



HAL
open science

Predictive process monitoring using object-centric graph embeddings

Wissam Gherissi, Mehdi Acheli, Joyce El Haddad, Daniela Grigori

► **To cite this version:**

Wissam Gherissi, Mehdi Acheli, Joyce El Haddad, Daniela Grigori. Predictive process monitoring using object-centric graph embeddings. International Conference on Service-Oriented Computing (ICSOC Workshops), Dec 2024, Tunis, Tunisia. pp.55-67, <10.1007/978-981-96-7238-7_5>. <hal-05171537>

HAL Id: hal-05171537

<https://hal.science/hal-05171537v1>

Submitted on 20 Jul 2025

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire HAL, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.



HAL Authorization

Predictive Process Monitoring Using Object-centric Graph Embeddings

Wissam Gherissi¹[0000-0002-2234-2963], Mehdi Acheli²[0000-0001-9649-7127],
Joyce El Haddad¹[0000-0002-2709-2430], and Daniela
Grigori¹[0000-0003-1741-8676]

¹ Université Paris-Dauphine, Université PSL, CNRS, LAMSADE, 75016 PARIS,
FRANCE

{wissam.gherissi, joyce.elhaddad, daniela.grigori}@dauphine.psl.eu

² Telecom SudParis, Institut Polytechnique de Paris, PARIS, FRANCE
mehdi.acheli@telecom-sudparis.eu

Abstract. Object-centric predictive process monitoring explores and utilizes object-centric event logs to enhance process predictions. The main challenge lies in extracting relevant information and building effective models. In this paper, we propose an end-to-end model that predicts future process behavior, focusing on two tasks: next activity prediction and next event time. The proposed model employs a graph attention network to encode activities and their relationships, combined with an LSTM network to handle temporal dependencies. Evaluated on one real-life and three synthetic event logs, the model demonstrates competitive performance compared to state-of-the-art methods.

Keywords: Predictive Process Monitoring · Object-centric · Graph Attention Network · Deep Learning.

1 Introduction

Predictive Process Monitoring (PPM) is a key area of process mining focused on predicting the future evolution, potential deviations, and variations in processes. The objective is to leverage advancements in machine learning and deep learning to develop models that can make accurate predictions based on event data, with a recent focus on object-centric event logs. Traditional process mining models simplify event logs by associating each event with a single case, but this approach has limitations in real-world scenarios. Van der Aalst highlighted in [1] shortcomings of this single case notion like convergence and divergence. Convergence occurs when the same event is linked to multiple cases, such as a shipping event tied to different items in the same package. Divergence arises when the same activity happens multiple times for a single case, such as picking and sorting multiple items for the same order.

To overcome the aforementioned shortcomings, Object-Centric Event Logs (OCEL) were recently introduced in [11]. OCEL captures multiple perspectives of a process by relating events to various object types, offering richer insights

compared to single-case event logs. This new object-centric data format has inspired advancements in traditional process mining techniques, including process discovery [2], its related metrics (precision and fitness) [5], conformance checking [15], and predictive process monitoring. In PPM, recent works [3], [4], [10] and [12] have explored different ways to leverage OCEL’s object-centric information to improve predictions. One research direction, as in [3], focuses on creating process graphs that maintain the structural integrity of processes, avoiding the convergence and divergence problems where events relate to multiple or repeated cases. This method predicts the next event across the entire process without focusing on specific object types, which can lead to cumulative errors when predicting across different objects. We argue for a different approach focusing predictions on a single object type at a time. This perspective-driven method, known as *flattening* operation and defined in [6], is useful when the analyst is primarily interested in one aspect of the process, such as the order in a retail system, rather than all related objects (e.g., individual items). For instance, if an analyst wants to predict the next event timestamp in an order process, this method directly targets order-related events without iterating through irrelevant item deliveries, thus reducing accumulation of deviations and prediction errors. However, while focusing on a single object type, our perspective-driven approach still benefits from considering interconnected object types. The dependencies between orders and items, for example, mean that item deliveries are influenced by order payments, and an order is only completed when all items are delivered. This relationship can be effectively modeled using an Object-Centric Directly-Follows Graph (OCDFG) [6], which captures both the activities and their connections to object types, allowing for more accurate predictions within the specific perspective.

