

# Structural Positional Encoding for knowledge integration in transformer-based medical process monitoring

Marco Dossena<sup>1</sup>, Christopher Irwin<sup>1,\*</sup>, Giorgio Leonardi<sup>1,2</sup> and Stefania Montani<sup>1,2</sup>

<sup>1</sup>*DISIT, Computer Science Institute, University of Piemonte Orientale, Alessandria, Italy*

<sup>2</sup>*Laboratorio Integrato di Intelligenza Artificiale e Informatica in Medicina DAIRI; Azienda Ospedaliera SS. Antonio e Biagio e Cesare Arrigo, Alessandria*

## Abstract

Predictive process monitoring is a process mining task aimed at forecasting information about a running process trace, such as the most correct next activity to be executed. In medical domains, predictive process monitoring can provide valuable decision support in atypical and nontrivial situations. In this paper, we propose a predictive process monitoring approach relying on the use of a *Transformer*, a deep learning architecture based on the attention mechanism, that we are testing in the domain of stroke management. A major contribution of our work lies in the incorporation of ontological domain-specific knowledge, carried out through a graph positional encoding technique. The paper also presents and discusses the first encouraging experimental result we are collecting.

## Keywords

Medical process monitoring, Transformers, Graphs, Positional encoding

## 1. Introduction

The diffusion of advanced medical information systems is progressively enabling the automatic collection of patients' *traces*, i.e., the sequences of activities executed on patients during the care procedures implemented at a hospital organization [1]. Patients' traces are a valuable source of information for several analyses and investigations, within the field of process mining [2]. Among the various available process mining techniques, predictive process monitoring [3, 4] is of particular interest. Predictive process monitoring aims at forecasting relevant information about a running process trace; specifically, it exploits the already logged traces to make predictions about the running trace completion, such as suggesting the next activity to be executed, or estimating the remaining time/cost/resources required to completion itself.

In the medical field, predictive process monitoring can support better time and resource allocation; moreover, and most importantly, it can support decision-making in nontrivial cases. Indeed, despite the availability of clinical guidelines, it is worth noting that guidelines are ideal processes, designed for ideal patients, and meant to be applied in an ideal setting, where all the

---


*HC@AIxIA 23: 2nd AIxIA Workshop on Artificial Intelligence For Healthcare, November 06–09, 2023, Rome, IT*

\*Corresponding author.

✉ marco.dossena@uniupo.it (M. Dossena); christopher.irwin@uniupo.it (C. Irwin); giorgio.leonardi@uniupo.it (G. Leonardi); stefania.montani@uniupo.it (S. Montani)



© 2023 Copyright for this paper by its authors. Use permitted under Creative Commons License Attribution 4.0 International (CC BY 4.0).

 CEUR Workshop Proceedings (CEUR-WS.org)

needed resources are always available [5]. In reality, this is often not the case: local resource constraints may prevent from the (timely) execution of the scheduled guidelines activities. Moreover, real patients may be atypical, for instance, due to the presence of comorbidities or rare disease variants. Finally, physicians may lack the necessary background knowledge to correctly interpret and apply guidelines in complex cases. Predictive process monitoring, by suggesting the next activity to be executed, can thus provide very helpful advice in these nontrivial situations.

Stemming from the considerations discussed above, in this paper we propose a predictive process monitoring approach for medical process traces; according to the most recent literature advances, we rely on the exploitation of a *Transformer*, a deep learning architecture based on the self-attention mechanism [6], which has already demonstrated its utility in modeling similar tasks, as exemplified in [7]. A major contribution of our work lies in the introduction of a technique that incorporates domain-specific knowledge using an ontology. This augmentation is carried out through a graph positional encoding technique, enhancing the accuracy of our model.

While the approach is general, we were motivated by a specific application domain, that we selected for our experiments, namely the one of stroke management. Indeed, in this field, predictive process monitoring can provide a valuable contribution to medical decision-making. To mention a few examples, it can help in the choice of the correct thrombolytic drug, by considering how indications and contraindications of the available medications were balanced in past traces [5], or it can help in deciding whether to confirm the presence of subarachnoid hemorrhage by taking into account pros and cons of a lumbar puncture [8], always referring to already completed traces. Indeed, the first experiments we performed, which are described in the following, have shown encouraging results.

The paper is organized as follows: Section 2 introduces some related work. Section 3 presents the deep learning architecture we exploit. Section 4 illustrates our experimental setting and results, while section 5 is devoted to conclusions.

## 2. Related work

Predictive process monitoring has been afforded by different types of machine learning techniques, such as Transition Systems [9], Hidden Markov Models [10] and Support Vector Machines [11]. In more recent approaches, however, deep learning architectures are being applied, and nowadays represent the state of the art in the field.

