PROPHET: Explainable Predictive Process Monitoring with Heterogeneous Graph Neural Networks

Vincenzo Pasquadibisceglie, and Raffaele Scaringi, and Annalisa Appice, and Giovanna Castellano, *Senior Member, IEEE* and Donato Malerba, *Senior Member, IEEE*

Abstract—In this paper, we introduce PROPHET, an innovative approach to predictive process monitoring based on Heterogeneous Graph Neural Networks. PROPHET is designed to strike a balance between accurate predictions and interpretability, particularly focusing on the next-activity prediction task. For this purpose, we represent the event traces recorded for different business process executions as heterogeneous graphs within a multi-view learning scheme combined with a heterogeneous graph learning approach. Using heterogeneous Graph Attention Networks (GATs), we achieve good accuracy by incorporating different characteristics of several events into graphs with different node types and leveraging different types of graph links to express relationships between event characteristics, as well as relationships between events. In addition, the use of a GAT model enables the integration of a modified version of the GNN Explainer algorithm to add the explainable component to the predictive model. In particular, the GNN Explainer algorithm is modified to disclose explainable information related to characteristics, events and relationships between events that mainly influenced the prediction. Experiments with various benchmark event logs prove the accuracy of PROPHET compared to several current state-of-the-art methods and draw insights from explanations recovered through the GNN Explainer algorithm.

Index Terms—Predictive process monitoring, Heterogeneous Graphs, Explainable AI, Deep learning, Multi-view learning, Graph Neural Networks.

1 Introduction

Predictive Process Monitoring (PPM) is a branch of process mining, which aims to forecast the unfolding (e.g., next-activity, completion time cycle or outcome) of a business process execution based on the raw event data recorded with its ongoing event trace (i.e., running trace). In fact, an event trace is recorded as a sequence of events, i.e., activities invoked at a specific timestamp by a process execution from the beginning of its execution to the current time. Activities and timestamps are mandatory characteristics recorded with events, but there are also optional process-defined characteristics (e.g., resources that perform activities or cost of activities) that can be recorded in event logs. Performing the proactive prediction of the next-activity of a running trace may help to ensure that the business activities will run in a desired manner by mitigating possible failures and deviations from designed process structures.

With the recent boom of deep learning, the PPM literature has achieved amazing results using different deep learning approaches, such as Long Short-Term Neural Network (LSTMs) [1], [2], [3], [4], Convolutional Neural Networks (CNNs) [5], [6] and Vision Transformers (ViTs) [7], to equip process-aware information systems with deep neural models able to produce accurate predictions of the next-activity of a running trace. On the other hand, over the last five years, multi-view learning has emerged as a prominent deep learning direction for predictive business process monitoring. In particular, the recent studies of [1], [7] have shown that the multi-view learning strategy allows the development of flexible deep neural models that handle multiple characteristics of an event as multiple views of the same trace and gain accuracy into PPM, thanks to their ability to leverage the diversity of the information in each view mitigating the curse of dimensionality. Nevertheless, deep neural models generally learn opaque models, while easier-to-explain predictive models are becoming increasingly desirable in PPM applications [3], [7], [8].

On the other hand, in the last decade, graph representation learning has achieved groundbreaking results in different tasks, including node classification, graph classification, graph generation, and link prediction, becoming a topic of intense research in deep learning [9]. Deep neural models for graphs have recently gained prominence across various domains, encompassing areas such as chemistry, recommender systems, and cultural heritage. Several recent studies have significantly advanced the research in graph-based deep learning methods contributing to the assessment of the potential of such methods as effective models to take advantage of pairwise relationships between components [10], [11] in several domains.

Accordingly the recent PPM literature has seen the emergence of different graph-based learning approaches as well. Event traces can be represented as graphs for several reasons. A graph serves as a generalized data type, relaxing some constraints inherent to images or sequence data. Unlike images, which can be viewed as grids of pixels, graphs

• *V. Pasquadibisceglie, R. Scaringi, A. Appice, G. Castellano and D. Malerba are with the Dipartimento di Informatica, Università degli Studi Aldo Moro di Bari and Consorzio Interuniversitario Nazionale per l'Informatica - CINI, via Orabona, 4 - 70125 Bari - Italy E-mail: vincenzo.pasquadibisceglie@uniba.it, raffaele.scaringi@uniba.it, annalisa.appice@uniba.it, giovanna.castellano@uniba.it, donato.malerba@uniba.it*

offer a more versatile data representation. In addition, graphs can inherit generic domain knowledge, that cannot be encoded in images. Finally, the use of graphs enhances a higher degree of comprehension among domain experts, making the information more interpretable and accessible [12]. In particular, several PPM studies (e.g., [13], [14], [15], [16]) have recently explored the performance of various deep neural architectures for graphs, e.g., Graph Neural Networks (GNNs), Graph Recurrent Neural Networks (GRNNs) and Graph Attention Networks (GATs), that have been trained for process monitoring. However, previous PPM studies consider homogeneous graphs to mainly represent the activity control-flow and eventually use external data synopsis to encode the multi-view information associated with event nodes. Although a few recent studies have started the exploration of deep neural models for heterogeneous graphs within a multi-view learning framework [17], [18], to the best of our knowledge, no previous PPM study investigates the use of heterogeneous graphs to represent information recorded in different views of an event log with different types of nodes. In addition, the priority of previous PPM studies leveraging graph-based deep neural methods was mainly to provide accurate predictions of future states of running traces. On the other hand, easier-to-explain predictive models are becoming increasingly desirable in PPM applications [7], [8]. This motivates our interest, in this article, towards recent general developments achieved to explain decisions regarding heterogeneous graphs (e.g., [19], [20]) and, hence, exploring solutions to explain PPM decisions regarding graph representations of event traces.

Problem formulation. Moving from considerations reported above, we address the PPM problem of the next-activity prediction with the aim of training a deep neural model to predict the executing activity of the next event in a running trace and explain which characteristics of the running mainly contribute to the decision. The prediction is performed by considering prefix traces, i.e. the sequence of multi-view information enclosed in past events recorded from the beginning of the current trace. We introduce a heterogeneous graph-based representation of prefix traces, which translates the multi-view information recorded in a given prefix event trace into a heterogeneous graph, where nodes represent different event characteristics, and links denote interactions among them. We formulate a new PPM graph-based method, called PROPHET (explainable Predictive heteROgeneous graPH nEural neTworks), that trains the PPM model to make the next-activity predictions of the running trace of a business process by resorting to a supervised predictive graph-based analytic with a labeled dataset that is extracted from the given event log recording historical traces of the same business process. Specifically, PROPHET trains a heterogeneous GNN [21], that applies the GAT layer [22]. Explainability is achieved thanks to the heterogeneous version of GNN Explainer algorithm [23]. This algorithm, which was originally formulated to provide an explanation of the effect of features on decisions for each type of nodes and links, has been here modified to deliver comprehensive and granular levels of explainability for each distinct node and link into the graph. This is to facilitate informed decision-making based on local predictions.

Contributions. The main scientific contributions of this article are as follows: (1) The design of an event data engineering scheme capable of mapping event traces recorded in an event log into heterogeneous graphs that directly embrace the multi-view representation of event data. (2) The use of a heterogeneous graph learning approach that integrates the attention mechanism implemented in the GAT layer. (3) The extension and application of the heterogeneous GNN Explainer algorithm to disclose how specific event and trace characteristics, as well as event relationships, influence the local decisions of the proposed predictive model. (4) Extensive focus on model evaluation, performed considering multiple real-world event logs collected in several domains and examining the capability of the method to both achieve accuracy comparable to related deep multi-view learning and graph learning approaches presented in the recent PPM literature and disclose useful explanations of the model behavior in terms of the most informative inputs.

Outline. The paper is organized as follows. Section 2 illustrates the recent background work in the PPM literature referred to PPM methods for next-activity prediction, XAI, and graph-based neural networks in PPM. The literature overview describes the advantages and limits of related methods to better highlight new achievements of this article. Section 3 reports the preliminary concepts of business process executions, and introduces the encoding schema used in this article to represent a prefix trace as a heterogeneous graph, as well as the next-activity problem formulated according to the adopted heterogeneous graph representation. Section 4 describes the training stage of the proposed PROPHET method. The experimental study is illustrated in Section 5. The experimentation evaluates the performance of the proposed method along the dimensions: accuracy, efficiency and explainability. Finally, Section 6 summarises the objectives of this research and draws conclusions.

2 RELATED WORK

The literature overview in the paper is organized on three main fronts. First, recent studies on next-activity prediction using deep learning in PPM are discussed in Section 2.1. Then, the use of eXplainable AI (XAI) techniques in PPM is revised in Section 2.2. Finally, the use of graph-based neural networks in PPM is examined in Section 2.3.

2.1 Next-activity prediction

The recent literature has seen the proliferation of different deep neural models to address the next-activity prediction task. Due to the sequential nature of event traces, the majority of the deep neural PPM approaches are based on LSTM neural networks. In [24], the authors describe a PPM method that trains an LSTM architecture to predict sequences of next-activities. This method takes into account the activity and role by resorting to a pre-trained embedding representation of the categorical information. Recently, the authors of [25] have described a multi-view approach that considers activity, resource, and timestamp by training an LSTM model equipped with an attention layer. The authors of [26] describe a Transformer-based method to predict the next-activity of an event trace. This method considers information recorded in the activity view of an event log. Differently, the authors of [1] define a multi-input LSTM network that account for the information recorded in any view of an event log to train a deep neural model for next-activity prediction. This study shows that a fully multi-view approach, that does not limit to process information recorded in the mandatory activity and timestamp views, can gain accuracy in performing predictive process monitoring. Similarly, authors of [3] describe a multi-input LSTM neural network coupled with the Layer-Wise Relevance Propagation method for the next-activity prediction task.

Despite the majority of recent PPM methods handle event traces as data sequences and use LSTM neural networks to train predictive process models, an emerging research direction has recently boosted the use of images to represent event trace data and the adoption of Computer Vision methods in PPM. For example, a color imagery representation of the information recorded in the activity, resource and timestamp view of a log is described in [5]. This study trains a CNN model to predict the next-activity of a running trace. More recently, the authors of [7] have described an approach to represent information of event traces recorded in multiple views of an event log as patches of a color image and used this patch-based representation within a multi-view learning scheme combined with ViTs.

Several methods described above achieve good accuracy performance in various problems by supporting a multi-view approach ([1], [3], [7]). However, they are not equipped with explainability techniques except for [3], [7]. Explainability is supported by [25] for a deep neural model trained with activities, resources and timestamps. The explainability of these methods is examined in Section 2.2.

2.2 eXplainable AI

The majority of the XAI studies in PPM use post-hoc explainers that are model-agnostic. For example, the authors of [8] use SHAP to explain how event characteristics have an effect on decisions produced with LSTM and CatBoost models. In [27], the authors use SHAP to explain how the effect of event characteristics on decisions changes over time due to concept drifts. In [28], the authors examine the properties of interpretability and faithfulness in the field of process outcome prediction by comparing different classifiers, which have been complemented with post-hoc XAI techniques. In [29], LIME is used to understand and analyze which factors mainly influence a Deviance Prediction Model trained to predict an event trace as deviant or normal.