In this paper, we propose a Graph Attention Network (GAT) to embed the nodes representing activities in the OCDFG, combined with a Long Short-Term Memory (LSTM) architecture for prediction. GATs excel at capturing the spatial relationships between nodes in a graph, enabling them to model interactions between activities across multiple object types. These graph embeddings serve as contextual information that enhances the LSTM model’s performance in two key prediction tasks: predicting the Next Activity and the Next Event Time. Both tasks are performed on the flattened event log of the selected object type.

The remainder of the paper is structured as follows. Section 2 reviews related work on Object-Centric Process Mining (OCPM) and PPM. Section 3 covers key concepts, including event logs, graph neural networks, and LSTM architectures. Section 4 details the proposed graph and LSTM model, from data preprocessing to predictions. Section 5 presents experimental results and discussions of findings, followed by conclusions in Section 6.

2 Related Work

Process mining, particularly predictive process monitoring, has evolved significantly with advancements in machine learning and deep learning. Various mod-

els have been adapted to predict future processes using techniques like sequence encoding with recurrent neural networks [14, 18], transformers [7] and graph networks [13]. In recent years, graph structures and embeddings have gained prominence. For example, previous work [16] combined graph encodings with Long Short-Term Memory (LSTM) architectures for process model graphs, while other research [8] and [9] focused on graph representations for event logs to predict remaining sequence time using gated graph neural networks or graph transformers. [8] used a gated graph neural network by transforming the prefix into a graph connecting events based on precedence (Forward, Backward, Repeat) whereas [9] deploy a graph transformer architecture by using a GPS recipe [17]. However, these studies focused on traditional event logs, unlike the OCELS used in this paper.

The introduction of OCELS has opened up new possibilities for PPM, allowing for richer representations of processes by incorporating multiple object types within the same event log. One study [3] analyzed OCELS using graph structures by defining process executions that grouped multiple objects and case notions. This method chose a leading object type and applied a distance threshold to limit object connections, which helped avoid an overwhelming number of events and related objects in the same execution. However, the prediction targets in such approaches are not specific to a particular object type, but rather predict the next activity independently of process perspectives.

Other works, such as those in [12] and [10], have aimed to improve prediction performance by adding object-centric features and context. They incorporated additional information, like the number of related objects for each event, to enrich the data and enhance prediction accuracy. In [10], authors also constructed traces that included events related to specific objects and their interactions with other objects, to predict key performance indicators. In contrast to these approaches [12] and [10], our paper introduces a method that incorporates object-centric relationships between activities in the form of graph embeddings, which are then passed through an LSTM architecture. Our approach allows for more accurate predictions by taking into account the relationships between object types, offering a distinct perspective compared to previous methods. Our evaluation study compares its performance against publicly available logs and implementations.

3 Preliminaries

3.1 Process Mining

To introduce the definition of object-centric event logs, we use some universes presented in [6]. Given \mathbb{U}_E a universe of events, \mathbb{U}_{act} a universe of activities, \mathbb{U}_o a universe of object identifiers, \mathbb{U}_{ot} a universe of object types, and \mathbb{U}_{time} a universe of timestamps, an object-centric event log can be defined as follow:

Definition 1 (Object-Centric Event Log). *An object-centric event log \mathcal{L} is defined as a tuple $(E, O, OT, \pi_{act}, \pi_{time}, \pi_{omap}, \pi_{otyp}, <)$ where,*

