectures for the baseline, S-former, and SOLMformer-1T variants are included in Figures 3, 4, 5.

1. **Baseline** - A transformer model that uses only sequences of activities as input and predicts only next activities. The baseline does not include any metadata as either inputs or outputs.
2. **S-former** Uses activities and case level metadata to predict next activities only (since observation level metadata is excluded).
3. **SOLMformer-1T** Uses case and event level metadata, but only performs 1 task at a time. A different model must be trained for each task.

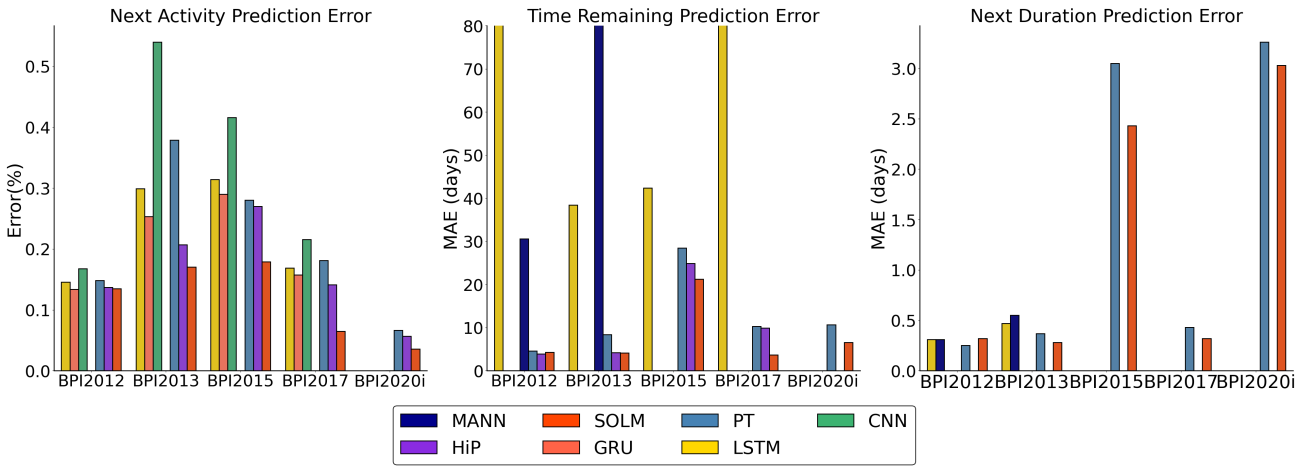

Figure 2. For next activity prediction, on BPI2012 SOLMformer performs comparably with HiP-Transformer and GRU; on the other tasks SOLMformer outperforms all competing methods. Results are omitted for models that do not produce predictions for a given problem, or where results are not available. For next duration and time remaining, SOLMformer outperforms baselines. Beyond being able to predict metadata, the inclusion of metadata into activity prediction improves performance.

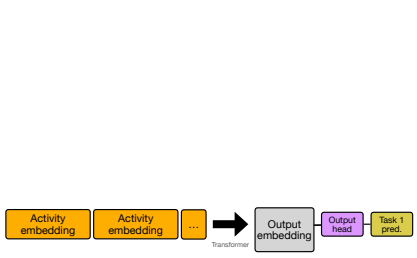

Figure 3. Baseline method has no sequence or observational metadata, and predicts next activity only.

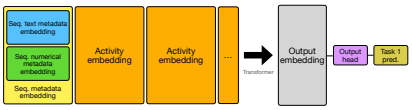

Figure 4. S-former only includes sequence level metadata, and predicts the next activity only.

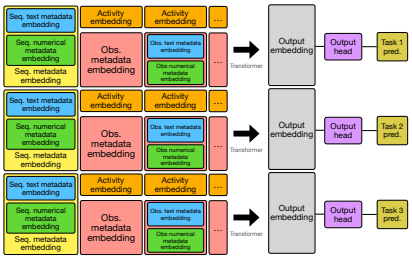

Figure 5. SOLMformer-1T includes sequence and observational metadata, and performs only 1 task at a time.

We can observe in Table 1 the general trend that adding case-level metadata significantly improves performance of next-activity prediction (Baseline vs S-former), particularly for BPI2012 and BPI2017. Adding event-level metadata leads to some further improvement as well, particularly for datasets rich in event metadata (BPI2017). Finally, multi-task learning improves performance for the next activity prediction task in datasets BPI2012, 2017, 2020i.