On the other hand, some recent PPM studies have investigated intrinsic explainable deep neural models to directly produce model explanations. Following this direction, the authors of [30] adopt a Gated Graph Neural network to discover the contribution of the different activities to the PPM model predictions. A comprehensible PPM model is designed in [31] for outcome prediction problems. This model relies on inherently interpretable fuzzy rules learned from event traces. Both [25] and [26] use attention mechanisms to allow a deep neural model to highlight the most important information driving decisions. However, both these studies force apriori-defined views at the input level. On the other hand, the authors of [3] add a Layer-Wise Relevance Propagation (LRP) layer to an LSTM network that can process any categorical and numerical data recorded in an event log. The LRP layer can identify the past activities and data that have a higher effect on the next-activity prediction. This study illustrates some examples of explanations of decisions produced locally for some sample running traces. Finally, the authors of [7] use the self-attention information that is available through a ViT model trained for the next-activity prediction by accounting for information recorded in all views of an event log. In this study, the attention information, retrieved through the roll-out method, explains the effect of views and event characteristics on the PPM model's reasoning. Similarly to [3], this study shows some examples of local explanations of decisions produced for some sample running traces. In addition, it shows global explanations disclosing information about which characteristics (views), among the ones recorded in the event logs, have a higher effect on the overall decisions of the trained models.

The explainable PPM method described in [7] is the closest to the proposed method. The methods described in both [7] and our study are developed according to a deep multi-view learning approach that uses of information recorded in all views of an event log. In addition, both methods integrate explainable elements in their approach. However, some significant differences exist in how and which explanations are provided. The method in [7] represents event traces as multi-patch color images. It trains a PPM model through a ViT and provides explanations retrieved through the attention information. Instead, we represent event traces as heterogeneous graphs. We train a PPM model through a heterogeneous GNN and use a modified version of the GNNExplainer algorithm to explain decisions. In particular, [7] provides global and local explainable information on which views of the process and events of the trace mainly influenced the prediction. The method described in this study provides the same global and local explainable information as [7]. In addition, to the best of our knowledge, this is the first explainable PPM method that explains the effect that relationships between characteristics belonging to either the same event or different events have on the decisions. These relationships remain unexplained in [7].

2.3 Graph Neural Networks

Recent research in PPM has started the investigation of graph-based representations used to encode event trace information. These representations leverage the structural and temporal information available in an event log without disregarding the order in which the events appear in a trace and giving the opportunity to model explicitly possible repetitions. In addition, they boost the use of GNN-based approaches in next-activity prediction problems.

In [14], the authors describe one of the pioneering studies to train a Graph Convolutional Network (GCN) for forecasting the next-activity of a running event trace. This study accounts for activity and timestamp information recorded in the event log. Specifically, the proposed method extracts the Directly Follows Graph (DFG) of an event log by representing the unique activity names along with their frequencies as nodes and the activity direct-follow relations along with their frequencies as links. In addition, they represent each running event trace by a matrix of timestamp-based features (i.e., the time since the previous event in the trace, the time since the trace started, the time since midnight, and the day of the week for the event). Specifically, this matrix contains a vector of time-stamped features for each distinct activity type. If an activity name occurs twice in the trace, then the feature vector of the last occurrence of the activity type in the trace is recorded in the matrix. Notably, the study investigates the effect of different encoding methods (e.g., binary, weighted) used in the adjacency matrix extracted by the DFG on the accuracy of the next-activity predictions.

In [13], the authors introduce a graph encoding technique that represents (time-ordered) events (with multi-view information) of an event trace as graph nodes and models the links between the trace's events according to the activity direct-following relationship. In particular, links connecting nodes associated with different activity names are labeled with the "follows" label. Links connecting nodes associated with the same activity name are labeled with the "repeat" label. For each event, both activity, resource, and timestamp are considered. A Gated Graph Neural Network is, finally, trained for the next-activity prediction task from the collection of running trace graphs extracted from the log.

The authors of [15] transform a running trace into an Instance Graph (IG) by accounting for the direct causal relationships (instead of the direct-follow relationships) between the activity names of trace events as they have been encoded in the Petri Net model of the event log. Each IG, which is initially constructed using the activity information, is subsequently enriched by incorporating information recorded in multiple views of the event log. This is realized by linking each node of the IG with a vector of selected features. In addition, timestamp-based features are computed according to the causal relationships. A GCN is, finally, trained for the next-activity prediction task from the collection of enriched IGs extracted from the event log.

Similarly to [15], the authors of [16] consider the Petri Net discovered from the entire log. Their study represents the place graph of the Petri Net as an adjacency matrix and replays each running event trace on the Petri Net by generating a node feature matrix to represent the activations of the places and transitions for that running trace. This study considers an attribute feature matrix containing additional information, such as the event-related temporal data recorded in the multiple views of the event log. Both the adjacency matrix and the node feature matrix are processed to train a Graph Recurrent Neural Network (GRNN). The output of the GRNN is concatenated to the attribute feature matrix and fed into an LSTM, whose output is passed to a softmax classifier to yield the next-activity prediction.

Both [15] and [16], similarly to the method proposed in this study, realize a multi-view learning approach. However, both require a (time-consuming) process discovery step to extract the global information used to represent the running event traces as graphs. Differently, our method is based on local trace information to extract the trace graph structure. This misses the global abstraction of the process model, but results in a lighter encoding process. On the other hand, graph learning-based PPM methods reported above adopt a homogeneous graph structure that is often enriched with an external data synopsis that encodes the multi-view information associated with nodes. To the best of our knowledge, this is the first PPM study that investigates the use of a heterogeneous graph to represent information recorded in different views of a log with different types of nodes. The homogeneous graph representations commonly adopted in the PPM literature allow us to express direct-follow or causal relationships between events and to encode intra-view relationships between different characteristics at the same event. Instead, the heterogeneous graph encoding proposed in this study enriches the ability to represent these relationships, with the inter-view relationships that arise among different characteristics observed at different events. For example, it can express explicitly that two different activities are executed by the same resource in a running event trace. A further novel contribution of this study is the use of a GNN with a GAT layer in a multi-relational setting, where [22] shows empirically that a GAT can outperform several GNN architectures. Finally, the literature studies reported above focus on the accuracy gained in the next-activity predictions produced under the umbrella of the GNN-based methods. Differently, in this study, we have the objective of contributing to bridging the gap between accuracy and explainability in PPM models. So, we adapt the GNNExplainer algorithm to the proposed approach to explain the effect that specific views, intra-view events and inter-view events may have on the GNN reasoning.

3 Preliminary concepts

In this section, we first report basic concepts (i.e., events, traces, prefix traces) of business process executions (Section 3.1). Then we illustrate the encoding schema adopted in this study to represent a running event trace (prefix trace) as a heterogeneous graph (Section 3.2). Finally, we formulate the next-activity prediction problem accounting for the proposed graph representation of prefix traces (Section 3.3).

3.1 Event, Trace, Prefix trace

Given a business process, an event trace provides a chronological description of the process execution through a sequence of events. An event is a structured entity with two mandatory characteristics, i.e., the activity itself and the timestamp at which the activity has been performed. The event may encompass several optional characteristics, such as the resource that initiated the activity or the cost of the activity execution. Following this definition, an event is portrayed from multiple views, where a view describes each event based on a specific characteristic. Hence, each event is represented in two mandatory views associated with the activity A and the timestamp T, respectively, and in m supplementary views, V_j with $j = 1, \ldots, m$ associated with optional event characteristics. Finally, an event trace may be associated with n supplementary global characteristics, C_h with $h = 1, \ldots, n$, which describe general trace information such as the age of a patient at the time of her hospitalization. Let $\mathcal{A}$ be the set of activity names, $\mathcal{S}$ be the set of trace identifiers, $\mathcal{T}$ be the set of timestamps, $\mathcal{V}_j$ with $1 \leq j \leq m$ be the set of values in the j-th characteristic of an event, while $\mathcal{C}_h$ $1 \leq h \leq n$ be the set of values in the h-th global characteristic of an event trace. For simplicity, let us denote $\mathbf{C} = [C_1, \ldots, C_n]$ – the vector of global trace characteristics , $\mathcal{C} = \mathcal{C}_1\times,\ldots,\times\mathcal{C}_n$ – the domain of $\mathbf{C}$, $\mathbf{V} = [A, T, V_1, \ldots, V_m]$ – the vector of local characteristics and $\mathcal{V} = \mathcal{A} \times \mathcal{T} \times \mathcal{V}_1 \times \ldots \times \mathcal{V}_m$ – the domain of $\mathbf{V}$.

Definition 1 (Event). Given the event universe $\mathcal{E} = \mathcal{S} \times \mathcal{V}$, an event $e \in \mathcal{E}$ is a tuple $e = (s, a, t, v_1, \ldots, v_m)$ that represents the occurrence of activity a in trace s at timestamp t with characteristics $v_1, \ldots, v_m$.
Let us introduce the functions $\pi_\mathcal{S}\colon \mathcal{E} \mapsto \mathcal{S}$ such that $\pi_\mathcal{S}(e) = s$, $\pi_\mathcal{A}\colon \mathcal{E} \mapsto \mathcal{A}$ such that $\pi_\mathcal{A}(e) = a$, $\pi_\mathcal{T}\colon \mathcal{E} \mapsto \mathcal{T}$ such that $\pi_\mathcal{T}(e) = t$, $\pi_{\mathcal{V}_j}\colon \mathcal{E} \mapsto \mathcal{V}_j$ such that $\pi_{\mathcal{V}_j}(e) = v_j$ and $j = 1, \ldots, m$.

Definition 2 (Trace). Let $\mathcal{E}^*$ denote the set of all possible sequences on $\mathcal{E}$. Let us introduce the function $\pi_\sigma\colon \mathcal{E}^* \times \mathbb{N} \mapsto \mathcal{E}$ such that $\pi_\sigma(\sigma, i) = e_i$. Thus, a trace is a pair $(\sigma, \mathbf{c}) \in \mathcal{E}^* \times \mathcal{C}$ where: (1) $\sigma = \langle e_1, \ldots, e_t\rangle \in \mathcal{E}^*$ is an event sequence so that (1.a) $\forall i = 1, \ldots, t, \exists e_i \in \mathcal{E}$ such that $\pi_\sigma(\sigma, i) = e_i$ and $\pi_\mathcal{S}(e_i) = s$, and (1.b) $\forall i = 1, \ldots, t-1, \pi_\mathcal{T}(e_i) \leq \pi_\mathcal{T}(e_{i+1})$. (2) $\mathbf{c} = (c_1, \ldots, c_n) \in \mathcal{C}$ is the vector of the values of the optional global trace characteristics.

Definition 3 (Prefix event sequence). Given a sequence $\sigma \in \mathcal{E}^*$, the prefix event sequence $\sigma^k = \langle e_1, \ldots, e_k\rangle$ is the event sub-sequence of σ starting from the beginning of σ, with $1 \leq k = |\sigma^k| < |\sigma|$.