- $E \subseteq \mathbb{U}_E$ is the set of events, $O \subseteq \mathbb{U}_o$ is the set of object identifiers, and $OT \subseteq \mathbb{U}_{ot}$ is the set of object types.
- $\pi_{act} : E \rightarrow \mathbb{U}_{act}$ is the function associating each event with its activity.
- $\pi_{time} : E \rightarrow \mathbb{U}_{time}$ is the function associating each event with a timestamp.
- $\pi_{omap} : E \rightarrow \mathcal{P}(O)$ is the function associating an event identifier to a set of related object identifiers.
- $\pi_{otyp} \in O \rightarrow OT$ is the function assigning precisely one object type for each object identifier.
- $<$ is a partial order based on the timestamps of events such that

$$e_1 < e_2 \iff \pi_{time}(e_1) < \pi_{time}(e_2)$$

Definition 2 (Flattened Log). Let \mathcal{L} be an object-centric event log, and $ot \in OT$ be an object type. We define the flattened log of \mathcal{L} over ot as $\mathcal{FL}(\mathcal{L}, ot) = (E^{ot}, \pi_{act}^{ot}, \pi_{time}^{ot}, \pi_{case}^{ot}, \leq^{ot})$ where:

- $E^{ot} = \{e \in E \mid \exists o \in O \pi_{otyp}(o) = ot \wedge o \in \pi_{omap}(e)\}$
- $\pi_{act}^{ot} = \pi_{act}|_{E^{ot}}$
- $\pi_{time}^{ot} = \pi_{time}|_{E^{ot}}$
- For $e \in E^{ot}$, $\pi_{case}^{ot}(e) = \{o \in \pi_{omap}(e) \mid \pi_{otyp}(o) = ot\}$
- $\leq^{ot} = \{e_1 < e_2 \mid \exists o \in O \pi_{otyp}(o) = ot \wedge o \in \pi_{omap}(e_1) \cap \pi_{omap}(e_2)\}$

The flattening of OCEL with an object-type will allow us to define prediction tasks specific to an object-based perspective.

Definition 3 (Directly-Follows Graph). A directly-follows graph \mathcal{DFG} is a tuple (A, F, π_{frege}) where:

- $A \subseteq U_{act}$ is a set of activities,
- $F \subseteq A \times A$ is the set of edges,
- $\pi_{frege} : F \rightarrow \mathbb{N}$ is a frequency measure on the edges.

Note that in this definition, we use the version of DFG considering only the frequency measures on edges.

Definition 4 (Discovery of DFG). Let $\mathcal{FL} = (E^{ot}, \pi_{act}^{ot}, \pi_{time}^{ot}, \pi_{case}^{ot}, \leq^{ot})$ be a flattened event log, its associated directly-follows graph $\mathcal{DFG}(\mathcal{FL}) = (A, F, \pi_{frege})$ is constructed following the discovery operation where :

- $A = \pi_{act}^{ot}(\mathcal{FL})$,
- $F = \{\cup_{c \in \pi_{case}^{ot}(\mathcal{FL}), \mathcal{FL}(c) = (a_1, \dots, a_n)} \{(a_i, a_{i+1}) \mid 1 \leq i < n\}\}$
- For $(a, b) \in F \cap (A \times A)$,

$$\pi_{frege}(a, b) = \sum_{\substack{c \in \pi_{case}^{ot}(\mathcal{FL}), \\ \mathcal{FL}(c) = (a_1, \dots, a_n)}} |\{i \in \mathbb{N} \mid 1 \leq i < n \wedge a_i = a \wedge a_{i+1} = b\}|$$

Definition 5 (Object-Centric Directly-Follows Graph). An object-centric directly-follows graph \mathcal{OCDFG} is a tuple (A, OT, F, π_{ofrege}) where:

- A is a set of activities,
- OT is a set of object types,
- $F \subseteq A \times A$ is a set of edges,
- $\pi_{ofreqe} : F \times OT \rightarrow \mathbb{R}^+$ assigns for each edge and object type a frequency measure.