Something else to note is that SOLM-1T outperforms ProcessTransformer on 4 out of 5 datasets for both next observation duration and time remaining in case (BPI2012 being the exception). ProcessTransformer is a 1-task model, so SOLMformer's incorporation of metadata makes a noticeable difference in performance.

Datasets	Baseline	S-former	SOLM-1T	SOLM
BPI2012	53.05	1.37	**87.10**	86.50
BPI2013	73.00	N/A	63.34	**76.15**
BPI2015	59.95	63.43	66.30	**73.10**
BPI2017	87.07	87.32	92.46	**93.53**
BPI2020i	87.01	86.73	92.89	**96.42**

Table 1. Ablation Study: Next Activity Prediction - Accuracy(%). We find that the inclusion of metadata significantly improves accuracy. Importantly, SOLMformer performs close to or better than SOLM-1T, which is much more flexible as each task is trained independently. This result suggests effective mutual information between the tasks, and is an open area for further investigation. BPI2013 lacks case-level metadata so S-former results are omitted.

Datasets	ND		TR	
	SOLM-1T	SOLM	SOLM-1T	SOLM
BPI2012	0.44	**0.32**	5.01	**4.31**
BPI2013	**0.29**	0.29	**4.09**	4.10
BPI2015	**2.40**	2.43	**20.99**	21.20
BPI2017	**0.30**	0.32	**4.04**	4.12
BPI2020i	**2.28**	3.03	**6.45**	6.56

Table 2. Ablation Study: Next observation Duration (ND) and Time Remaining in sequence (TR) - MAE (days). SOLM-1T performs marginally better than SOLM for all datasets except BPI2012, indicating that performance on next activity prediction is boosted by auxiliary tasks such as ND and TR but not vice versa.

5. Discussion & Future Work

We have presented SOLMformer, an adaptation of transformers to include sequence and object level metadata both as inputs and in prediction. While we have specifically focused on process mining, SOLMformer is a very general framework that applies to transformers in a broad range of domains. In particular, as transformers see more use for time series problems in domains like finance and climatology which contain significant amounts of metadata, we see SOLMformer as a simple but natural extension to the transformer architecture.

References

Deblina Bhattacharjee, Tong Zhang, Sabine Süsstrunk, and Mathieu Salzmann. Mult: An end-to-end multitask learning transformer, 2022.

Zainab Abbas Bukhsh, Ayesha Saeed, and Remco M. Dijkman. Process transformer: Predictive business process monitoring with transformer network. *arXiv preprint arXiv:2104.00721*, 2021.

Manuel Camargo, Marlon Dumas, and Oscar González-Rojas. Learning accurate lstm models of business processes. In *Business Process Management: 17th International Conference, BPM 2019, Vienna, Austria, September 1–6, 2019, Proceedings*, page 286–302, Berlin, Heidelberg, 2019. Springer-Verlag. ISBN 978-3-030-26618-9. doi: 10.1007/978-3-030-26619-6_19.

Zhao Chen, Vijay Badrinarayanan, Chen-Yu Lee, and Andrew Rabinovich. Gradnorm: Gradient normalization for adaptive loss balancing in deep multitask networks, 2018.

Arman Cohan, Sergey Feldman, Iz Beltagy, Doug Downey, and Daniel S Weld. Specter: Document-level representation learning using citation-informed transformers. *arXiv preprint arXiv:2004.07180*, 2020.

Chiara Di Francescomarino and Chiara Ghidini. *Predictive Process Monitoring*, pages 320–346. Springer International Publishing, Cham, 2022. ISBN 978-3-031-08848-3. doi: 10.1007/978-3-031-08848-3_10. URL https://doi.org/10.1007/978-3-031-08848-3_10.

Remco Dijkman, Marlon Dumas, and Luciano García-Bañuelos. Graph matching algorithms for business process model similarity search. In Umeshwar Dayal, Johann Eder, Jana Koehler, and Hajo A. Reijers, editors, *Business Process Management*, pages 48–63, Berlin, Heidelberg, 2009. Springer Berlin Heidelberg. ISBN 978-3-642-03848-8.

Qingxiu Dong, Lei Li, Damai Dai, Ce Zheng, Zhiyong Wu, Baobao Chang, Xu Sun, Jingjing Xu, and Zhifang Sui. A survey on in-context learning. *arXiv preprint arXiv:2301.00234*, 2022.