Definition 4 (Prefix trace). Given the trace $(\sigma, \mathbf{c}) \in \mathcal{E}^* \times \mathcal{C}$, its prefix trace with length k is the pair $(\sigma^k, \mathbf{c})$ where σ^k is the the prefix event sub-sequence of σ with length equal to k, while $\mathbf{c}$ is the vector of global characteristic values recorded in the trace.

A trace describes the event sequence of a complete (i.e., started and ended) process instance, while a prefix trace describes the event sequence of a process instance in execution (running trace). The activity $\pi_\mathcal{A}(e_{k+1}) = a_{k+1}$ corresponds to the next-activity of σ^k in σ, i.e., $next(\sigma^k) = \pi_\mathcal{A}(e_{k+1})$ with $e_{k+1} = \pi_\sigma(\sigma, k+1)$.

Definition 5 (Multiset of labeled prefix traces). Let $\mathcal{LOG} \in \mathcal{B}(\mathcal{E}^* \times \mathcal{C})$ be an event log, $\mathcal{P} \subseteq \mathcal{B}(\mathcal{E}^* \times \mathcal{C} \times \mathcal{A})$ is the multiset of all prefix traces extracted from traces recorded in $\mathcal{LOG}$. Each prefix trace is labeled with the next-activity associated with the prefix sequence in the corresponding trace so that $\mathcal{P} = \{(\sigma^k, \mathbf{c}, a_{k+1}) | (\sigma, \mathbf{c}) \in \mathcal{LOG}, 1 \leq k < |\sigma|, a_{k+1} = \pi_\mathcal{A}(e_{k+1})\}$.

3.2 Heterogeneous graph encoding

A graph is a smart data structure that allows the representation of entity characteristics and entity relationships. In this study, we introduce a graph encoding strategy to transform a prefix trace into a heterogeneous graph.

Definition 6 (Graph). Let $\mathcal{G}$ denote the set of all graphs. A graph $g \in \mathcal{G}$ is an ordered pair $(\mathcal{N}, \mathcal{L})$, where $\mathcal{N}$ is the set of nodes, while $\mathcal{L}$ is the set of weighted links between nodes, i.e., $\mathcal{L} \subseteq \mathcal{N} \times \mathcal{N} \times \mathbb{R}$. Each link $(n_1, n_2, w) \in \mathcal{L}$ is a link between nodes $n_1 \in \mathcal{N}$ and $n_2 \in \mathcal{N}$ associated with a numerical value $w \in \mathbb{R}$, called weight.
Let us introduce the function $\omega\colon \mathcal{L} \mapsto \mathbb{R}$ such that $\omega(l) = w$ is the weight associated with link l.

Let $\mathcal{T_N}$ be the set of distinct node types and $\mathcal{T_L}$ be the set of distinct link types appearing in a graph g, with $\mathcal{T_L} \subseteq \mathcal{T_N} \times \mathcal{T_N}$. In a *heterogeneous graph*, $|\mathcal{T_N}| > 1$ and $|\mathcal{T_L}| > 1$.

Definition 7 (Node type mapping). Let us consider a graph $g = (\mathcal{N}, \mathcal{L})$. Each node $n \in \mathcal{N}$ is associated with the node type mapping function $\eta\colon \mathcal{N} \mapsto \mathcal{T_N}$ that assigns a node $n \in \mathcal{N}$ to its node type $\eta(n) \in \mathcal{T_N}$.

For each node type $t_n \in \mathcal{T_N}$, let us consider $\mathcal{N}_{t_n} = \{n \in \mathcal{N} | \eta(n) = t_n\}$, that is the set of nodes with type t_n.

Definition 8 (Node type-based feature association). Given a node type $t_n \in \mathcal{T_N}$, each node $n \in \mathcal{N}_{t_n}$ is associated with the type-based node feature vector mapping function $\phi_{t_n}\colon \mathcal{N}_{t_n} \mapsto \mathbb{R}^{d_{t_n}}$ that assigns n with a d_{t_n}-dimensional numerical feature vector $\phi_{t_n}(n)$.

In the following, we define concepts behind the encoding strategy adopted to represent a prefix trace $(\sigma^k, \mathbf{c})$ recorded in $\mathcal{LOG}$ as a heterogeneous graph $g = (\mathcal{N}, \mathcal{L})$.

Regarding the graph encoding strategy, we transform the distinct values of the local event characteristics recorded in σ^k and the vector of the values of the global trace characteristics recorded in $\mathbf{c}$ into distinct nodes of the node set $\mathcal{N}$. Each node records the embedded representation of the mapped characteristic value. In particular, every local event characteristic $V \in \mathbf{V}$ is one-to-one associated with an event-based node type $t_V \in \mathcal{T_N}$, while the global trace characteristic vector $\mathbf{C}$ is one-to-one associated with the trace-based node type $t_\mathbf{C} \in \mathcal{T_N}$. Following this mapping, a prefix trace is transformed into a heterogeneous graph that contains $m+3$ distinct node types (i.e., $|\mathcal{T_N}| = m+3$) namely a node type associated with the mandatory activity characteristic, a node type associated with the mandatory timestamp characteristic, a node type associated with each one of the m optional event characteristics recorded in the event sequence of the prefix trace and a node type associated with the vector of global trace characteristics.

Formally, given an event characteristic $V \in \mathbf{V}$, let us introduce the function $\gamma_V\colon \mathcal{E} \mapsto \mathcal{N}$ that maps the value of V recorded in an event $e \in \sigma^k$ into a node $\gamma_V(e) = n$ so that $\eta(n) = t_V$ and $\phi_{t_V}(n) = embedding(\pi_V(e))$, where $embedding(\pi_V(e))$ denotes the embedded representation of the value $\pi_V(e)$. In the following, we formulate the mapping function $\Gamma^*_\mathcal{E}$ to transform the distinct values recorded in the event characteristics of a prefix trace into the corresponding set of event-based nodes.

Definition 9 (Mapping local event characteristics of the event sequence of a prefix trace into a set of event-based nodes). Given the event sequence $\sigma^k \in \mathcal{E}^*$ of a prefix trace $(\sigma^k, \mathbf{c}) \in \mathcal{E}^* \times \mathcal{C}$, let us introduce the function $\Gamma_{\mathcal{E}^*}\colon \mathcal{E}^* \mapsto \mathcal{B}(\mathcal{N})$ that transforms σ^k into the node set $\Gamma_{\mathcal{E}^*}(\sigma^k) = \bigcup_{e\in\sigma^k} \bigcup_{V\in\mathbf{V}} \{\gamma_V(e)\}$.

Similarly, we define the mapping function $\Gamma_\mathcal{C}$ to transform the vector of the values of the global trace characteristics into the corresponding trace-based node of the graph.

Definition 10 (Mapping the vector of global trace characteristics of a prefix trace into a trace-based node). Given the vector of the values of the global trace characteristics $\mathbf{c} \in \mathcal{C}$ of a prefix trace $(\sigma^k, \mathbf{c}) \in \mathcal{E}^* \times \mathcal{C}$, let us introduce the function $\Gamma_{\mathcal{C}} \colon \mathcal{C} \mapsto \mathcal{N}$ so that $\Gamma_C(\mathbf{c}) = n$, with $\eta(n) = t_{\mathbf{C}}$ and $\phi_{t_{\mathbf{C}}}(n) = embedding(\mathbf{c})$.

Regarding the links of the graph encoding strategy, we identify three types of links: (1) links pertaining intra-view relationships on the same local event characteristic observed in different events; (2) links pertaining inter-view relationships involving two different local event characteristics measured in the same event; (3) links that establish a relationship between each specific local event characteristic and the vector of the global trace characteristics.

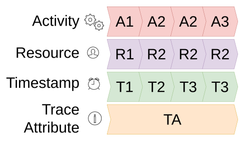
Definition 11 (Mapping the intra-view relationships of a prefix trace into links connecting event-based nodes of the same type). Let $\Psi_{intra} \colon \mathcal{E}^* \mapsto \mathcal{B}(\mathcal{L})$ be a function that maps the event sequence σ^k of a prefix trace $(\sigma^k, \mathbf{c}) \in \mathcal{E}^* \times \mathcal{C}$ into a set of links connecting pairs of event-based nodes, belonging to the same view, which are present in σ^k as consecutive events $\langle e_i, e_{i+1} \rangle \subseteq \sigma^k$. Formally, $\Psi_{intra}(\sigma^k) = \{l \in \mathcal{L}, l = (n_1, n_2, w) | \exists \langle e_i, e_{i+1} \rangle \subseteq \sigma^k, V \in \mathbf{V},\ so\ that\ \eta(n_1) = \eta(n_2) = t_V, \gamma_V(e_1) = n_1, \gamma_V(e_2) = n_2\ and\ w = |\mathbb{F}_{intra}|\}$, with $\mathbb{F}_{intra} = \{\langle e_i, e_{i+1} \rangle \subseteq \sigma^k | \exists V \in \mathbf{V}\ so\ that\ \eta(n_1) = \eta(n_2) = t_V, \gamma_V(e) = n_1, \gamma_V(e) = n_2\}$.

Definition 12 (Mapping the inter-view relationships of a prefix trace into links connecting event-based nodes of different types). Let $\Psi_{inter} \colon \mathcal{E}^* \mapsto \mathcal{B}(\mathcal{L})$ be a function that maps the event sequence σ^k of a prefix trace $(\sigma^k, \mathbf{c}) \in \mathcal{E}^* \times \mathcal{C}$ into a set of links connecting each pair of event-based nodes corresponding to two distinct event characteristics observed simultaneously into at least an event $e \in \sigma^k$. Formally $\Psi_{inter}(\sigma^k) = \{l \in \mathcal{L}, l = (n_1, n_2, w) | \exists e \in \sigma, V_i \in \mathbf{V}, V_j \in \mathbf{V},\ so\ that\ V_j \neq V_i, \eta(n_1) = t_{V_i}, \eta(n_2) = t_{V_j}, \gamma_{V_i}(e) = n_1, \gamma_{V_j}(e) = n_2\ and\ w = |\mathbb{F}_{inter}|\}$, with $\mathbb{F}_{inter} = \{e \in \sigma^k | \exists V_i \in \mathbf{V}, V_j \in \mathbf{V}\ so\ that\ V_j \neq V_i, \eta(n_1) = t_{V_i}, \eta(n_2) = t_{V_j}, \gamma_{V_i}(e) = n_1, \gamma_{V_j}(e) = n_2\}$

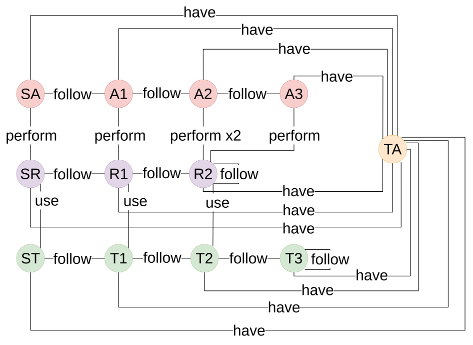
Definition 13 (Mapping the relationships between each local event characteristic and the vector of global trace characteristics into links connecting the event-based nodes to the trace-based node). Let $\Psi_{global} \colon \mathcal{E}^* \times \mathcal{C} \mapsto \mathcal{B}(\mathcal{L})$ be a function that maps a prefix trace $(\sigma^k, \mathbf{c}) \in \mathcal{E}^* \times \mathcal{C}$ to a set of links connecting the event-based nodes to the global trace nodes. Formally, $\Psi_{global}(\sigma^k, \mathbf{c}) = \{l \in \mathcal{L}, l = (n_1, n_2, w) | \exists V \in \mathbf{V},\ so\ that\ \eta(n_1) = t_V, \eta(n_2) = t_{\mathbf{C}}, w = 1\}$.