Definition 6 (Discovery of OCDFG). Given an OCEL \mathcal{L} , and a set of DFGs of flattened logs $\{G_i \mid G_i = \text{DFG}(\mathcal{FL}(\mathcal{L}, ot_i)) = (A_i, F_i, \pi_i), \text{ for } i = 1 \dots |OT|\}$, the discovery operation of $\text{OCDFG}(\mathcal{L}) = (A, OT, F, \pi_{ofreqe})$ where for given $n1, n2 \in A$ we have $(n1, n2) \in F \Leftrightarrow \exists o_i \in OT$ such that $(n1, n2) \in F_i$, and where edge frequencies for each object type are computed as follows : $\forall o_i \in OT$, if $(n1, n2) \in F_i$ then $\pi_{ofreqe}(n1, n2, o_i) = \pi_i(n1, n2)$; else $\pi_{ofreqe}(n1, n2, o_i) = 0$. The discovery is summed up in the following steps:

- Flatten OCEL for each object type to allow an object type-based prediction,
- Perform discovery of DFG for each flattened event log (Def.4),
- Merge edges of the resulting DFGs between each pair of nodes. Each edge will have a frequency associated to each object type, equal to the frequency of the edge of the corresponding DFG of the flattened log or 0 if no edge exist. The resulting OCDFG aggregates dependencies between activities for different object types.

3.2 Neural networks

Graph Attention Networks (GAT) [19] are a type of Graph Neural Network (GNN) that incorporate attention mechanisms to process graph-structured data. Unlike traditional GNNs, which apply fixed or uniform weights to aggregate information from neighboring nodes, GAT dynamically learns to weigh the importance of each neighboring node during message-passing. This is useful when nodes vary in their relevance to the prediction task. In PPM, processes can be represented as graphs, where nodes correspond to activities and edges reflect inter-relationships, such as causal dependencies, or directly-follows relationships in our case. GAT focuses on the most informative event relationships, filtering out less relevant connections, and assigns different weights to neighboring nodes, based on their influence on the outcomes. By capturing these complex dependencies, GAT improves the accuracy and interpretability of prediction models. Its ability to highlight important interactions within event logs makes GAT an effective tool for understanding and predicting process behaviors.

Long Short Term Memory (LSTM) networks are a type of Recurrent Neural Network (RNN) designed to overcome the limitations of traditional RNNs in modeling long-term dependencies. LSTMs use a gating mechanism to control the flow of information, allowing them to retain important information over long time horizons while mitigating the vanishing gradient problem. LSTMs use a gating mechanism to manage information flow, retaining essential details over extended periods and solving the vanishing gradient problem. This makes

LSTMs optimal for handling sequential data, such as event sequences in PPM, where past events influence future outcomes. By maintaining a memory of past events, LSTMs are able to predict future events or outcomes more accurately, making them an ideal candidate for our use-case.

To conclude, the combination of GAT and LSTM leverages the strengths of both architectures. GAT focuses on the relational aspect of activities in an OCDFG by learning the dependencies between them, while LSTM models the temporal dynamics of the process sequence. By combining these two approaches, the model can simultaneously learn both the structural (graph-based) and temporal (sequence-based) characteristics of event logs.

4 Our approach

Our goal is to develop an end-to-end model combining a Graph Attention Network (GAT) with LSTM layers to predict future behavior in a flattened event log for a specific object type. In real-life business scenarios, tracking and predicting each object’s lifecycle is crucial. To achieve this, we apply a flattening operation, focusing on individual object types while still utilizing inter-object contextual information from the OCEL via the OCDFG. Our approach consists of two main steps: preprocessing and prediction, each detailed further in the following sections.

4.1 Step 1 : Preprocessing

Below, we outline the key operations involved in the preprocessing step, which prepares the model’s inputs. Starting with the OCEL as the initial input, we perform a flattening operation (see Def.2) on each object type in the OCEL to generate single object-type event logs. For the selected object type and its corresponding flattened log, we extract the temporal features and prediction targets, as further detailed and shown in Fig.1(a). Simultaneously, we use the collection of flattened event logs to discover the OCDFG (see Def.4), as illustrated and explained in Fig.1.

Temporal features and prediction targets. We begin by flattening the OCEL (\mathcal{L}) on different object types. For a given flattened event log ($\mathcal{FL}(OT_i)$), we extract each object sequence and generate multiple prefixes of varying lengths. For example, for object type *item*, if the object is *i2*, we extract a prefix of length 3: **Place Order(PO)**, **Pick Item (PI)**, **Send Delivery(SD)**.