Different deep learning architectures have been proposed to deal with the task of next activity prediction. The approach in [12], for instance, uses Autoencoders [13]. In predictive monitoring, Autoencoders can successfully reduce the number of attributes of the input activities; however, they are unable to treat long (temporal) dependencies between activities of a trace. Convolutional Neural Networks (CNNs) [14] have been applied in [15], where sequential data in process traces are treated as a one-dimensional grid.

Due to the sequential nature of traces, however, Recurrent Neural Networks (RNNs) [16] probably represent the most natural solution. In fact, RNNs can capture longer dependencies between the activities of a trace, while in a CNN an activity only depends on the  $k$  most recent

activities, where  $k$  is the size of the kernel. Within RNNs, Long-Short Term Memory (LSTM) networks [17] represent a particularly performant approach. Indeed, LSTM can potentially learn the complex dynamics within the temporal ordering of input sequences; therefore, they are well suited to manage the sequential data of process activity logs. They can also manage long-distance dependencies between activities, since they implement a long-term memory where the information flows from cell to cell with minimal variations, keeping certain aspects constant during the processing of all inputs. The works in [18, 19, 20, 21], for instance, rely on LSTMs to predict the next activity of a running case.

Similarly to LSTMs, Gated Recurrent Networks (GRUs) [22] also create paths through time that allow the gradients to flow deeper in the sequence than in basic RNNs; with respect to LSTMs, they have few parameters. GRUs are exploited in, e.g., [23] for predictive monitoring.

In [24], instead, a Memory Augmented Neural Network (MANN) is proposed. MANNs allow to learn even longer dependencies; training is more expensive, but MANNs are suitable for very long traces, or when cycles of the same activity may lead LSTMs or GRUs to forget/ignore activities located at the beginning of the trace.

The approaches which most closely relate to ours are in [25, 7], which rely on a Transformer, an architecture that substitutes the recurrence by the attention mechanism [6]. In particular, these works adopt multi-head attention, i.e., they perform the attention operation over different parts of the sequence simultaneously, and, instead of training an encoder-decoder architecture as in [6], they only rely on the decoding part. As for the incorporation of external domain knowledge in the form of graphs, this approach has been explored in a few works, such as [26, 27, 28], aiming to enrich sequence-based examples with geometric information derived from a graph structure. An example of this is the combination of protein structures with the geometric arrangement of molecules. The knowledge-driven approach has also been used in several works involving pre-trained transformers (e.g. BERT) in order to align the model to a specific knowledge domain by means of ontologies and knowledge-graphs [29, 30]. We are not aware of previous works that incorporate structural positional encoding of domain knowledge into predictive process monitoring.

### 3. Methodologies

In the following sections, we are going to present the base architecture of the transformer model we are using. Subsequently, we will describe the method that enables leveraging the ontology of the activities using Structural Positional Encoding (SPE), which is based on the Laplacian eigenvalue encoding technique [27].

#### 3.1. Transformer for PPM

Our model architecture is based on the transformer decoder (as in [25, 7]). The model input is a sequence  $S = \{a_1, \dots, a_i, \dots, a_n\}$  (representing a trace) where every element of the sequence represents an activity coming from a set  $A$  of possible activities. Figure 1 depicts the model structure. Below, we describe the building blocks of the model and how the inputs are processed.

**Embedding layer:** it takes  $S$  as input and returns a vector of dimension  $\mathbb{R}^d$  for each activity  $a_i$ . At this point, our samples are in the form  $X \in \mathbb{R}^{n \times d}$ .

**Positional encoding:** this layer is responsible for augmenting each embedding representing an activity with information about its position. In this paper, we tested two versions of positional encoding. The first version (PE henceforth), described in [6], uses a sine and a cosine function to obtain a representation that respects the order of the activities within the single sequence. The second version is Structural Positional Encoding (SPE henceforth), which embeds knowledge from a graph  $G$  (in our case, the graph is an ontology of the activities, and is described in section 3.2). Positional encoding generates a vector for each activity in a sequence that is added to the original embedding as follows:

$$X = \begin{cases} X + PE(X) & \text{for PE method} \\ X + SPE(X, G) & \text{for SPE method} \end{cases}$$

**Multi-head attention layer:** this component enables the model to selectively attend to different parts of a sequence and find long-range dependencies. To this end, the self-attention layer [6] generates three representation  $Q, K, V$  for every activity in the sequence  $S$ . Subsequently, it applies the scaled dot-product attention:

$$H = softmax\left(\frac{QK^T}{\sqrt{d}}\right)V$$

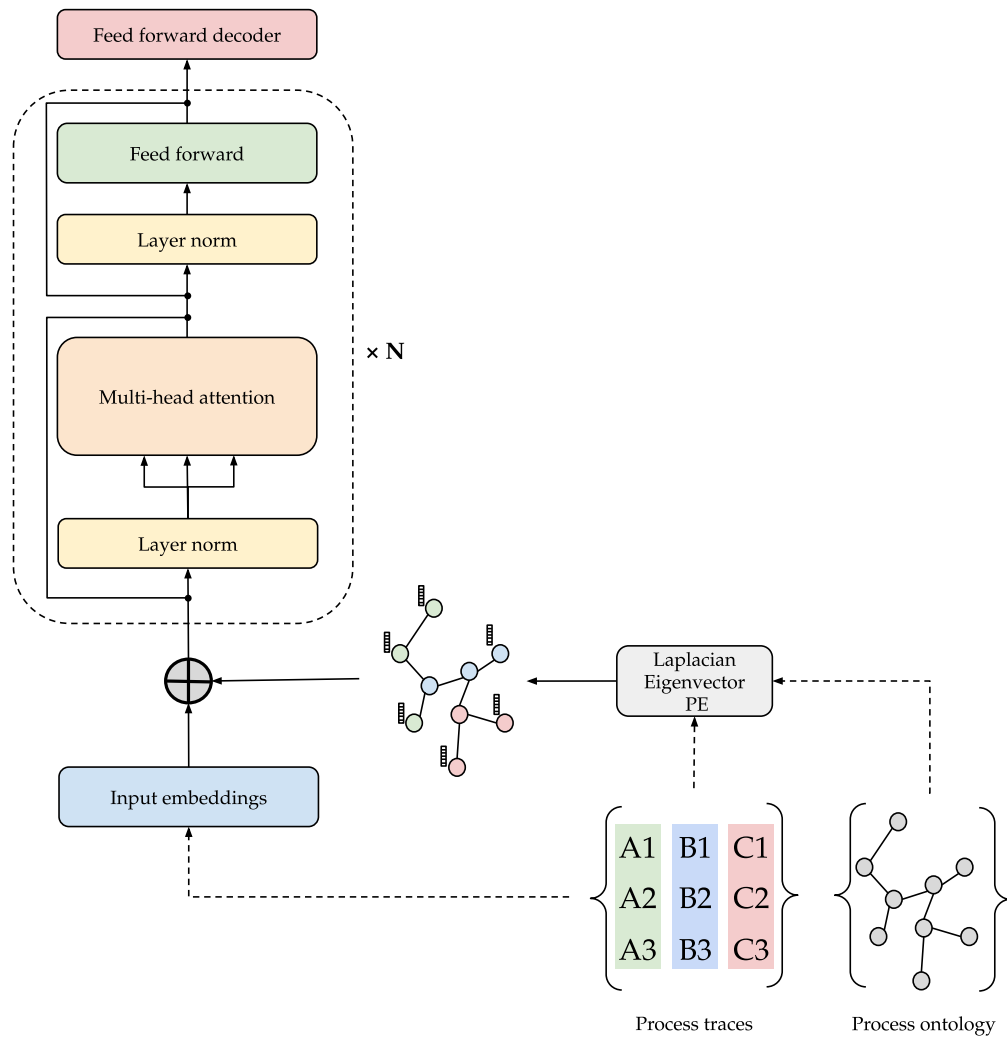
This operation is carried out in parallel on multiple attention heads in order to learn different types of dependencies. The resulting representations are concatenated to produce the final output. Note that, since the task is to predict the next token in the sequence (in our application, the next activity in the trace), during the self-attention process we apply a mask so that an activity can only attend to previous ones, and ignore those that follow it. The output is finally added to the input  $X$  using a skip connection.

**Layer norm:** this normalisation layer is used to mitigate vanishing or exploding gradients phenomena during the model training phase. The process ensures that the activations are centered around zero mean and unit variance making the optimization more stable [31]. In our implementation, we resorted to the *Pre-LN* configuration where the layer-norm is put inside the residual blocks which improves the model convergence [32].

Lastly,  $X$  is subsequently passed through 2 fully connected layers, where the final layer acts as a decoder, projecting the representations into a dimensionality equivalent to the number of potential activities.