Definition 14 (Labeled Prefix Trace Graph Mapping function). Let us define the graph mapping function $\Gamma \colon \mathcal{E}^* \times \mathcal{C} \mapsto \mathcal{G}$ that maps a given prefix trace $(\sigma^k, \mathbf{c}) \in \mathcal{E}^* \times \mathcal{C}$ to a heterogeneous graph $g = (\mathcal{N}, \mathcal{L}) = \Gamma(\sigma^k, \mathbf{c})$ so that (1) $\mathcal{N} = \Gamma_{\mathcal{E}^*}(\sigma^k) \cup \{\Gamma_C(\mathbf{c})\}$ and (2) $\mathcal{L} = \Psi inter(\sigma^k) \cup \Psi intra(\sigma^k) \cup \Psi global(\sigma^k, \mathbf{c})$.

Figure 1 shows an example of the transformation of a prefix trace into a heterogeneous graph. The prefix trace (Fig. 1.a) is composed of a sequence of four events with values of activity, resource, and timestamp recorded in each event, and a single global trace characteristic that assumes a unique value for the entire prefix trace. The heterogeneous graph (Fig. 1.b) associated with the prefix trace is populated with 12 nodes, namely four nodes of type "activity-node" to record activities SA, A1, A2 and A3; three nodes of type "resource-node" to record resources SR, R1 and R2; four nodes of type "timestamp-node" to record timestamps ST, T1, T2 and T3; one node of type "trace-node" to record the value of the global trace characteristic TA.



(a) Prefix trace



(b) Heterogenous graph

Figure 1: Example of conversion of a prefix trace into a heterogeneous graph.

For each event characteristic, we introduce a starting node (in the example, SA for Activity, SR for Resource, and ST for Timestamp) to represent the characteristic within the starting event of the prefix trace.[1] In addition, the graph contains four "following" links between nodes of type "activity-node", three "following" links between nodes of type "resource-node", four "following" links between nodes of type "timestamp-node", four "performing" links between nodes of type "activity-node" and nodes of type "resource-node", four "using" links between nodes of type "resource-node" and nodes of type "timestamp-node" and eleven "having" links between each node with types "activity-node" or "resource-node" or "timestamp-node" and the trace node of the graph. The "following" links express intra-view relationships. This kind of links may describe cycles (e.g., A2 following A2). Both "performing" and "using" links express inter-view relationships. Finally, "having" links denote the relationships between each local event characteristic and the vector of global trace characteristics.

3.3 Graph-based next-activity prediction problem

Given a heterogeneous graph $g \in \mathcal{G}$ such that g represents the prefix trace $(\sigma^k, \mathbf{c})$ with length k of a longer trace $(\sigma, \mathbf{c})$, recorded in an event log $\mathcal{LOG}$ for which we do not know the actions in the rest of the sequence $\langle e_{k+1}, e_{k+2}, \ldots, e_t \rangle$. Let us consider a function $F \colon \mathcal{G} \mapsto \mathcal{A}$, such that $F(g)$ predicts the expected next-activity a_{k+1} of the graph g. Based on these premises, we frame the *next-activity prediction* task as a multi-class, heterogeneous graph classification problem.

1. The starting node is added a prefix trace to be able to express the "following" relationship also in a graph originated from a prefix trace composed of a single event.

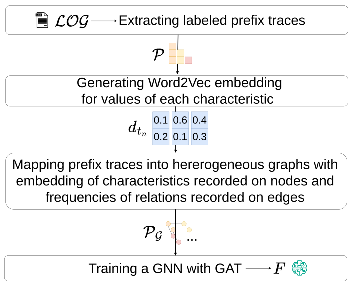

$\mathcal{LOG} \longrightarrow$ Extracting labeled prefix traces
$\mathcal{P}$
Generating Word2Vec embedding for values of each characteristic
d_{t_n}
0.1 0.6 0.4
0.2 0.1 0.3
Mapping prefix traces into hererogeneous graphs with embedding of characteristics recorded on nodes and frequencies of relations recorded on edges
$\mathcal{P}_{\mathcal{G}}$ ...
Training a GNN with GAT $\longrightarrow F$

Figure 2: PROPHET pipeline

According to this formulation F can be trained by resorting to a graph-based learning technique from the multiset $\mathcal{P}_{\mathcal{G}}$ of the labeled heterogeneous graphs representing the multiset $\mathcal{P}$ of labeled prefix traces recorded in $\mathcal{LOG}$ and labeled with the next-activity in the corresponding traces.

Definition 15 (Multiset of labeled heterogeneous graphs). Let $\mathcal{LOG}$ be an event log, $\mathcal{P} \subseteq \mathcal{E}^* \times \mathcal{C} \times \mathcal{A}$ be the the multiset of labeled prefix traces recorded in $\mathcal{LOG}$. $\mathcal{P}_{\mathcal{G}} \subseteq \mathcal{B}(\mathcal{G} \times \mathcal{A})$ is the multiset of heterogeneous graphs representing all prefix traces extracted from traces recorded in $\mathcal{LOG}$ as heterogeneous graphs. Each heterogeneous graph is labeled with the next-activity associated to the prefix sequence in the corresponding trace so that $\mathcal{P}_{\mathcal{G}} = \{(g, a_{k+1}) \in \mathcal{G} \times \mathcal{A} \mid g = \Gamma(\sigma^k, \mathbf{c}),\ \forall(\sigma^k, \mathbf{c}, a_{k+1}) \in \mathcal{P}\}$.

In this work, given the multiset $\mathcal{P}_{\mathcal{G}}$ of labeled heterogeneous graphs extracted from an event log $\mathcal{LOG}$, $\mathcal{P}_{\mathcal{G}}$ is processed to learn a next-activity prediction function F through a GAT-equipped GNN architecture.

4 Proposed Approach

In this section we describe PROPHET, an explainable, graph-based, PPM approach for next-activity prediction.

4.1 PROPHET approach

Figure 2 describes the main phases of the proposed approach, namely, extracting the multi-set of labeled prefix traces recorded in an input event log $\mathcal{LOG}$, representing the extracted prefix traces as heterogeneous graphs and estimating parameters of an output GAT model that is trained for next-activity prediction.

The first phase takes $\mathcal{LOG}$ as input and generates the multi-set of labeled prefix traces $\mathcal{P}$ as output. This phase is composed of three steps: 1) Transforming timestamps and discretizing numeric information. 2) Extracting labeled prefix traces. 3) Generating the embeddings of all the extracted prefix trace characteristics. According to the multi-view formulation illustrated in Section 3.1, an event log records both global trace characteristics and local event characteristics. Both types of characteristics may describe either categorical (e.g., activity, resource) or numerical (e.g., timestamp, cost) information. The timestamp information recorded in an event is transformed into the time in seconds passed from the beginning of the trace. Numerical characteristics are converted into categorical by applying the equal-frequency discretization algorithm as in [7]. The number of discretization bins of a characteristic is set equal to the average number of distinct categories in the original categorical characteristics recorded in $\mathcal{L}$. After this step, $\mathcal{LOG}$ contains all characteristics in the categorical format. Subsequently, the multiset $\mathcal{P}$ is extracted from $\mathcal{LOG}$ according to Definition 5. Finally, the Continuous-Bag-of-Words (CBOW) architecture of the Word2Vec scheme [32] is adopted to transform each categorical characteristic of a prefix trace collected in $\mathcal{P}$ into a numeric representation. CBOW is an embedding approach that was originally formulated in the field of natural language processing and as recently used in several PPM approaches (e.g., [4], [7], [33]). It uses a feed-forward neural network to predict a target value from the neighbored context. In this work, a distinct CBOW architecture is trained for each characteristic recorded in the prefix traces. After this phase, $\mathcal{P}$ contains the labeled prefix traces which record the values of their characteristics in the recovered embedding representations.

Subsequently, $\mathcal{P}$ is converted into the multiset of labeled heterogeneous graphs $\mathcal{P}_{\mathcal{G}}$. In particular, each labeled prefix trace $(\sigma^k, \mathbf{c}, a_{k+1}) \in \mathcal{P}$ is transformed into a labeled heterogeneous graph $(g, a_{k+1}) \in \mathcal{P}_{\mathcal{G}}$ following Definition 14. For each event $e \in \sigma^k$, every event characteristic recorded in e is mapped into a graph node that contains the embedding representation of the characteristic value (as described in Definitions 8 and 9) The embedding is the one computed with the corresponding CBOW model. The type of the node is that one-to-one associated with the mapped characteristic. At the same time, based on Definition 10, the embeddings of the values the global trace characteristics collected in $\mathbf{c}$ are recorded into the trace-based node of the graph. Once the nodes of g have been generated, the links between the created nodes are produced following the Definitions: 11, 12 and 13. Regarding inter-view links (which pertain nodes of different types), in this work, we generate the inter-view links considering the graph nodes whose types are associated with each pair of consecutive views in the vector $\mathbf{V}$ that represent the event characteristics recorded in $\mathcal{LOG}$.

Finally, a GNN equipped with a number L of GAT layers [22] is considered to learn a next-activity prediction function F from $\mathcal{P}_{\mathcal{G}}$. For each node type $t_n \in \mathcal{T}_{\mathcal{N}}$, let $\mathbf{X_{t_n}} \in \mathbb{R}^{|\mathcal{N}_{t_n}| \times d_{t_n}}$ denote the data feature matrix that is one-to-one associated with t_n so that every row of $\mathbf{X_{t_n}}$ records the embedding value recorded in every one of the nodes of g having type t_n. Let $\mathbf{X} = [\mathbf{X_{t_n}} | t_n \in \mathcal{T}_{\mathcal{N}}]$ be the list of data feature matrices associated with the distinct node types of $\mathcal{T}_{\mathcal{N}}$ which occur in g. For each link type $t_l = (t_a, t_b) \in \mathcal{T}_{\mathcal{L}}$, let $\mathbf{A_{t_l}} \in \{0,1\}^{|\mathcal{N}_{t_a}| \times |\mathcal{N}_{t_b}|}$ denote the adjacency matrix associated with t_l in $\mathcal{T}_{\mathcal{L}}$. Let $\mathbf{A} = [\mathbf{A_{t_l}} \in \{0,1\}^{|\mathcal{N}_{t_a}| \times |\mathcal{N}_{t_b}|} | t_l \in \mathcal{T}_{\mathcal{L}}]$ be the list of the adjacency matrices associated with the distinct link types of $\mathcal{T}_{\mathcal{L}}$ which occur in g. The GAT learns iteratively new node representations, differentiating the learning process concerning the link types that are represented in the graph, and aggregating information coming from each neighborhood. Formally:

$$\mathbf{H}^{(j+1)} = \{\mathbf{H_{t_n}}^{(j+1)} = \rho\left(AGG_{t_l=(t_a,t_n)} \mathbf{A_{t_l}}^{\top} \mathbf{H_{t_a}}^{(\mathbf{j})} \mathbf{W_{t_a}}^{(\mathbf{j})}\right), \\ \forall\ t_l = (t_a, t_n) \in \mathcal{T}_{\mathcal{L}}, \forall\ t_n \in \mathcal{T}_{\mathcal{N}}\}$$

where $\mathbf{H_{t_n}}^{(j)}$ is the representation learned for nodes of type t_n at the j-th layer, having the base case $\mathbf{H_{t_n}}^{(0)} = \mathbf{X_{t_n}}$;