For each event in the prefix, we compute the following temporal features: *time since the last event*, *time since the start of the prefix*, *time since midnight* of the same day of the event and *weekday of the event*.

As prediction targets, we retain information on the next activity following the prefix (NA) and the time until the execution of that next activity (NE).

Finally, we apply average scaling to all temporal features and time targets to maintain computational stability while preserving the temporal distribution..

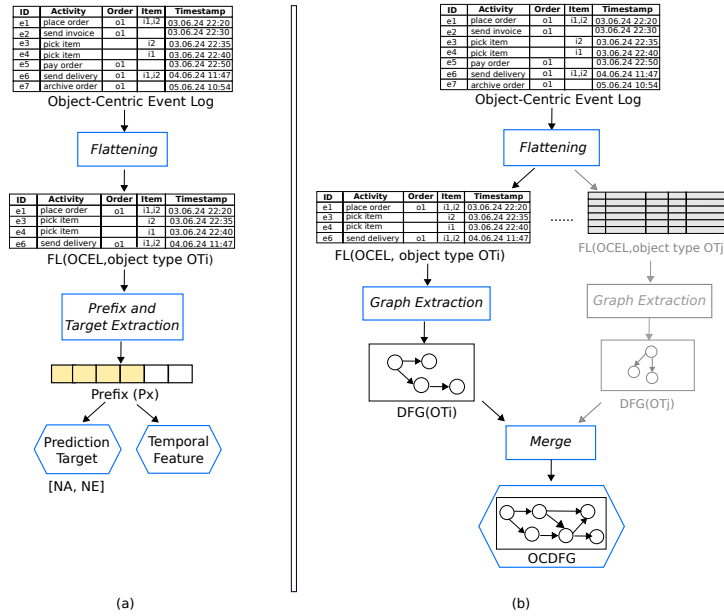


Fig. 1. (a) Temporal features and prediction targets preprocessing, and (b) OCDFG construction

Object-centric directly-follows graph. From each flattened log, we discover a DFG (see Def. 4). We then merge these individual DFGs to construct an Object-Centric Directly-Follows Graph (OCDFG) (see Def.6). The resulting OCDFG captures the connections between activities across all object types and flattened event logs, as illustrated in Fig.1(b). Specifically, each edge between consecutive activities represents all object types and their corresponding frequencies. This resulting graph serves as the input for the GAT in the model.

4.2 Step 2 : Prediction model

In this step, first, we feed OCDFG as input for the GAT to generate embeddings for the nodes representing the activities. Next, we perform a *Match and Concatenate* operation combining the temporal features of the events with their corresponding graph embeddings. These combined features are then fed into the LSTM network to make predictions for the next activity (\widehat{NA}) and the time until the next event (\widehat{NE}). This approach ensures smooth gradient flow through the pipeline, allowing both the GAT and LSTM networks to be updated. The overall process is illustrated in Fig.2.

The neural network model consists of two components: a Graph Attention Network (GAT) and an LSTM architecture. The GAT generates node embeddings from the OCDFG, while the LSTM captures the sequential information from

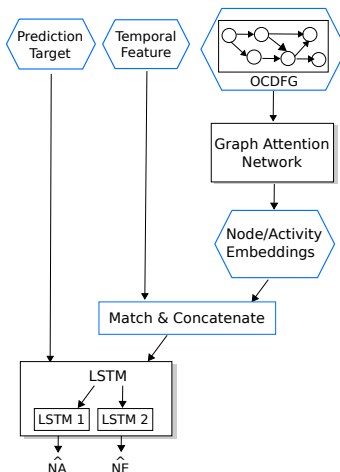


Fig. 2. Application of the approach on object type OT_i

prefixes to predict future behavior. This structure is depicted in Fig. 2. Further details of the model are provided below.