Joerg Evermann, Jochen R. Rehse, and Peter Fettke. A deep learning approach for predicting process behaviour at runtime. In Marlon Dumas and Marcello Fantinato, editors, *International Conference on Business Process Management*, volume 281, pages 327–338, Cham, Switzerland, 2017. Springer.

Matti Hinkka, Timo Lehto, Keijo Heljanko, and Adrian Jung. Classifying process instances using recurrent neural networks. In Florian Daniel, Quan Z. Sheng, Hamid Motahari, and Eds., editors, *International Conference on Business Process Management*, volume 342, pages 313–324, Cham, Switzerland, 2018. Springer.

Xiaoqi Jiao, Yichun Yin, Lifeng Shang, Xin Jiang, Xiao Chen, Linlin Li, Fang Wang, and Qun Liu. Tinybert: Distilling BERT for natural language understanding. *CoRR*, abs/1909.10351, 2019. URL http://arxiv.org/abs/1909.10351.

Aaqib Khan, Huynh Le, Kieu Do, Truyen Tran, Aditya Ghose, Hoa Dam, and Ramasuri Sindhgatta. Memory augmented neural networks for predictive process analytics. *arXiv preprint arXiv:1802.00938*, 2018.

Baijiong Lin, Feiyang Ye, Yu Zhang, and Ivor W. Tsang. Reasonable effectiveness of random weighting: A litmus test for multi-task learning, 2022.

Bo Liu, Xingchao Liu, Xiaojie Jin, Peter Stone, and Qiang Liu. Conflict-averse gradient descent for multi-task learning, 2024.

Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization, 2019.

B.W. Matthews. Comparison of the predicted and observed secondary structure of t4 phage lysozyme. *Biochimica et Biophysica Acta (BBA) - Protein*

Structure, 405(2):442–451, 1975. ISSN 0005-2795. doi: https://doi.org/10.1016/0005-2795(75)90109-9. URL https://www.sciencedirect.com/science/article/pii/0005279575901099.

Ying Mo, Hongyin Tang, Jiahao Liu, Qifan Wang, Zenglin Xu, Jingang Wang, Wei Wu, and Zhoujun Li. Multi-task transformer with relation-attention and type-attention for named entity recognition, 2023.

Nicolò Navarin, Beatrice Vincenzi, Mirko Polato, and Alessandro Sperduti. Lstm networks for data-aware remaining time prediction of business process instances, 2017.

An Nguyen, Srijeet Chatterjee, Sven Weinzierl, Leo Schwinn, Martin Matzner, and Bjoern Eskofier. Time matters: Time-aware lstms for predictive business process monitoring, 2020.

Weijian Ni, Gang Zhao, Tong Liu, Qingtian Zeng, and Xingzong Xu. Predictive business process monitoring approach based on hierarchical transformer. *Electronics*, 12(6):1273, 2023. doi: 10.3390/electronics12061273. URL https://doi.org/10.3390/electronics12061273.

Vito Pasquadibisceglie, Annalisa Appice, Giovanni Castellano, and Donato Malerba. Using convolutional neural networks for predictive process analytics. In *Proceedings of the IEEE International Conference on Process Mining*, pages 129–136, Aachen, Germany, 2019.

Efrén Rama-Maneiro, Juan Carlos Vidal, and Manuel Lama. Deep learning for predictive business process monitoring: Review and benchmark. *CoRR*, abs/2009.13251, 2020. URL https://arxiv.org/abs/2009.13251.

Manfred Reichert. Visualizing large business process models: Challenges, techniques, applications. In Marcello La Rosa and Pnina Soffer, editors, *Business Process Management Workshops*, pages 725–736, Berlin, Heidelberg, 2013. Springer Berlin Heidelberg. ISBN 978-3-642-36285-9.

Walid Shalaby, Sejoon Oh, Amir Afsharinejad, Srijan Kumar, and Xiquan Cui. M2trec: Metadata-aware multi-task transformer for large-scale and cold-start free session-based recommendations. In *Proceedings of the 16th ACM Conference on Recommender Systems*, RecSys '22. ACM, September 2022. doi: 10.1145/3523227.3551477. URL http://dx.doi.org/10.1145/3523227.3551477.

Ward Steeman. Bpi challenge 2013, incidents, 2013. URL https://data.4tu.nl/articles/_/12693914/1.

Niek Tax, Ivan Verenich, Marcello La Rosa, and Marlon Dumas. Predictive business process monitoring with lstm neural networks. In *Proc. of CAiSE*. Springer, 2017.