ρ is a non linear activation function; $\mathbf{W_{t_n}}^{(j)}$ is a learnable matrix for projecting the embeddings in the target hidden space at the layer j, having $\mathbf{W_{t_n}}^{(j)} \in \mathbb{R}^{d_j \times d_{j+1}}$ and the base case $\mathbf{W_{t_a}}^{(0)} \in \mathbb{R}^{d_{t_a} \times d_1}$; and AGG is a general-purpose aggregation function (e.g., sum, mean, min, max). However, this first formulation assumes equal importance among all the nodes in a given neighborhood. To overcome this limitation, the authors of [22] propose a new graph convolution framework expressed as:

$$\mathbf{H}^{(j+1)} = \{\mathbf{H_{t_n}}^{(j+1)} = \rho\left(\sum_{t_l=(t_a,t_n)} smax\left(\mathbf{A_{t_l}} \odot \mathbf{E_{t_l}}^{(j)}\right)^\top \mathbf{H_{t_n}}^{(j)} W_{t_n}^{(j)}\right) \forall\, t_l = (t_a, t_n) \in \mathcal{T_L}, \forall\, t_n \in \mathcal{T_N}\},$$

where $smax()$ denotes the softmax. This new formulation introduces the attention mechanism expressed by the learnable matrix $\mathbf{E_{t_l}}^{(j)}$. Each value of this matrix is computed as $e_{xy} = \mathbf{a}_{(j)}^\top LeakyReLU\left(\mathbf{W_a}^{(j)} \cdot \left[h_x^{(j)} || h_y^{(j)}\right]\right)$ where $\mathbf{a}_{(j)}^\top \in \mathbb{R}^{d_j}$, $\mathbf{W_a}^{(j)} \in \mathbb{R}^{d_j \times 2d_j}$ is a learnable projection matrix, and $\left[h_x^{(j)} || h_y^{(j)}\right]$ is the concatenation between node embeddings referring to x and y such that $\eta(x) = t_a \wedge \eta(y) = t_n$. Note that $h_x^{(j)} = \mathbf{H_{t_a}}^{(j)}[x, *]$ and $h_y^{(j)} = \mathbf{H_{t_n}}^{(j)}[y, *]$.

As soon as all graph convolutions have been computed, each node is associated with a new representation that accounts for the semantics within the graph. To obtain a single representation of the input graph, all the node representations are averaged, to obtain the final graph embedding $h_g = \frac{1}{|\mathcal{N}|} \sum_{t_n \in \mathcal{T_N}} \sum_{i=0}^{|\mathcal{N}_{t_n}|} \mathbf{H_{t_n}}^{(J)}[i, *]$. Finally, the graph embedding h_g is fed into a classification head, to obtain the next-activity prediction.

According to [34], the time complexity of training the GNN model in PROPHET is defined as $O(E_{GNN} \times L \times \sum_{g=(\mathcal{N},\mathcal{L}) \in \mathcal{G}} |\mathcal{L}|)$, where L is the number of GAT layers and E_{GNN} is the number of epochs completed to train the GNN.

4.2 Graph explanation for PPM

To explain decisions of the GAT model, we adopt a variant of GNNExplainer [23], a post-hoc XAI algorithm formulated to explain decisions of GNN models by extracting insights on how each node and link contribute to a decision. In this study, we prefer this explainer to other post-hoc XAI algorithms (e.g., Grad) due to its unique capability to extract meaningful sub-graphs that mainly explain the decision produced for an input graph. The basic implementation of the aforementioned algorithm works as follows.

Given a heterogeneous graph $g = (\mathcal{N}, \mathcal{L})$, the prediction $\tilde{y} = F(g)$ can be equally written as $\tilde{y} = F(\mathbf{X}, \mathbf{A})$, where $\mathbf{X}$ and $\mathbf{A}$ are the list of data feature matrices and the list of adjacency matrices that are, respectively, associated with g. The basic GNNExplainer algorithm explains this decision by jointly computing the following outputs: a node feature mask $\hat{\mathbf{X}} = [\hat{\mathbf{X}}_{\mathbf{t_n}} \in [0,1]^{1 \times d_{t_n}}, \forall t_n \in \mathcal{T_N}]$ and a link mask $\hat{\mathbf{A}} = [\hat{\mathbf{A}}_{\mathbf{t_l}} \in [0,1]^{|\mathcal{N}_{t_a}| \times |\mathcal{N}_{t_b}|}, t_l = (t_a, t_b), \forall t_l \in \mathcal{T_L}]$. To this aim, the algorithm attempts to perturb the input graph g, composing a new "masked" graph $\hat{g} = (\hat{\mathcal{N}}, \hat{\mathcal{L}})$ so that $\mathbf{X}^m = [\mathbf{X_{t_n}}^m = \mathbf{X_{t_n}}^m \odot \hat{\mathbf{X}}_{\mathbf{t_n}}, \forall t_n \in \mathcal{T_N}]$ is the list of data feature matrices associated with $\hat{g}$. The higher the value of $\hat{\mathbf{X}}_{\mathbf{t_n}}[i]$, the higher the importance of the corresponding i-th piece of information on the prediction. Notably, $\hat{\mathbf{X}}_{\mathbf{t_n}}[i]$ is a unique value estimated within for the nodes of g with type t_n. On the other hand, the higher the value of $\hat{A}_{t_l}[i, j]$, the more important the corresponding link. So, the main issue of this basic algorithm is that it cannot identify the most important node subset in the graph. For this reason, we adapted GNNExplainer to estimate the most important subgraph for a given prediction. Specifically, we modified the shape of parameter $\hat{\mathbf{X}}_{\mathbf{t_n}} \in [0,1]^{|\mathcal{N}_{t_n}| \times 1}, \forall t_n \in \mathcal{T_N}$ keeping the rest of the algorithm unchanged comprising the element-wise multiplications to obtain the masked graph $\hat{g}$. In this way, the higher the value estimated for $\hat{\mathbf{X}}_{\mathbf{t_n}}[i]$ the more important the effect of the i node on the decision.

According to [23], the time complexity to explain a GNN decision provided for a single graph $g = (\mathcal{N}, \mathcal{L})$ is defined as $O\left(E_{EXP} \times L \times |\mathcal{L}|\right)$, where L is the number of GAT layers and E_{EXP} is the number of epochs completed to optimize both $\hat{\mathbf{X}}$ and $\hat{\mathbf{A}}$.

5 Experimental study

In this section, we describe the event logs and the experimental set-up adopted for evaluating the accuracy and explainability performance of PROPHET, the implementation details, and the achieved results.

5.1 Event logs and experimental set-up

We used nine real-life event logs that are all available on the 4TU Centre for Research [2], except for SP2020 that is available on Zenodo[3]. These logs contain event traces collected by monitoring processes performed in various domains (e.g., finance, healthcare, maintenance). A summary of the characteristics of the logs considered in the evaluation is reported in Table 1. BPI12AC and BPI12WC are two logs recorded in financial domains. They collect event traces recorded monitoring the loan application process of a Dutch financial institute. They contain complete life-cycle traces of sub-processes "Application" (BPI2012AC) and "Work" (BPI2012W), respectively. BPI13O collects event traces recorded in a maintenance domain. It contains event traces of the problem management system of Volvo IT in Belgium. BPI17O is a log recorded in a financial domain. It contains event traces regarding all offers made for an accepted application through the online system of a Dutch financial institute in 2016 and their subsequent events until February 1st 2017, 15:11. BPI20R pertains the administration of an organization. It records event traces pertaining not travel-related requests for payment, collected for two departments in 2017, and the entire university in 2018. Helpdesk was recorded in a custom service domain. It contains event traces from the ticketing management process within the helpdesk operations of an Italian software company. Invoice is a log recorded in a financial domain. It collects event traces of an electronic invoicing process. We considered the pre-processed version of this event log

2. https://data.4tu.nl/portal
3. https://zenodo.org/records/3928487

Table 1: Event log description: **domain** (F– Finance, M – Maintenance, A – Administration, C – Customer Service, H – Healthcare) of the business process, number of traces (♯**Trace**), number of events (♯**Event**), number of activities (♯**Activities**), mean duration of traces measured in days (**Mean duration (days)**), mean length of traces (**Mean length**), average time passed in hours between consecutive events (**Avg next time (h)**), number of global trace characteristics (**# Trace charact.**) and number of local event characteristics (**# Event charact.**)

Event Log	Domain	#Trace	#Events	#Activities	Avg duration (days)	Mean length	Avg next time (h)	#Trace charact.	#Event charact.
BPI12AC	F	13087	60849	10	8.1	5	41.7	1	3
BPI12WC	F	9658	72413	6	11.4	7	36.5	1	3
BPI13O	M	819	2351	5	58.7	3	490.8	3	6
BPI17O	F	42995	193849	8	19.1	5	101.4	5	4
BPI20R	A	6886	36796	19	12	5	53.9	3	4
Helpdesk	C	3804	13710	9	8.8	4	58.6	0	2
Invoice	F	5123	62740	25	2.16	12	4.2	5	3
Logboek	H	1897	6973	10	5	4	32.6	4	3
SP2020	C	23906	178078	13	20.2	7	65.0	3	2

described in [35]. Logboek was collected in the healthcare domain. It records event traces from the Urinary Tract Infection patients handled from a Dutch hospital. SP2020 is recorded in a customer service domain. It was recorded from the service process of a home appliances vendor for repairing faulty devices.

The experimental setup described in [36] was used to conduct the evaluation study of this work. Specifically, a temporal split was performed on each event log to train and test the predictive model. Event traces were sorted based on their starting timestamp, with the first 2/3 traces as training data for training the predictive model. The performance of the resulting model was then evaluated on the remaining 1/3 traces, which represented unseen traces.

5.2 Implementation details

PROPHET[4] was implemented in Python 3.9.18 - 64 bit version using DGL 1.1.2, that is a Python framework for deep learning on graphs using Torch 1.13.0 and Torch Geometric 2.3.1 as the back-end. The Word2Vec algorithm implemented in the Gensim Version 4.3.2 library was used to learn the embeddings. Word2Vec was used with default parameters (embedding dimension equal to 100 and window equal to 5). The selection of the GAT hyper-parameters was performed using the tree-structured Parzen estimator. to explore the following hyper-parameter search spaces: number of Heads in $\{1, 2, 3, 4\}$, feature dropout in $[0.0, 0.5]$, the hidden dimension in $[2^4, 2^7]$, number of GAT layers in $\{1, 2, 3\}$, batch size in $[2^6, 2^9]$ and learning rate in $[10^{-4}, 10^{-2}]$. The aggregate operator adopted was the SUM. The 20% of the training set was used in the optimization phase as the validation set. For each log, an automatic selection process was performed to determine the hyper-parameter configuration that minimized the loss on the validation set within the defined search space. The gradient-based optimization used the Adam's update rule to optimize the cross-entropy categorical loss function. The maximum number of epochs was set equal to 200 and an early-stopping strategy was used to terminate the training phase when there was no validation loss improvement for 20 consecutive epochs.