- **Graph Attention Network** consist of three Graph Attention Convolution layers, with variable hidden and embedding output size depending on the event log used. The OCDFG is passed through this network to generate node embeddings, which encode each activity. For a given sequence of events, each event has an associated activity. We perform a differential matching operation, aligning each activity in the sequence with its corresponding embedding, and then concatenating the selected embedding vector with the other event features. For instance, consider an event e_1 with the activity `Place_order`. In the OCDFG embeddings, the node `P0`, for activity `Place_order`, is represented by the vector V_1 . We match this node with the activity `Place_order` in the event and concatenate the embedding vector V_1 with the temporal features of e_1 , resulting in the final feature vector.
- **LSTM network** consists of a shared LSTM layer with hidden and output size 100, followed by two other specialized LSTM layers, each dedicated to a specific prediction task. Both layers have a hidden size of 100. The specialized layer for Next Activity prediction outputs the probabilities for each possible activity, while the Next Event Time prediction layer has an output size of 1 for the regression task. The input to the LSTM network is a sequence prefix, where each event is represented by its activity embedding and temporal features.

5 Experimental results

In this section, we evaluate our model (**LSTM+GAT**) and compare it against two publicly available prediction models: one-hot encoding+LSTM of [12] (**LSTM**)

and ProcessTransformer of [7] (**PT**). The evaluation is performed using four OCELS, including a real-life event log for loan applications from the BPIC17 dataset³ and three synthetic logs simulating an order management process, a logistics process and a Purchase-to-Pay process⁴.

The experiments were conducted on a server equipped with AMD EPYC 7702 64-Core Server Processor and GeForce RTX 2080 Ti. The code and results are reproducible and can be accessed here⁵.

5.1 Event logs

Order Management. describes an order management process. It contains 22367 events, 11522 objects and 5 object types. We focus only on three specific object types that are essential to the process which are: **item**, **order** and **package**. Hence, we flatten the OCEL input based only on these object types.

Logistics. describes the logistics process of a company that sells goods overseas. It contains 35761 events, 14013 object and 7 object types. In this log, we focus on object types with processes containing at least three different types of activities, otherwise, the next activity prediction can be considered meaningless if there is one possible future activity. We consider only the predictions for the following object types : **container**, **transport document** and **vehicle**.

Purchase-to-Pay. describes the Procure-to-Pay process within an organization following the creation of a purchase until the execution of the payment. It contains 14671 events, 9543 objects and 7 object types. Similarly, we consider only processes with more than two different types of activities. We consider only the following object types: **goods receipt**, **invoice receipt**, **purchase order**, **purchase requisition**, and **quotation**.

BPIC17. is a real-life event log describing a loan application process in a financial institution following the process of applications and the offers provided by the institution. It contains 507553 events, 67498 objects and 2 object types: **application** and **offer**.

5.2 Results

Next Activity Prediction. In Tab.1, we see that our model consistently outperforms the basic LSTM model across all event logs and object types. This performance improvement is closely tied to the difference in activity encoding methods. The integration of the GAT model, which incorporates object-centric information, provides deeper insights that enhance the accuracy of activity predictions. Compared to ProcessTransformer, our model and the basic LSTM perform better on three out of the four event logs (excluding Order and Transport Doc). This highlights the robustness of the LSTM architecture for Next Activity prediction.

³ https://data.4tu.nl/articles/dataset/BPI_Challenge_2017/12696884

⁴ Made available by <https://www.ocel-standard.org/event-logs>

⁵ <https://github.com/wissam-gherissi/GATLSTM>

Event logs	Object Type	Model		
		LSTM	PT	LSTM+GAT
Order Management	Order	58.7(54.4)	72.2(65.4)	59.6(52.2)
	Item	73.1(66.5)	76.1(64.4)	77.4(73.6)
	Package	90.4(87.6)	82.3(71.1)	90.5(86.3)
BPIC17	Application	87.2(85.4)	82.3(71.1)	88.0(86.1)
	Offer	81.7(74.5)	79.4(73.0)	81.7(74.9)
Container Logistics	Container	95.6(95.4)	97.3(96.0)	97.3(97.2)
	Transport Doc	97.4(96.2)	98.3(97.1)	97.4(96.2)
	Vehicle	87.9(84.6)	86.4(86.5)	88.0(84.8)
P2P	Goods	76.4(71.5)	74.2(65.0)	76.4(71.6)
	Invoice	81.0(80.1)	85.8(78.2)	80.7(79.5)
	Purch Order	83.0(82.6)	70.4(67.8)	83.2(82.3)
	Purch Requisition	92.0(88.3)	93.4(85.7)	92.0(88.3)
	Quotation	81.1(81.1)	87.9(82.3)	82.4(82.2)