### 3.2. Structural Positional Encoding

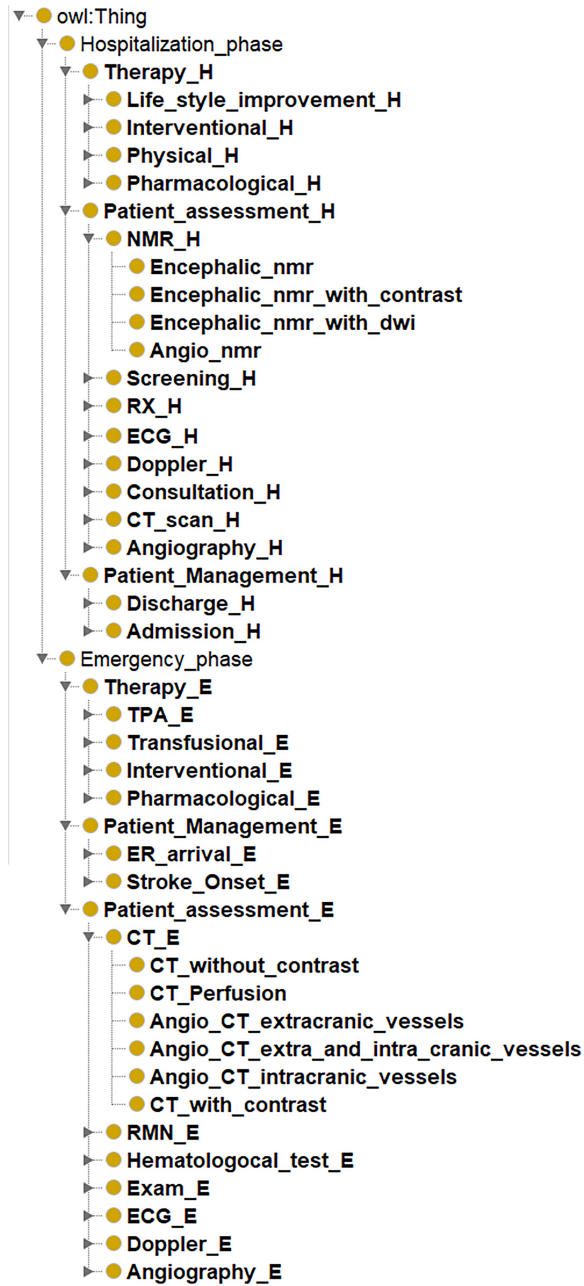
Structural Positional Encoding (SPE) is based on a graph representing an ontology that encodes the relations between activities. This ontology allows for the integration of domain knowledge into the AI model, in order to improve the prediction capabilities of the system. In particular,



**Figure 1:** The model architecture. The left side of the figure represents the Transformer-decoder model, the right side shows the Structural positional encoding module.

the ontology aims at grouping together the activities involved in similar diagnostic or treatment goals. This is achieved through a taxonomic representation, where activities involved in similar goals are placed in a close "relationship" (having a shorter path to reach one from the other). Figure 2 shows an excerpt of the ontology integrated in this system. The activities and their relations was defined in collaboration with expert neurologists, on the basis of their experience and according to the recommendations of the SPREAD guideline for stroke patients [33].

To implement and to integrate this ontology, we coded it as a graph and we relied on the Laplacian eigenvector technique [34], which is employed to calculate node embeddings, yielding a vector for each action (i.e., node) that encodes its position in the graph. This ensures that nodes close to each other possess similar embeddings. During the training phase, these



**Figure 2:** Excerpt of the domain ontology

node representations are added to the token embeddings, before the multi-head attention layer (in a positional encoding manner). This augmentation enriches the representations of the transformer with the relational information inherent in the ontology. This enables the model to learn additional types of interactions among activities. We proceed by giving a more formal

description of the operations carried out during this phase.

The graph  $G = (V, E)$  representing the ontology is structured as follows:  $V$  represents the nodes, one for every activity in the set  $A$  (activity nodes), along with some nodes that represent the "type of activity" (activity-type nodes).  $E$  captures the set of edges, connecting activity nodes to activity-type nodes. Activity-type nodes are linked together so that the graph is overall connected. The Laplacian eigenvectors are defined by the factorization of the graph Laplacian matrix:

$$\Delta = I - D^{-\frac{1}{2}}AD^{-\frac{1}{2}} = U^T\Lambda U$$

where  $I$  is the identity matrix,  $A \in \mathbb{R}^{n \times n}$  assuming  $n = |V|$  is the adjacency matrix where  $A_{ij} = 1$  if node  $i$  and node  $j$  are connected by an edge (i.e. the graph connectivity) and  $D \in \mathbb{R}^{n \times n}$  where  $D_{ii} = \sum_{j=1}^n A_{ij}$  is a diagonal matrix representing the degree of every vertex in  $G$ ;  $\Lambda, U$  denote the eigenvalues and eigenvectors respectively. The embedding for a node  $i$  is a vector  $\lambda_i \in \mathbb{R}^k$  defined as the  $k$  smallest nontrivial eigenvector components of that node [35]. Finally, to incorporate the node embedding  $\lambda_i \in \mathbb{R}^k$  into the corresponding activity embedding  $X_i \in \mathbb{R}^d$ , we employ a fully-connected layer  $\Theta \in \mathbb{R}^{k \times d}$  to ensure compatibility between the embedding sizes.

## 4. Experiments

During our experiments, we applied the model to a dataset relative to Stroke management. The dataset contains 5342 process traces with 15 activities on average and 82 possible activities (See Table 3 for some statistics about the traces). Given that cases have varying lengths (Figure 3), we applied padding to ensure that all sequences had the same number of activities (i.e. the length of the longest sequence). Moreover, we added two special tokens indicating "start of sequence" and "end of sequence" at the start and end of each case. The ontology associated to the process was parsed in order to obtain a graph containing 110 nodes and 111 edges.