5.3 Results and analysis

The evaluation was conducted to answer the following three research questions: (Q1) How does the defined method compare to state-of-the-art deep neural methods selected from the recent PPM literature? (Q2) How much the variety and quantity of information represented in a heterogeneous graph can affect the time spent producing PPM decisions? (Q3) How does the heterogeneous graph representation adopted in this study to model prefix traces help us to equip next-activity predictions with valuable explanations of the effect of the structure of running traces on PPM decisions? The results of the analysis of the accuracy performance of the related PPM methods to answer Q1 are reported in Section 5.3.1. The results of the analysis of the efficiency performance of PROPHET to answer Q2 are reported in Section 5.3.2, while the results of the analysis of the explainability to answer Q3 are illustrated in Section 5.3.3, respectively.

5.3.1 Accuracy performance analysis

We evaluated the accuracy performance of both PROPHET and the related deep neural methods [1], [5], [7], [15], [16], [25], [26]. All these PPM methods were run with the information enclosed in all views recorded in the event logs for a safe comparison. This comparative study allows us to explore the accuracy performance of heterogeneous graphs coupled with GAT models in next-activity prediction problems. In detail, the related methods described in [1], [25] and [26] adopt a sequence representation of prefix traces to train a LSTM model in [1], a BiLSTM model in [25] and a Transformer model in [26], respectively. The method described in [7] represents prefix traces through multi-patch color images used to train a ViT model, while the method described in [5] represents prefix traces through color images used to train a CNN with Inception. The method described in [15] represents prefix traces through homogeneous instance graphs and trains a GCN-based model. Finally, the method presented on [16] trains a GRNN model from homogeneous graphs extracted by replaying prefix traces on a Petri Net discovered from the event log. The output of the GRNN model is concatenated with event-related temporal data and used to train an LSTM model adopted to yield next-activity predictions. All related methods, except for [25] and [26], were originally formulated within the multi-view learning schema. In particular, [25] was originally experimented by its authors with activity, resource, and timestamp information, while [26] was originally experimented with activity information. However, to provide a fair comparison, we ran also these two related methods by accounting for all views recorded in the considered event logs. In fact, as the authors

4. https://github.com/vinspdb/PROPHET

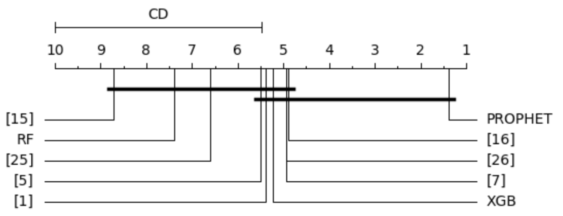


Figure 3: Comparison of the macro FScore of PROPHET and related methods with the Nemenyi test. Groups of methods not significantly different (at $p \leq 0.05$) are connected.

of the considered related methods made the code available, we were able to run all the compared algorithms in the same experimental setting, thus performing a safe comparison. To complete the comparative study we also compared the performance of PROPHET to that of PPM decisions yielded with machine learning methods. To this aim, we considered next-activity prediction models trained with Random Forest (RF) and XGBoost (XGB). Both models were trained on a fixed-length feature vector representation of prefix traces extracted for all views through the Aggregation encoding schema. We selected the Random Forest and XGBoost as machine learning algorithms and the Aggregation as encoding schema according to the results of an extensive comparative analysis including several machine learning algorithms and several encoding schema illustrated in [37]. Parameters of the related methods were set according to the best parameter set-up determined in the code provided by the authors and described in the code repositories.

For each event log, the next-activity predictive function of each method was learned on the prefix traces recorded in the training set, and its ability to forecast the next-activity was evaluated on the prefix traces recorded in the testing set. We measured the accuracy of each next-activity predictive function through the macro FScore. This is a multi-class classification metric commonly measured in imbalanced domains. Specifically, the FScore of each activity i was measured the harmonic mean of precision and recall of i, so that the macro $FScore = \frac{1}{k}\sum_{i=1}^{k} FScore_i$, where k is the number of distinct activities observed in the event log.

Table 2 collects the macro FScore measured for both PROPHET and the related methods. Figure 3 shows the critical difference diagram produced for the macro FScore of the compared methods after rejecting the null hypothesis of equal performance with a *p-value* ≤ 0.05 in Friedman's test and using the post-hoc Nemenyi test for pairwise method comparisons. These results show that PROPHET commonly outperforms the related methods having [16] as runner-up. In particular, there are only two event logs where a related method outperforms PROPHET. Specifically, [16] performs better than PROPHET in BPIC20R training a GRNN model followed by an LSTM model. However, PROPHET is the runner-up of the comparative analysis, while the gap with the first place is low. On the other hand, the machine learning methods RF and XGB outperform PROPHET in BPI13O only. However, PROPHET still outperforms all the deep neural methods of this study also in this event log.

In short, the findings of this accuracy performance analysis assess the effectiveness of our idea of representing prefix traces through heterogeneous graphs, to disclose smart information concerning event characteristics, relationships among characteristics within the same event, and relationships among characteristics across multiple events, which are implicit in the event sequence. This representation can boost the development of a PPM model that gains accuracy in forecasting the next-activity of a running trace, in addition to supporting decision explanations. The runner-up method [16] also leverages graph data to train a GRNN model. However, it considers homogeneous graphs that concern how prefix trace activities replay a Petri Net. Instead, the remaining event information is processed through an additional LSTM model.

5.3.2 *Efficiency performance analysis*

In this section, we analyse the efficiency performance of the training, prediction and explanation stages of PROPHET. Table 3 collects the computation time spent in hours training a GNN model from the training set of each log of this study, and the average and standard deviation of the time spent in seconds predicting the next activity and explaining the decision for every running trace recorded in the testing set of each log. The training stage required several hours to complete the optimization phase and estimation of GNN parameters in all logs. The more time (about 23 hours) was spent completing the training stage in BPI17O that is also the log recording the higher number of traces (42995) spanned across the higher number of views (5 global views and 4 local views). In any case, the observed long time spent completing the GNN training stage is still acceptable as long as the monitored process is in a steady state. This condition makes it plausible that the PPM learning process is performed once, while new process traces are still stored long after the PPM model has been learned. Adapting the training of a GNN model to an evolving stream environment, where the decision model may be trained repeatedly, requires the investigation of mechanisms to speed-up the learning stage. A recent PPM study [4] has shown that fine-tuning coupled with concept drift handling may be a good strategy to adapt a LSTM deep neural model to data drifts occurring in a stream environment by accounting for computation constraints. The investigation of a data stream extension of the proposed approach is a future research direction for this study. Finally, in this study, the time spent, in average, predicting and explaining a next-activity prediction is always less than 12 seconds in average. Notably, the average time spent predicting a next-activity is always less than 0.05 seconds. This performance gives time to the process management stakeholders to handle the information on the next-activity forecast and its explanation, before the new activity is actually executed in the trace. This consideration applies in processes where events are executed on average every 20 seconds. On the other hand, if we leave explanations aside, then the next-activity predictions are available well in advance of events recorded every second.

To complete this analysis, Figure 4 shows the radar chart to compare the average time spent in seconds predicting and explaining the next-activity of the running traces recorded in the testing set of all logs, the number of views used to de-

Table 2: Macro Fscore of PROPHET and related deep learning methods defined in [1], [7], [15], [16], [25], [26], [5] and the machine learning methods RF and XGB defined with the Aggregation encoding according to [37]. The best results are in bold, while the runner-up results are underlined.

Method	Model type	BPI12AC	BPI12WC	BPI13O	BPI17O	BPIC20R	Helpdesk	Invoice	Logboek	SP2020
PROPHET	GNN with GAT	**69.0**	**70.5**	40.6	**72.1**	50.5	**26.3**	**93.1**	**79.4**	**60.3**
[1]	LSTM	64.1	68.5	39.0	71.4	45.0	25.5	91.9	79.3	59.4
[7]	ViT	66.1	**70.5**	35.8	72.0	49.1	25.3	92.6	79.2	58.3
[15]	GCN	66.1	59.8	20.3	50.5	39.5	25.5	78.5	32.6	55.1
[16]	GRNN+LSTM	67.2	68.7	39.8	50.5	**51.2**	25.8	92.2	73.0	59.6
[25]	BiLSTM	66.0	69.4	31.5	69.1	45.5	25.0	92.7	75.9	58.9
[26]	Transformer	68.5	66.8	36.2	71.2	48.1	24.4	92.8	78.9	59.7
[5]	CNN with Inception	68.1	66.1	37.5	70.5	48.3	25.3	91.9	77.5	59.7
[37]	RF	67.9	67.3	43.4	71.2	31.0	23.9	88.4	68.5	58.6
[37]	XGB	67.2	67.3	**44.6**	71.3	38.4	24.4	92.7	79.1	59.4

Table 3: Computation time of PROPHET: **Learn(h)**–total time spent in hours (h) training the GNN model from the training set of each log, **Predict(s)**, **Explain(s)** and **Predict+Explain(s)**–average and standard deviation of the time spent in seconds (s) predicting the next-activity of each running trace recorded in the testing set of a log, explaining the decision, as well as predicting and explaining the decision

Event log	Learn(h)	Predict(s)	Explain(s)	Predict+Explain(s)
BPI12AC	3.97	0.047±0.000	8.01±0.206	8.06±0.206
BPI12WC	9.75	0.030±0.000	6.53±0.228	6.57±0.228
BPI13O	0.24	0.045±0.002	11.20 ± 0.260	11.25 ± 0.26
BPI17O	22.62	0.014 ±0.000	5.68 ±0.251	5.69 ±0.251
BPI20R	4.67	0.022±0.000	5.84±0.234	5.87±0.234
Helpdesk	0.52	0.005±0.000	1.20±0.080	1.21±0.080
Invoice	5.31	0.030±0.001	6.55±0.231	6.58±0.231
Logboek	1.24	0.020±0.001	4.07±0.16	4.09±0.16
SP2020	12.17	0.016±0.000	3.52±0.117	3.54±0.117

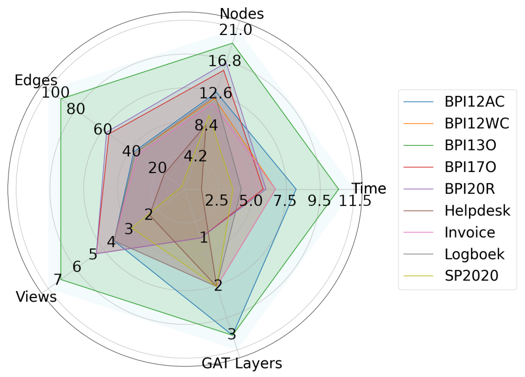
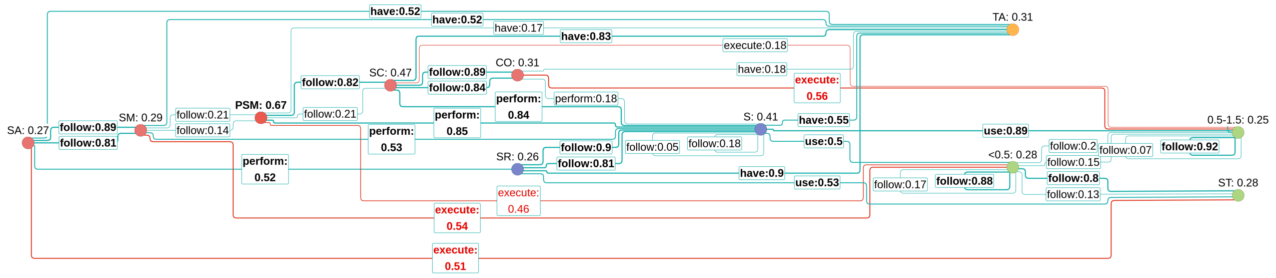


Figure 4: Radar chart of the average time spent in seconds predicting and explaining the next-activity of the testing running traces, the number of views recorded in the event logs to describe the running traces, the average number of nodes, the average number of edges used to encode the considered running traces as heterogeneous graphs and the number of GAT layers in the GNN model. The running traces with length equal to three are displayed in the chart.