Table 1. Accuracy (F1 score) expressed in percentages for Next Activity prediction task

Event logs	Object Type	Model		
		LSTM	PT	LSTM+GAT
Order Management	Order	0.97	1.60	0.95
	Item	0.94	1.63	1.29
	Package	0.32	0.26	0.37
BPIC17	Application	0.42	0.41	0.41
	Offer	1.28	2.55	1.28
Container Logistics	Container	0.69	0.74	0.69
	Transport Doc	0.59	2.7	0.60
	Vehicle	0.64	0.61	0.67
P2P	Goods	0.81	0.92	0.80
	Invoice	0.44	0.91	0.43
	Purch Order	1.66	1.50	1.67
	Purch Requisition	1.03	1.40	1.07
	Quotation	0.55	1.00	0.58

Table 2. Mean Absolute Error (MAE) expressed in days for Next Event Time prediction task

However, for the specific object types where ProcessTransformer outperforms our model or the LSTM model, we observe that LSTM-based models tend to struggle. This underperformance suggests that LSTMs can suffer from issues like loops, overfitting, or converging to local optima, which limit their ability to predict the next activity effectively in certain cases.

Next Event Time Prediction. In Tab.2, we observe a slight improvement in our model’s performance for predicting the next event time compared to the LSTM model. The use of graph embeddings adds valuable information about activities and their interrelations for each object type. However, this improvement is not

substantial or consistent across all tested event logs. This suggests that while the directly-follows relationship between activities helps, the primary influence still comes from the temporal features extracted from the prefix. Our model outperforms the others on at least half of the event logs and object types. The LSTM model follows as the second best, with ProcessTransformer performing best only on three object types.

5.3 Discussion

This section discusses the proposed model and draws conclusions from the experimental results. The end-to-end nature of the model allows the GAT to update activity embeddings based on the prediction loss, leading to better representation of activities and their relationships using the OCDFG for capturing contextual information. The analysis of results in Tab.1 shows significant improvement and superior performance of the LSTM+GAT model compared to both LSTM and PT models. However, there is room for improvement in addressing issues like loops and overfitting, particularly with the **Order** object type. Additionally, Tab.2 highlights the robust performance of the LSTM architecture for sequential prediction, with LSTM-based models outperforming PT in the Next Event Time task. However, the slight improvement from the baseline LSTM to LSTM+GAT suggests that incorporating additional temporal features may enhance model performance further.

Overall, the experiments demonstrate promising results in two key aspects: the effectiveness of LSTM for predictions and the use of graph embeddings to represent activities via the OCDFG. ProcessTransformer was outperformed by LSTM models in both Next Activity and Next Event Time prediction tasks, except for a few object types in the latter task. The superior performance of LSTM+GAT over LSTM, particularly in Next Activity prediction, confirms the advantage of using object-centric graph embeddings instead of one-hot encoding. To improve Next Event Time prediction, incorporating temporal information, such as average running times and waiting times in the OCDFG, is suggested.

6 Conclusion and future work

In this paper, we proposed a solution for predictive monitoring using object-centric logs. By leveraging a graph model to embed activities based on an OCDFG, we introduced an end-to-end architecture that outperformed baseline models, including a simple LSTM and ProcessTransformer. Our approach focused on each object type by flattening the OCEL while utilizing the object-centric information in the OCDFG. We discussed the key findings and model performance compared to state-of-the-art models. Future work will explore more real-life OCEs, unsupervised embeddings like Node2Vec, and time-related features in the OCDFG. Additionally, we will investigate advanced graph neural networks, such as Dynamic Graph Neural Networks, to capture evolving patterns and enhance Next Event Time prediction.