The transformer model has been trained in an auto-regressive language modeling manner to predict the 'next activity' given a context (i.e., the previous activities in the sequence). The training process involves the model attempting to predict the next activity given a sequence prefix. The loss function applied is the cross-entropy loss between the model's predicted probability distribution over all possible activities and the true tokens in the training data. Notably, during the training phase, examples containing padding tokens or "start of sequence" tokens were excluded from loss calculation. This ensures that the model ignores these examples during the calculation of gradient preventing potential degradation of the model's performance. We instead decided to keep the "end of sentence" tokens so that the model could learn to predict when a sequence of activities should end.

We performed a 80/10/10 split to the data (training, validation and test). We tuned the model hyper-parameters using the Optuna [36] optimization framework. Table 1 reports the search space considered and the best parameters found.

**Table 1**

Parameter search space used to find optimal hyperparameters and optimal configuration found by Optuna.

Parameter	Search space	Best configuration
Embedding size	[16, 32, 64, 128, 256]	64
Hidden size	[16, 32, 64, 128, 256]	128
Number of heads	[1, 2, 4, 8]	4
Number of layers	[1, 2, 3, 4, 5]	4
Dropout rate	0.1 - 0.5	0.216375
Optimizer	AdamW	AdamW
Scheduler	StepLR	StepLR
Gamma	0.85 - 0.99	0.989695
Learning rate	$1 \times 10^{-2}$ - $3 \times 10^{-2}$	0.002836
Ontology embedding size	[8, 16, 32]	32

During our experiments (see Table 2) we aimed to compare how the model performed using the two different positional encoding methods (PE and SPE) and various model sizes. We also included a baseline model in which we skipped the PE phase. For each model configuration, we ran 10 fits with random training-validation-test splits and initialisation. We then reported the mean and standard deviation of accuracy-at- $k$ . The results show that there is a clear benefit in using the SPE method, which substantially improves the model performances across all model sizes. Moreover, the model metrics are quite stable and saturate at with a model size of 64, improving only marginally using 128-sized embedding. This suggests that the model does not tend to overfit.

Another important finding is that the model does not seem to benefit from the classic PE method which only encodes an order to the activities (see Figure 4). This observation may be due to the nature of these processes, where the next activity depends more on which activities have already been performed rather than their sequential order [37]. This also hints to why the model benefits from the SPE technique: activities of a similar type are likely to be executed in close proximity. As a result, the contextual information about an activity’s position within the ontology graph describing relations between activities in the process is much more informative than its position within an individual sequence.

## 5. Conclusions

This paper presents an approach to predictive process monitoring, specifically applied in the challenging domain of stroke management. Leveraging the power of Transformer-based models, we have demonstrated the potential for accurate forecasting of critical information within running process traces. Our key innovation, the integration of domain-specific knowledge through Structural Positional Encoding, has been shown to increase predictive accuracy.

The initial experimental results presented in this study are encouraging, pointing towards the effectiveness of our proposed approach. These findings suggest that predictive process monitoring, using transformer-based models and domain-specific ontological knowledge, holds significant potential in providing valuable decision support in complex medical scenarios.



**Table 2**

Test accuracy-at- $k$  and standard deviation on 10 random initialisation and train-val-test splits.

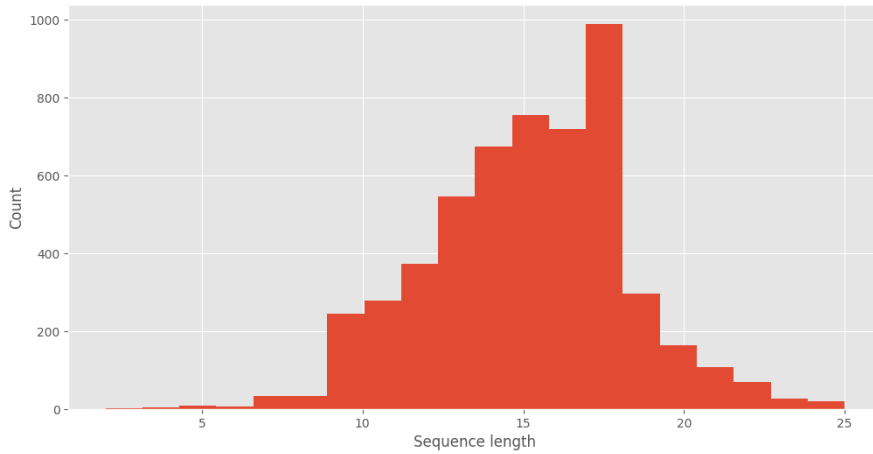
PE method	Model size	Accuracy@1	Accuracy@3	Accuracy@5
None	16	45.6±0.3	67.1±0.3	75.8±0.3
None	32	46.9±0.2	69.4±0.3	78.8±0.3
None	64	47.3±0.2	70.4±0.2	80.0±0.2
None	128	47.2±0.1	70.4±0.3	80.1±0.2
PE	16	45.2±0.3	66.8±0.3	75.7±0.2
PE	32	46.9±0.2	69.4±0.2	78.9±0.2
PE	64	47.1±0.1	70.3±0.3	79.9±0.2
PE	128	47.0±0.2	70.1±0.2	79.8±0.2
SPE	16	48.5±0.4	69.7±0.4	77.9±0.4
SPE	32	52.1±0.3	75.3±0.4	83.5±0.3
SPE	64	54.0±0.3	78.0±0.2	86.5±0.2
SPE	128	54.3±0.3	78.3±0.4	86.8±0.3