Wil van der Aalst. *Process Mining: Data Science in Action*. Springer Berlin, Heidelberg, 2 edition, 2016. ISBN 978-3-662-49850-7. doi: 10.1007/978-3-662-49851-4. URL https://doi.org/10.1007/978-3-662-49851-4. Springer-Verlag Berlin Heidelberg 2016.

W.M.P. van der Aalst, M.H. Schonenberg, and M. Song. Time prediction based on process mining. *Information Systems*, 36(2):450–475, 2011. ISSN 0306-4379. doi: https://doi.org/10.1016/j.is.2010.09.001. URL https://www.sciencedirect.com/science/article/pii/S0306437910000864. Special Issue: Semantic Integration of Data, Multimedia, and Services.

B.F. (Boudewijn) van Dongen. Bpi challenge 2015, 2015. URL https://data.4tu.nl/collections/_/5065424/1.

Boudewijn van Dongen. Bpi challenge 2012, 2012. URL https://data.4tu.nl/articles/_/12689204/1.

Boudewijn van Dongen. Bpi challenge 2017, 2017. URL https://data.4tu.nl/articles/_/12696884/1.

Boudewijn van Dongen. Bpi challenge 2020, 2020. URL https://data.4tu.nl/collections/_/5065541/1.

Simon Vandenhende, Stamatios Georgoulis, Marc Proesmans, Dengxin Dai, and Luc Van Gool. Revisiting multi-task learning in the deep learning era. *CoRR*, abs/2004.13379, 2020. URL https://arxiv.org/abs/2004.13379.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 30, 2017.

Cort J Willmott and Kenji Matsuura. Advantages of the mean absolute error (mae) over the root mean square error (rmse) in assessing average model performance. *Climate research*, 30(1):79–82, 2005.

Chuhan Wu, Fangzhao Wu, Tao Qi, and Yongfeng Huang. Hi-transformer: Hierarchical interactive transformer for efficient and effective long document modeling. *CoRR*, abs/2106.01040, 2021. URL https://arxiv.org/abs/2106.01040.

Hanrong Ye and Dan Xu. Taskprompter: Spatial-channel multi-task prompting for dense scene understanding. In *ICLR*, 2023.

A. Data

A.1. Datasets

The following 5 real-world datasets were selected for the benchmarking of SOLMformer. BPI datasets are important benchmarks in the process mining community and have been benchmarked extensively (Tax et al., 2017; Bukhsh et al., 2021; Rama-Maneiro et al., 2020; Hinkka et al., 2018; Pasquadibisceglie et al., 2019; Ni et al., 2023). Summary statistics regarding the datasets are available in table A.1, and their metadata is summarized in table A.1

1. **BPI2012** is a Holland financial institution loan applying process during the period from October 2011 to March 2012 (van Dongen, 2012).
2. **BPI2013** documents the process related to the Volvo back office management system during the period from April 2010 to May 2012 (Steeman, 2013).
3. **BPI2015** is the Holland building permit applying process during the period from April 2014 to September 2014 (van Dongen, 2015).
4. **BPI2017** is the Holland financial institution loan applying process during the period from January 2016 to December 2016 (van Dongen, 2017).
5. **BPI2020i** captures the permits, declaration documents, and administrative approvals necessary for travel expense claims at a university (van Dongen, 2020).

Table 3. Datasets

Dataset	Num. Cases	Num. Observations	Num. Activities	Mean Num. Observations per Case	Max Num. Activities per Case	Mean Case Duration (days)
BPI2012	13,087	262,200	24	20	175	8.6
BPI2013	586	6,499	13	11.4	123	11.9
BPI2015	1,199	27,409	38	22.8	61	95.9
BPI2017	31,509	15,214	26	36	180	18
BPI2020i	6,449	72,151	34	11.2	27	84.15

A.2. Data Processing

All sequences and observations were included with no particular preprocessing or treatment. No additional context regarding the dataset, such as process models, were used.

For all observations, three numerical features were included by default: time since beginning of case, duration of previous activity, and duration of second to last activity. These temporal features are shown to have a positive effect on process prediction in past works (Bukhsh et al., 2021; Nguyen et al., 2020; Navarin et al., 2017).