References

1. van der Aalst, W.M.P.: Object-centric process mining: Dealing with divergence and convergence in event data. In: *Software Engineering and Formal Methods*. pp. 3–25 (2019)
2. van der Aalst, W.M., Berti, A.: Discovering Object-centric Petri Nets. *Fundamenta Informaticae* **175**(1-4), 1 – 40 (2020)
3. Adams, J.N., Park, G., van der Aalst, W.M.: Preserving complex object-centric graph structures to improve machine learning tasks in process mining. *Engineering Applications of Artificial Intelligence* **125**, 106764 (2023)
4. Adams, J.N., Schuster, D., Schmitz, S., Schuh, G., van der Aalst, W.M.: Defining cases and variants for object-centric event data. In: *2022 4th International Conference on Process Mining (ICPM)*. pp. 128–135 (2022)
5. Adams, J.N., Van Der Aalst, W.M.: Precision and fitness in object-centric process mining. In: *2021 3rd International Conference on Process Mining (ICPM)*. pp. 128–135 (2021)
6. Berti, A., Aalst, W.: Oc-pm: analyzing object-centric event logs and process models. *International Journal on Software Tools for Technology Transfer* **25**, 1–17 (09 2022)
7. Bukhsh, Z.A., Saeed, A., Dijkman, R.M.: Processtransformer: Predictive business process monitoring with transformer network (2021)
8. Duong, L.T., Travé-Massuyès, L., Subias, A., Merle, C.: Remaining cycle time prediction with graph neural networks for predictive process monitoring. In: *Proceedings of the 2023 8th International Conference on Machine Learning Technologies*. p. 95–101. ICMLT '23 (2023)
9. Elyasi, K.A., van der Aa, H., Stuckenschmidt, H.: Pgtnet: A process graph transformer network for remaining time prediction of business process instances (2024)
10. Galanti, R., de Leoni, M., Navarin, N., Marazzi, A.: Object-centric process predictive analytics. *Expert Syst. Appl.* **213**(Part), 119173 (2023)
11. Ghahfarokhi, A.F., Park, G., Berti, A., van der Aalst, W.: Ocel standard (2020), <http://ocel-standard.org/1.0/specification.pdf>
12. Gherissi, W., El Haddad, J., Grigori, D.: Object-centric predictive process monitoring. In: *Service-Oriented Computing – ICSOC 2022 Workshops*. pp. 27–39 (2023)
13. Harl, M., Weinzierl, S., Stierle, M., Matzner, M.: Explainable predictive business process monitoring using gated graph neural networks. *Journal of Decision Systems* pp. 1–16 (06 2020)
14. Hinkka, M., Lehto, T., Heljanko, K.: Exploiting event log event attributes in rnn based prediction. In: *Data-Driven Process Discovery and Analysis*. pp. 67–85 (2020)
15. Park, G., van der Aalst, W.M.P.: Monitoring constraints in business processes using object-centric constraint graphs. In: *Process Mining Workshops*. pp. 479–492 (2023)
16. Rama-Maneiro, E., Vidal, J.C., Lama, M.: Embedding graph convolutional networks in recurrent neural networks for predictive monitoring. *IEEE Transactions on Knowledge and Data Engineering* **36**(1), 137–151 (2024)
17. Rampášek, L., Galkin, M., Dwivedi, V.P., Luu, A.T., Wolf, G., Beaini, D.: Recipe for a general, powerful, scalable graph transformer. In: *Advances in Neural Information Processing Systems*. vol. 35, pp. 14501–14515 (2022)
18. Tax, N., Verenich, I., La Rosa, M., Dumas, M.: Predictive business process monitoring with lstm neural networks. In: *CAiSE 2017*. pp. 477–492 (2017)
19. Veličković, P., Cucurull, G., Casanova, A., Romero, A., Liò, P., Bengio, Y.: Graph attention networks (2018)