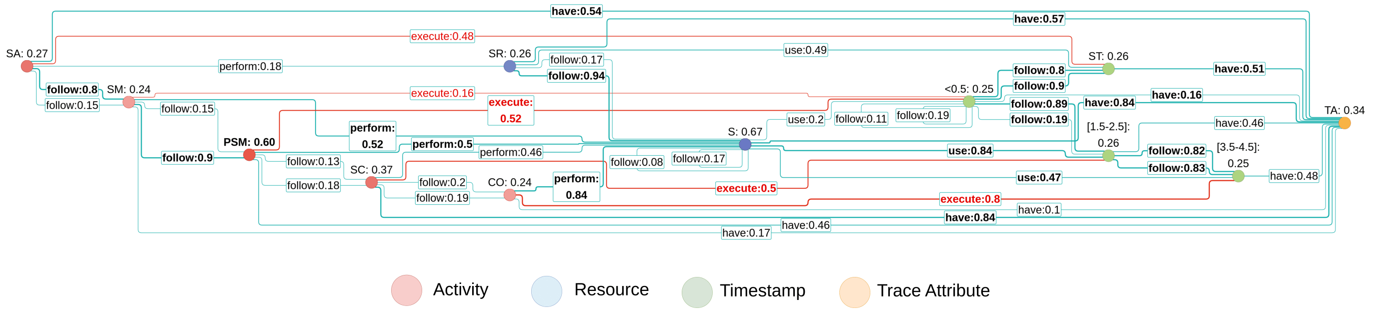
scribe the running traces, the average number of nodes and the average number of edges used to encode the considered running traces as heterogeneous graphs, and the number of GAT layers in the GNN model as it was automatically selected with the optimizer. This multivariate analysis is displayed for running traces with length equal to 3. The higher the number of GAT layers, the higher the computation time spent predicting and explaining the next-activity. On the other hand, the higher the number of views and, consequently, the higher the number of nodes and edges, the higher the time to predict and explain the GNN decision. The same analysis can be repeated by varying the length of the running traces and drawing the same conclusions.

5.3.3 Explainability performance analysis

In this section, we illustrate some examples of the explanation information, that can be disclosed post-hoc in PROPHET using the developed extension of the GNN Explainer algorithm. First, we examine the explanation information produced for the local decisions concerning two sample prefix traces recorded in the testing set of the event log Invoice. Figures 5a and 5b show the heterogeneous graphs of these two prefix traces enriched with explanations. In this example, PROPHET correctly predicts ”ME” – ”Manual enter the order number-Entered” and ”CS” – ”Compare of Sums missing” as the next activity of the two running traces, respectively. Notably, both prefix traces share the same ”SM - PSM - SC - CO” activity prefix, where: ”SM” – ”Start Missing”, ”PSM” – ”Process Start Missing”, ”SC” – ”Status Change to being approved missing” and ”CO” – ”Check Order number missing”. However, the two prefix traces represent running executions of the Invoice business process, which will proceed with executing two different activities in the future. In this case, the information enclosed only in the control flow of activities is not enough to correctly predict the two different next activities for the two prefix traces. Instead, PROPHET is able to yield the correct next-activity decision for both prefix traces by leveraging the multi-view, global and local, information recorded in the event log. In addition, decision explanations produced by PROPHET allow us to understand how the multi-view trace characteristics can help in correctly disentangling the next-activity ”ME” from the next-activity ”CS” in the considered prefix traces. For example, the explanation-enriched graph representation of the two prefix traces shows that, although the activities executed in the two prefix traces are all performed by the resource ”S” (Server), some differences occur in their time cycle. In addition, the explanation values computed with GNN Explainer highlight which information of each time cycle is actually relevant for predicting the correct next activity in the two cases. In particular, the two explanation-enriched graphs show that in both prefix traces, activities ”SM” and ”PSM” have been executed before 0.5



(a) Next-activity: "ME" – "Manual enter the order number-Entered"



(b) Next-activity: "CS" – "Compare of Sums missing"

Figure 5: Local explanations of the next-activity predicted by PROPHET for two prefix traces of Invoice. The real values reported on nodes and links quantify the effect of the corresponding nodes or links on the decision. The highest the value associated with a node or link, the most important the effect of the node or link on the decision. Explanation values greater than 0.5 are in bold. The links "execute" between activities and timestamps are in red.

seconds passed from the beginning of the trace. On the other hand, in Figure 5a, activities "SC" and "CO" have been executed in the time interval that goes from 0.5 to 1.5 seconds passed from the beginning of the trace with the timestamp of "CO" that has high relevance (greater than 0.5) on the correct decision "ME" as the next-activity of the prefix trace. In Figure 5b, activities "SC" and "CO" have been executed in the time intervals that go from 1.5 to 2.5 seconds and 2.5 to 4.5 seconds passed, respectively, from the beginning of the trace. Notably, the timestamps of "PSM", "SC" and "CO" have all high relevance (greater than 0.5) on the correct decision "CS" as the next activity of the prefix trace. So, this explanation suggests that the longer the time passed from the beginning of the running trace before the execution of activities "SC" and "CO", the more important the symptoms indicating "CS" as the upcoming activity of the running trace. This information can be used at the process management level to plan the work requested to better manage the execution, as well as the consequences of the execution of activity "CS" on the current process execution. For example, whenever this activity was considered as a failure symptom for the process execution, this information can be used to proactively recognize the upcoming activity and activate actions to mitigate (or possibly avoid) the failure. Exploring the potential of this kind of explanation insights for mitigation or prescriptive actions is a future direction of this research.

Finally, we examine the global effect of information recorded in the different views of an event log on decisions provided by the next-activity predictive model of PROPHET. This analysis is based on the explanation values computed through GNN Explainer for each type of node

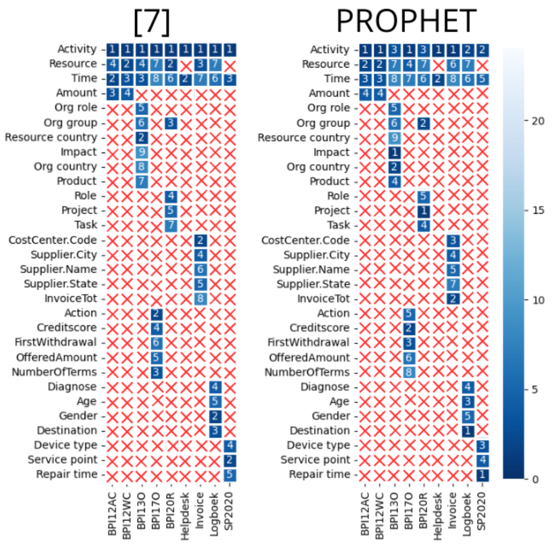


Figure 6: Heatmaps of the global effect of views (axis Y) recorded in the study event logs (axis X) on the next-activity predictive models trained by both [7] and PROPHET. "×" denotes that the view reported on the axis Y is missing in the event log reported on the axis X.

in the graph (i.e., view recorded in the event log). For each view, we averaged the explanation values of the node type associated with the view and measured the prefix traces recorded in the testing set of the event log. The higher the average explanation value of the view, the more important the view is for the model decisions. Notably this global explanation analysis is similarly performed in [7] using the explanation information intrinsically disclosed through the attention mechanisms of the ViT model. So, Figure 6 shows the heatmaps of the view ranks computed for the next-activity predictive models trained with both [7] and PROPHET. As expected both maps show that the activity information is globally the most important information for decisions of both methods in almost all the event logs. The only exceptions are observed for the event logs BPI13O, BPI20R, Logboek and SP2020 for which the information recorded in the views "Impact", "Project", "Destination" and "Repair Time" become globally the most important for decisions yielded by the predictive models learned by PROPHET. However, in all four cases, the information recorded in the view "Activity" is globally in the top-three ranked views. In general, the two heatmaps show that, despite in each event log there is commonly high overlap between views that are globally top-ranked for the the two methods, some differences occur. For example, in the event log BPI13O, the view "Resource Country" that is globally in the top-three ranked views for decisions yielded by [7], is in the bottom-three ranked views for decisions yielded by PROPHET. This difference may be imputed to the fact that PROPHET, thanks to the adopted heterogeneous graph representation of prefix traces, can account for relationships between different characteristics that are encoded in the inter-view links of the heterogeneous graphs. Our idea is that the opportunity of accounting for interactions among views may contribute to changing the role of each single view on decisions. In fact, the global explanation heatmaps show that the information recorded in an event log may change its importance within the PPM process depending on the data representation used to represent the running traces and the learning approach adopted to train the decision model for predicting the next-activity. In addition, the gloabl explanation heatmaps show that, independently of the decision model considered, the type of information most valuable for predicting the next-activity of a prefix trace may depend on the type of study process of which the trace is an execution. So, we believe that one of the achievements of the XAI analysis illustrated here is that the global explanation findings contribute to lend support to the decision to develop a multi-view PPM approach that does not solely rely on the standard views (activity, timestamp, and resource). Another result is the assessment that the relevance of different event log views for next-activity decisions depends not only on the peculiarities of the business process considered, but also on the data representation and learning algorithm adopted.

6 Conclusion

In this work, we present a novel PPM method for next-activity prediction. We introduce a representation to encode the multi-view information of a prefix trace with multiple types of nodes of a heterogeneous graph. We use a GAT architecture to leverage multi-view information and relationships, and an extended version of the GNN Explainer algorithm to incorporate explainability into GAT's decisions. The evaluation study analyses accuracy and efficiency of the proposed approach and explores the effectiveness of the produced explanations. As future work, we plan to explore how GNN-based explanations can be used from the process management stakeholders to plan actions to better support the efficient execution of next-activities or proactively mitigate their effects whenever they are unwanted activities. In addition, we intend to start the exploration of how framing the proposed graph-based approach in a streaming setting.