In the future, further research and experimentation will be crucial to validate and refine our approach. In particular, we would like to experiment with different kinds of embedding techniques for the graph nodes, and explore this methodology using different datasets, where we can explore in depth the temporal aspects.

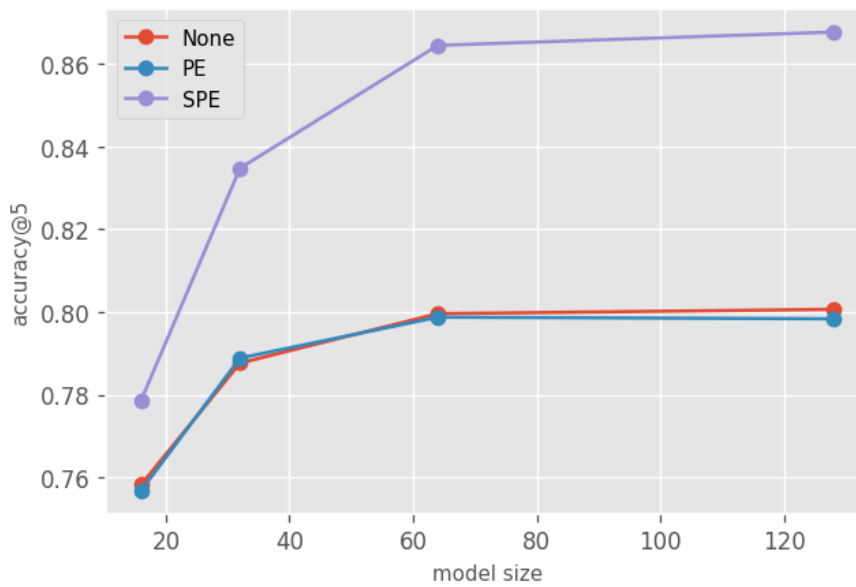
**Table 3**

Datasets traces length statistics measured in number of activities.

Traces length statistics	
Mean	15
Standard Deviation	3
Minimum	2
Maximum	25



**Figure 3:** Sequence length histogram.



**Figure 4:** Model performance with different embedding sizes on testset.

## Acknowledgments

We would like to thank Chameleon Cloud [38] for providing the essential computational resources that facilitated the execution of our experiments and model training.

Christopher Irwin and Marco Dossena are a PhD students enrolled in the National PhD in Artificial Intelligence, XXXVIII cycle, course on Health and life sciences, organized by Università