B. Experiment Details

We use a train/validation/test split of 80/10/10% and batch sizes of 128 for BPI2012/2017/2020i or 24 for BPI 2013/2015. we train for 25 epochs using a cosine annealing learning rate schedule with a minimum of 1e-4 and a maximum of 1e-1. The model with the best performance on the validation set was used for testing. We used 4 attention heads with 2 layers, an activity embedding dimension of 256, an observation text metadata embedding dimension of 128, and a observation ordinal metadata embedding dimension of 128 (total input dimension being 512), with a hidden dimension of 1024. We used an AdamW optimizer with a weight decay of 1e-4 (Loshchilov and Hutter, 2019). Data processing, training, evaluation, and model code are available to view in this repository.

Experiments were performed on 1x NVIDIA A10 (24 GB VRAM 30 vCPUs 200 GiB RAM) for BPI2012, 2013, 2015, 2020i and 1x NVIDIA A6000 (48 GB VRAM, 14 vCPUs, 100 GiB RAM) for BPI2017.

Table 4. Dataset Metadata

Dataset	Sequence-level metadata		Observation-level metadata	
	Numerical	Text	Numerical	Text
BPI2012	None	'AMOUNT REQ'	None	'org:resource', 'lifecycle:transition'
BPI2013	None	None	None	'product', 'org:group', 'lifecycle:transition', 'resource country', 'org:role', 'organization involved', 'impact', 'organization country', 'org:resource'
BPI2015	'SUMleges'	'caseProcedure', 'caseStatus', 'parts', 'requestComplete', 'IDofConceptCase', 'concept:name', 'termName', 'IncludesSubCases', 'landRegisterID', 'ResponsibleActor', 'lastPhase', 'caseType'	None	'action code', 'monitoringResource', 'question', 'activityNameNL', 'dateFinished', 'activityNameEN', 'org:resource', 'lifecycle:transition', 'dateStop'
BPI2017	'RequestedAmount'	'ApplicationType', 'LoanGoal'	'MonthlyCost', 'FirstWithdrawalAmount', 'OfferedAmount'	'Selected', 'Action', 'EventOrigin', 'EventID', 'Accepted', 'org:resource', 'lifecycle:transition', 'OfferID'
BPI2020i	'AdjustedAmount', 'Permit RequestedBudget', 'OriginalAmount', 'RequestedAmount', 'Amount'	'Permit BudgetNumber', 'Permit ActivityNumber', 'Permit ProjectNumber', 'DeclarationNumber', 'Permit TaskNumber', 'concept:name', 'Permit OrganizationalEntity', 'Permit travel permit number', 'travel permit number', 'id', 'BudgetNumber', 'Permit ID'	-	'id', 'org:resource', 'org:role'

Table 5. Next Activity Prediction - Accuracy (%)

Dataset	LSTM	GRU	CNN	PT	HiP	SOLM
BPI2012	85.46	86.65	83.25	85.20	86.30	**86.75**
BPI2013	70.09	74.69	46.03	62.11	**79.33**	76.15
BPI2015	68.58	71.02	58.43	71.98	73.02	**73.10**
BPI2017	83.15	84.25	78.45	81.88	85.88	**93.53**
BPI2020i	-	-	-	93.35	94.33	**96.42**

Table 6. Next observation duration prediction - MAE (days)

Dataset	**LSTM**	**MANN**	**PT**	**SOLM**
BPI2012	0.31	0.31	**0.25**	0.32
BPI2013	0.47	0.55	0.37	**0.28**
BPI2015	-	-	3.05	**2.43**
BPI2017	-	-	0.43	**0.32**
BPI2020i	-	-	3.26	**3.03**

Table 7. Time remaining in case prediction - MAE (days)

Dataset	**LSTM**	**MANN**	**PT**	**Hi-P**	**SOLM**
BPI2012	330.61	30.56	4.60	**3.89**	4.31
BPI2013	38.41	260.64	8.36	4.17	4.10
BPI2015	42.36	-	28.42	24.88	21.20
BPI2017	127.80	-	10.28	9.93	**3.66**
BPI2020i	-	-	10.68	-	**6.56**

In order to train the model reasonably given the different scales of task losses, we log-scale the observation duration and time remaining in sequence-related observation numerical metadata features. Losses for the next observation duration and time remaining in sequence tasks are calculated in log-space during training. Model outputs for these tasks are scaled back to linear space in order to make predictions and in the reported metrics.

C. Results

Results containing exact results corresponding to Fig. 2 are included in Tables 5, 6, 7. ”-” denotes that no baseline assessment was possible.