Acknowledgments

Vincenzo Pasquadibisceglie and Giovanna Castellano are partially supported by the project FAIR - Future AI Research (PE00000013), Spoke 6 - Symbiotic AI, under the NRRP MUR program funded by the NextGenerationEU. Raffaele Scaringi is funded by a Ph.D. fellowship within the framework of the Italian "D.M. n. 352, April 9, 2022" - under the NRRP, co-supported by "Exprivia S.p.A." (CUP H91I22000410007). The work of Donato Malerba is in partial fulfillment of the project MAD - The Additive Metamorphosis of Design (ARS01_00717) funded by MUR.

References

[1] V. Pasquadibisceglie, A. Appice, G. Castellano, and D. Malerba, "A multi-view deep learning approach for predictive business process monitoring," IEEE Trans. Serv. Comput., vol. 15, no. 4, pp. 2382–2395, 2022.

[2] B. R. Gunnarsson, S. vanden Broucke, and J. De Weerdt, "A direct data aware lstm neural network architecture for complete remaining trace and runtime prediction," IEEE Trans. Serv. Comput., vol. 16, no. 4, 2023.

[3] L. Aversano, M. L. Bernardi, M. Cimitile, M. Iammarino, and C. Verdone, "A data-aware explainable deep learning approach for next activity prediction," Eng. App. of Artif. Intell., vol. 126, p. 106758, 2023.

[4] V. Pasquadibisceglie, A. Appice, G. Castellano, and D. Malerba, "DARWIN : An online deep learning approach to handle concept drifts in predictive process monitoring," Eng. Appl. Artif. Intell., vol. 123, no. Part C, p. 106461, 2023.

[5] ——, "Predictive process mining meets computer vision," in BPM Forum 2020, ser. LNBIP, vol. 392. Springer, 2020, pp. 176–192.

[6] H. Weytjens and J. De Weerdt, "Process outcome prediction: Cnn vs. lstm (with attention)," in BPM 2020 International Workshops. Springer, 2020, pp. 321–333.

[7] V. Pasquadibisceglie, A. Appice, G. Castellano, and D. Malerba, "JARVIS: Joining adversarial training with vision transformers in next-activity prediction," IEEE Trans. Serv. Comput., pp. 1–14, 2023.

[8] R. Galanti, M. de Leoni, M. Monaro, N. Navarin, A. Marazzi, B. Di Stasi, and S. Maldera, "An explainable decision support system for predictive process analytics," Eng. App. of Artif. Intell., vol. 120, p. 105904, 2023.

[9] M. Li, A. Micheli, Y. G. Wang, S. Pan, P. Lió, G. S. Gnecco, and M. Sanguineti, "Guest editorial: Deep neural networks for graphs: Theory, models, algorithms, and applications," IEEE Trans. Neural Networks Learn. Syst., vol. 35, no. 4, pp. 4367–4372, 2024.

[10] L. Bai, L. Cui, Y. Wang, M. Li, J. Li, P. S. Yu, and E. R. Hancock, "Haqjsk: Hierarchical-aligned quantum jensen-shannon kernels for graph classification," IEEE Trans. Knowl. Data Eng., pp. 1–14, 2024.

[11] K. Huang, Y. G. Wang, and M. L. P. Liò, "How universal polynomial bases enhance spectral graph neural networks:heterophily, over-smoothing, and over-squashing," in ICML 2024,. PMLR, 2024, pp. 1–20.

[12] C. Agarwal, O. Queen, H. Lakkaraju, and M. Zitnik, "Evaluating explainability for graph neural networks," Scientific Data volume, vol. 10, 144, pp. 1–18, 2023.

[13] S. Weinzierl, "Exploring gated graph sequence neural networks for predicting next process activities," in BPM 2022 International Workshops. Springer, 2022, pp. 30–42.

[14] I. Venugopal, J. Töllich, M. Fairbank, and A. Scherp, "A comparison of deep-learning methods for analysing and predicting business processes," in IJCNN 2021. IEEE, 2021, pp. 1–8.

[15] A. Chiorrini, C. Diamantini, L. Genga, and D. Potena, "Multi-perspective enriched instance graphs for next activity prediction through graph neural network," J. Intell. Inf. Syst., pp. 1–21, 2023.

[16] E. Rama-Maneiro, J. C. Vidal, and M. Lama, "Embedding graph convolutional networks in recurrent neural networks for predictive monitoring," IEEE Trans. Knowl. Data Eng., vol. 36, no. 1, pp. 137–151, 2024.

[17] Z. Shao, Y. Xu, W. Wei, F. Wang, Z. Zhang, and F. Zhu, "Heterogeneous graph neural network with multi-view representation learning," IEEE Trans. Knowl. Data Eng., vol. 35, no. 11, pp. 11 476–11 488, 2023.

[18] Z. Chen, L. Fu, J. Yao, W. Guo, C. Plant, and S. Wang, "Learnable graph convolutional network and feature fusion for multi-view learning," Information Fusion, vol. 95, pp. 109–119, 2023.

[19] T. Li, J. Deng, Y. Shen, L. Qiu, Y. Huang, and C. C. Cao, "Towards fine-grained explainability for heterogeneous graph neural network," in AAAI 2023. AAAI Press, 2023, pp. 8640–8647.

[20] G. P. Mika, A. Bouzeghoub, K. Wegrzyn-Wolska, and Y. M. Neggaz, "Hgexplainer: Explainable heterogeneous graph neural network," in WI-IAT 2023. IEEE, 2023, pp. 221–229.

[21] X. Wang, D. Bo, C. Shi, S. Fan, Y. Ye, and P. S. Yu, "A survey on heterogeneous graph embedding: Methods, techniques, applications and sources," IEEE Trans. Knowl. Data Eng., vol. 9, no. 2, pp. 415–436, 2023.

[22] S. Brody, U. Alon, and E. Yahav, "How attentive are graph attention networks?" in ICLR 2022.

[23] Z. Ying, D. Bourgeois, J. You, M. Zitnik, and J. Leskovec, "GNNExplainer: Generating explanations for graph neural networks," in NeurIPS 2019, 2019, pp. 9240–9251.

[24] M. Camargo, M. Dumas, and O. G. Rojas, "Learning accurate LSTM models of business processes," in BPM 2019, ser. LNCS, vol. 11675. Springer, 2019, pp. 286–302.

[25] B. Wickramanayake, Z. He, C. Ouyang, C. Moreira, Y. Xu, and R. Sindhgatta, "Building interpretable models for business process prediction using shared and specialised attention mechanisms," Knowledge-Based Systems, vol. 248, pp. 1–22, 2022.

[26] Z. A. Bukhsh, A. Saeed, and R. M. Dijkman, "Processtransformer: Predictive business process monitoring with transformer network," CoRR, vol. abs/2104.00721, 2021.

[27] V. Pasquadibisceglie, A. Appice, G. Ieva, and D. Malerba, "TSUNAMI - an explainable PPM approach for customer churn prediction in evolving retail data environments," J. Intell. Inf. Syst., 2023.

[28] A. Stevens and J. De Smedt, "Explainability in process outcome prediction: Guidelines to obtain interpretable and faithful models," European Journal of Operational Research, 2023.

[29] F. Folino, G. Folino, M. Guarascio, and L. Pontieri, "Data- & compute-efficient deviance mining via active learning and fast ensembles," J. Intell. Inf. Syst., 2024.

[30] M. Harl, S. Weinzierl, M. Stierle, and M. Matzner, "Explainable predictive business process monitoring using gated graph neural networks," Journal of Decision Systems, pp. 1–16, 2020.

[31] V. Pasquadibisceglie, G. Castellano, A. Appice, and D. Malerba, "FOX: a neuro-fuzzy model for process outcome prediction and explanation," in ICPM 2021. IEEE, 2021, pp. 112–119.

[32] T. Mikolov, K. Chen, G. Corrado, and J. Dean, "Efficient estimation of word representations in vector space," in ICLR 2013, 2013.

[33] G. M. Tavares, R. S. Oyamada, S. Barbon, and P. Ceravolo, "Trace encoding in process mining: A survey and benchmarking," Eng. Appl. of Artif. Intell., vol. 126, p. 107028, 2023.

[34] Z. Wu, S. Pan, F. Chen, G. Long, C. Zhang, and S. Y. Philip, "A comprehensive survey on graph neural networks," IEEE Trans. Neural Netw. Learn. Syst., vol. 32, no. 1, pp. 4–24, 2020.

[35] I. Verenich, M. Dumas, M. L. Rosa, F. M. Maggi, and I. Teinemaa, "Survey and cross-benchmark comparison of remaining time prediction methods in business process monitoring," ACM Trans. Intell. Syst. Technol., vol. 10, no. 4, pp. 1–34, 2019.

[36] N. Tax, I. Verenich, M. La Rosa, and M. Dumas, "Predictive business process monitoring with LSTM neural networks," in CAISE 2017, ser. LNCS. Springer, 2017, pp. 477–492.

[37] I. Teinemaa, M. Dumas, M. L. Rosa, and F. M. Maggi, "Outcome-oriented predictive process monitoring: Review and benchmark," ACM TKDD, vol. 13, no. 2, pp. 1–57, 2019.

Vincenzo Pasquadibisceglie received a Ph.D. in Computer Science from the University of Bari He is a Researcher in the Department of Computer Science of the University of Bari Aldo Moro. His research activity concerns process mining, deep learning and data-centric AI. He was involved in a regional research project on process mining. He received 2019 IET's Vision and Imaging Award ICDP 2019. He is member of the IEEE Task Force on Process Mining.

Raffaele Scaringi received his master's degree in Data Science from the University of Bari Aldo Moro, Italy. He is currently a PhD student at the Department of Computer Science, University of Bari Aldo Moro, Italy. His research interests include deep representation learning, knowledge graphs and computer vision.

Annalisa Appice is a Full Professor at the Department of Computer Science, University of Bari Aldo Moro, Italy. Her research interests include AI, cybersecurity and process mining. She has published more than 190 papers in journals and conferences. She was the Program co-Chair of several international conferences. She is a member of the editorial board of several international journals. She is member of the IEEE Task Force on Process Mining.

Giovanna Castellano is an Associate Professor at the Department of Computer Science, University of Bari Aldo Moro, Italy, where she coordinates the Computational Intelligence Lab. She is member of the IEEE Society, the EUSFLAT society and the INDAM-GNCS society. Her research interests are in the area of Computational Intelligence and Computer Vision. She has published more than 300 papers in international journals and conferences. She is Associate Editor of several international journals. She was General chair of IEEE-EAIS2020. She is a member of the IEEE Task Force on Explainable Fuzzy Systems.

Donato Malerba is a Full Professor in the Department of Computer Science at the University of Bari Aldo Moro in Italy. He has been responsible for the local research unit in several EU and national projects. He is the scientific coordinator of Spoke 6 - Symbiotic AI within the project FAIR funded by NextGenerationEU. He served as the Program co-Chair for several conferences. He is a member of the editorial boards of several international journals. He published more than 350 papers in international journals and conferences. His research interests revolve around AI and big data analytics. He participates to the IEEE Task Force on Process Mining.