## References

- [1] M. Reichert, B. Weber, *Enabling Flexibility in Process-Aware Information Systems - Challenges, Methods, Technologies*, Springer, 2012. URL: <https://doi.org/10.1007/978-3-642-30409-5>. doi:10.1007/978-3-642-30409-5.
- [2] W. van der Aalst, *Process Mining. Data Science in Action*, Springer, 2016.
- [3] F. Maggi, C. DiFrancescomarino, M. Dumas, C. Ghidini, Predictive monitoring of business processes, in: M. Jarke, J. Mylopoulos, C. Quix, C. Rolland, Y. Manolopoulos, H. Mouratidis, J. Horkoff (Eds.), *Advanced Information Systems Engineering - 26th International Conference, CAiSE 2014, Thessaloniki, Greece, June 16-20, 2014. Proceedings*, volume 8484 of *Lecture Notes in Computer Science*, Springer, 2014, pp. 457–472. URL: [https://doi.org/10.1007/978-3-319-07881-6\\_31](https://doi.org/10.1007/978-3-319-07881-6_31). doi:10.1007/978-3-319-07881-6\_31.
- [4] I. Teinmaa, M. Dumas, M. LaRosa, F. Maggi, Outcome-oriented predictive process monitoring: Review and benchmark, *ACM Trans. Knowl. Discov. Data* 13 (2019) 17:1–17:57. URL: <https://doi.org/10.1145/3301300>. doi:10.1145/3301300.
- [5] H. Xu, J. Pang, X. Yang, M. Li, D. Zhao, Using predictive process monitoring to assist thrombolytic therapy decision-making for ischemic stroke patients, *BMC Medical Informatics Decis. Mak.* 20-S (2020) 120. URL: <https://doi.org/10.1186/s12911-020-1111-6>. doi:10.1186/s12911-020-1111-6.
- [6] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, I. Polosukhin, Attention is all you need, 2017. [arXiv:1706.03762](https://arxiv.org/abs/1706.03762).
- [7] Z. A. Bukhsh, A. Saeed, R. M. Dijkman, Processtransformer: Predictive business process monitoring with transformer network, 2021. [arXiv:2104.00721](https://arxiv.org/abs/2104.00721).
- [8] A. Bottrighi, L. Canensi, G. Leonardi, S. Montani, P. Terenziani, Trace retrieval for business process operational support, *Expert Syst. Appl.* 55 (2016) 212–221.
- [9] M. Le, B. Gabrys, D. Nauck, A hybrid model for business process event prediction, in: M. Bramer, M. Petridis (Eds.), *Research and Development in Intelligent Systems XXIX, Incorporating Applications and Innovations in Intelligent Systems XX: 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*, Springer, 2012, pp. 179–192.
- [10] G. Lakshmanan, D. Shamsi, Y. N. Doganata, M. Unuvar, R. Khalaf, Markov prediction model for data-driven semi-structured business processes, *Knowl. Inf. Syst.* 42 (1) (2015) 97–126.
- [11] C. Cabanillas, C. DiCiccio, J. Mendling, A. Baumgrass, Predictive task monitoring for business processes, in: S. W. Sadiq, P. Soffer, H. Völzer (Eds.), *Business Process Management - 12th International Conference, BPM 2014, Haifa, Israel, September 7-11, 2014. Proceedings*, volume 8659 of *Lecture Notes in Computer Science*, Springer, 2014, pp. 424–432. URL: [https://doi.org/10.1007/978-3-319-10172-9\\_31](https://doi.org/10.1007/978-3-319-10172-9_31). doi:10.1007/978-3-319-10172-9\_31.
- [12] N. Mehdiyev, J. Evermann, P. Fettke, A multi-stage deep learning approach for business process event prediction, in: P. Loucopoulos, Y. Manolopoulos, O. Pastor, B. Theodoulidis,

- J. Zdravkovic (Eds.), 19th IEEE Conference on Business Informatics, CBI 2017, Thessaloniki, Greece, July 24-27, 2017, Volume 1: Conference Papers, IEEE Computer Society, 2017, pp. 119–128.
- [13] G. E. Hinton, R. R. Salakhutdinov, Reducing the dimensionality of data with neural networks, *Science* 313 (2006) 504–507.
- [14] M. Z. Alom, T. M. Taha, C. Yakopcic, S. Westberg, P. Sidike, M. S. Nasrin, M. Hasan, B. C. Van Essen, A. A. S. Awwal, V. K. Asari, A state-of-the-art survey on deep learning theory and architectures, *Electronics* 8 (2019).
- [15] N. D. Mauro, A. Appice, T. M. A. Basile, Activity prediction of business process instances with inception CNN models, in: M. Alviano, G. Greco, F. Scarcello (Eds.), *AI\*IA 2019 - Advances in Artificial Intelligence - XVIIIth International Conference of the Italian Association for Artificial Intelligence*, Rende, Italy, November 19-22, 2019, Proceedings, volume 11946 of *Lecture Notes in Computer Science*, Springer, 2019, pp. 348–361.
- [16] R. Pascanu, Ç. Gülçehre, K. Cho, Y. Bengio, How to construct deep recurrent neural networks, in: Y. Bengio, Y. LeCun (Eds.), *2nd International Conference on Learning Representations, ICLR 2014, Banff, AB, Canada, April 14-16, 2014, Conference Track Proceedings*, 2014.
- [17] S. Hochreiter, J. Schmidhuber, Long short-term memory, *Neural Computation* 9 (1997) 1735–1780.
- [18] J. Evermann, J. Rehse, P. Fettke, Predicting process behaviour using deep learning, *Decision Support Systems* 100 (2017) 129–140.
- [19] N. Tax, I. Teinmaa, S. J. van Zelst, An interdisciplinary comparison of sequence modeling methods for next-element prediction, *CoRR abs/1811.00062* (2018).
- [20] M. Camargo, M. Dumas, O. G. Rojas, Learning accurate LSTM models of business processes, in: T. T. Hildebrandt, B. F. van Dongen, M. Röglinger, J. Mendling (Eds.), *Business Process Management - 17th International Conference, BPM 2019, Vienna, Austria, September 1-6, 2019, Proceedings*, volume 11675 of *Lecture Notes in Computer Science*, Springer, 2019, pp. 286–302.
- [21] N. Tax, I. Verenich, M. La Rosa, M. Dumas, Predictive business process monitoring with LSTM neural networks, in: *Proceedings of the 29th International Conference on Advanced Information Systems Engineering (CAiSE)*, Springer, 2017, pp. 477–492.
- [22] K. Cho, B. van Merriënboer, C. Gulcehre, D. Bahdanau, F. Bougares, H. Schwenk, Y. Bengio, Learning phrase representations using rnn encoder-decoder for statistical machine translation, 2014. [arXiv:1406.1078](https://arxiv.org/abs/1406.1078).
- [23] M. Hinkka, T. Lehto, K. Heljanko, A. Jung, Classifying process instances using recurrent neural networks, in: F. Daniel, Q. Z. Sheng, H. Motahari (Eds.), *Business Process Management Workshops - BPM 2018 International Workshops, Sydney, NSW, Australia, September 9-14, 2018, Revised Papers*, volume 342 of *Lecture Notes in Business Information Processing*, Springer, 2018, pp. 313–324.
- [24] M. A. Khan, H. Le, K. Do, T. Tran, A. Ghose, K. H. Dam, R. Sindhgatta, Memory-augmented neural networks for predictive process analytics, *CoRR abs/1802.00938* (2018). URL: <http://arxiv.org/abs/1802.00938>. [arXiv:1802.00938](https://arxiv.org/abs/1802.00938).
- [25] P. Philipp, R. Jacob, S. Robert, J. Beyerer, Predictive analysis of business processes using neural networks with attention mechanism, in: *2020 International Conference on Artificial*

- Intelligence in Information and Communication, ICAIIC 2020, Fukuoka, Japan, February 19-21, 2020, IEEE, 2020, pp. 225–230.
- [26] G. Mialon, D. Chen, M. Selosse, J. Mairal, Graphit: Encoding graph structure in transformers, arXiv preprint arXiv:2106.05667 (2021).
- [27] V. P. Dwivedi, C. K. Joshi, A. T. Luu, T. Laurent, Y. Bengio, X. Bresson, Benchmarking graph neural networks, arXiv preprint arXiv:2003.00982 (2020).
- [28] C. Di Francescomarino, C. Ghidini, F. M. Maggi, G. Petrucci, A. Yeshchenko, An eye into the future: leveraging a-priori knowledge in predictive business process monitoring, in: Business Process Management: 15th International Conference, BPM 2017, Barcelona, Spain, September 10–15, 2017, Proceedings 15, Springer, 2017, pp. 252–268.
- [29] W. Liu, P. Zhou, Z. Zhao, Z. Wang, Q. Ju, H. Deng, P. Wang, K-bert: Enabling language representation with knowledge graph, 2020, p. 2901 – 2908. URL: <https://www.scopus.com/inward/record.uri?eid=2-s2.0-85106402604&partnerID=40&md5=f871a87e7bb104c4921c1497e39c8366>, cited by: 308.
- [30] J. Shang, T. Ma, C. Xiao, J. Sun, Pre-training of graph augmented transformers for medication recommendation, CoRR abs/1906.00346 (2019). URL: <http://arxiv.org/abs/1906.00346>. arXiv:1906.00346.
- [31] J. L. Ba, J. R. Kiros, G. E. Hinton, Layer normalization, arXiv preprint arXiv:1607.06450 (2016).
- [32] R. Xiong, Y. Yang, D. He, K. Zheng, S. Zheng, C. Xing, H. Zhang, Y. Lan, L. Wang, T. Liu, On layer normalization in the transformer architecture, in: International Conference on Machine Learning, PMLR, 2020, pp. 10524–10533.
- [33] G. Lanza, C. Setacci, S. Ricci, P. Castelli, A. Cremonesi, J. Lanza, C. Novali, C. Pratesi, P. Santalucia, F. Speziale, A. Zaninelli, G. Gensini, An update of the italian stroke organization-stroke prevention awareness diffusion group guidelines on carotid endarterectomy and stenting: A personalized medicine approach., International journal of stroke : official journal of the International Stroke Society 12(5) (2017) 560–567. doi:<https://doi.org/10.1177/1747493017694395>.
- [34] M. Belkin, P. Niyogi, Laplacian eigenmaps for dimensionality reduction and data representation, Neural computation 15 (2003) 1373–1396.
- [35] V. P. Dwivedi, X. Bresson, A generalization of transformer networks to graphs, AAAI Workshop on Deep Learning on Graphs: Methods and Applications (2021).
- [36] T. Akiba, S. Sano, T. Yanase, T. Ohta, M. Koyama, Optuna: A next-generation hyperparameter optimization framework, in: Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, 2019.
- [37] Italian-Stroke-Association, Current guidelines, 2023. URL: <https://isa-aii.com/linee-guida/linee-guida-attuali/>.
- [38] K. Keahey, J. Anderson, Z. Zhen, P. Riteau, P. Ruth, D. Stanzione, M. Cevik, J. Colleran, H. S. Gunawi, C. Hammock, J. Mambretti, A. Barnes, F. Halbach, A. Rocha, J. Stubbs, Lessons learned from the chameleon testbed, in: Proceedings of the 2020 USENIX Annual Technical Conference (USENIX ATC '20), USENIX Association, 2020.