Käppel, Martin; Ackermann, Lars; Jablonski, Stefan; Härtl, Simon

Explaining transformer-based next activity prediction by using attention scores

Process Science

Provided in Cooperation with:
Springer Nature

Suggested Citation: Käppel, Martin; Ackermann, Lars; Jablonski, Stefan; Härtl, Simon (2025) : Explaining transformer-based next activity prediction by using attention scores, Process Science, ISSN 2948-2178, Springer International Publishing, Cham, Vol. 2, Iss. 1, https://doi.org/10.1007/s44311-025-00018-4

Käppel *et al. Process Science (2025) 2:11*
https://doi.org/10.1007/s44311-025-00018-4

Process Science

RESEARCH Open Access

Explaining transformer-based next activity prediction by using attention scores

Martin Käppel[1*], Lars Ackermann[2], Stefan Jablonski[3] and Simon Härtl[3]

*Correspondence:
martin.kaeppel@fau.de

[1] Chair of Digital Industrial Service Systems, Friedrich-Alexander-University Erlangen-Nuremberg, Fürther Str. 248, 90429 Nuremberg, Germany
[2] Department for Computer Science, Hof University of Applied Sciences, Alfred-Goppel-Platz 1, 95028 Hof, Germany
[3] Institute for Computer Science, University of Bayreuth, Universitätsstraße 30, 95447 Bayreuth, Germany

Abstract

Predictive business process monitoring aims to enhance process execution by providing real-time predictions about the future evolution of a process instance. In recent years, several deep learning approaches have been established as state of the art for various predictive tasks, including those based on the transformer architecture. The transformer architecture is equipped with a powerful attention mechanism that assigns attention-based importance scores to each input element, guiding the model to focus on the most relevant parts of the sequence, regardless of their position. This capability leads to more accurate and contextually grounded predictions. However, like most deep learning models, transformers largely operate as a black box, making it challenging to trace how specific features influence the model's predictions. In this paper, we conduct a series of experiments to examine the role of attention scores in a transformer-based next activity prediction model. Specifically, we investigate whether these scores provide meaningful explanations for the model's decisions. Our findings reveal that attention scores can indeed serve as effective explanations. Building on these insights, we propose two novel, global, graph-based explanation approaches that illustrate the model's understanding of the process's control flow. Our evaluation using various metrics on both real-world and synthetic event logs demonstrates that these explainers effectively capture the model's decision-making process. By improving interpretability, these insights not only enhance process participants' confidence in predictive models but also offer a valuable foundation for refining model performance. Furthermore, our investigation into the reliability of attention scores offers valuable insights into how transformer models encapsulate temporal and sequential dependencies in prediction tasks.

Keywords: Predictive process monitoring, Transformer, Attention mechanism, Explainability

Introduction

In recent years, *predictive business process monitoring* (Maggi et al. 2014; Grigori et al. 2004) has experienced remarkable growth, driven by advances in artificial intelligence (Weinzierl et al. 2024), and has established itself as a subfield in process mining (Di Francescomarino and Ghidini 2022). In contrast to post-mortem analyses of event data, which focus on past and current events, predictive business process monitoring provides runtime support for the execution of a business process by making various predictions

about the future evolution of the process instance (Di Francescomarino and Ghidini 2022). This includes, among others, behavior-related predictions (e.g., next activity (Pasquadibisceglie et al. 2019; Camargo et al. 2019; Evermann et al. 2017)), outcome-related predictions (e.g., outcome (Kratsch et al. 2021)), and time-related predictions (e.g., remaining time (Verenich et al. 2019)).

From a business perspective, early knowledge about the future of a process instance offers significant advantages, as it permits improved resource and time planning, better preparation of upcoming steps, and early identification of potential problems (Márquez-Chamorro et al. 2017; Maggi et al. 2014). The latter, for example, allows process participants to take timely corrective actions to mitigate risks (Di Francescomarino and Ghidini 2022).

Typically, predictive business process monitoring approaches involve constructing predictive models based on historical event log data captured by information systems (Grigori et al. 2004). These models are then applied to ongoing process instances to generate valuable predictive insights (Maggi et al. 2014). In recent years, deep learning approaches have been established as state of the art for predictive tasks for mainly two reasons: First, they have proven to outperform traditional machine learning models like decision trees or support vector machines for different targets (e.g., next activity (Mehdiyev et al. 2020), outcome (Kratsch et al. 2021), remaining time (Verenich et al. 2019)) in terms of accuracy and earliness of prediction. Second, they are completely data-driven so that they no longer require an explicit representation of the underlying process model (Senderovich et al. 2019). As a consequence, various kinds of deep learning architectures have been employed, among others Convolutional Neural Networks (Pasquadibisceglie et al. 2019), Long Short Term Memory Neural Networks (LSTM) (Evermann et al. 2017; Camargo et al. 2019), or more recently transformer architectures (Bukhsh et al. 2021).

Despite their strong predictive performance, deep learning models largely operate as black boxes, limiting insights into their reasoning and decision-making processes (Nauta et al. 2023). This lack of transparency is a significant factor in why people distrust these models and, hence, marks a major obstacle to their usage in practice (Carvalho et al. 2019). The field of *explainable artificial intelligence (XAI)* aims to address this issue by developing techniques (so-called *explainers*) that explain the decisions of machine learning models (Carvalho et al. 2019; Nauta et al. 2023). In predictive business process monitoring both task-agnostic explanation techniques and techniques tailored towards the specific needs of predictive business process monitoring are applied (Weinzierl et al. 2020). In general, explainers are categorized as either *local* or *global* based on their explanation scope. While local explainers aim to explain the prediction for a single process instance, global explainers seek to find an explanation to uncover how a model makes decisions across a collection of instances (e.g., an event log). Hence, global approaches provide insights into how a model makes decisions in general, whereas local approaches focus on explaining single predictions. Previous research in predictive business process monitoring has predominantly concentrated on developing local explainers, so the development of process-specific global explainers is still in its infancy.

Hence, the hypothesis proposed in the landmark work of Evermann et al. (2017) – that deep learning models for next activity prediction inherently learn the underlying process structure – remains largely untested. *Peeperkorn* et al. investigated this hypothesis

specifically for LSTMs, finding that these models sometimes *"struggle to learn process model structure"* (Peeperkorn et al. 2023). Nevertheless, examining this hypothesis for other deep learning architectures remains a valuable research direction (Peeperkorn et al. 2023), especially given that LSTMs often face challenges with long sequences, as they tend to pay less attention to elements appearing earlier in the sequence. The transformer architecture (Vaswani et al. 2017) addresses this issue with a powerful attention mechanism, which assigns importance scores (so-called attention scores) to each element of the input sequence, guiding the model to focus on the most relevant parts of the sequence, regardless of their position (Bukhsh et al. 2021; Vaswani et al. 2017). This architecture has led to new state-of-the-art approaches across multiple research fields (Vaswani et al. 2017) (e.g., large language models such as GPT-4 (OpenAI et al. 2024) and BERT (Devlin et al. 2019)).

As this architecture also demonstrates strong performance in predictive business process monitoring by effectively predicting the next activity (Bukhsh et al. 2021), this paper explores the following research question: *Can the attention scores within the transformer architecture provide insights into whether the trained prediction model has developed an understanding of a process's control flow?*

This article is an extended and revised version of a previously published conference paper (Käppel et al. 2024). In Käppel et al. (2024), we conducted initial experiments to examine whether the aforementioned attention scores could serve as a solid basis for developing XAI approaches. We also proposed two novel, global, transformer-specific explanation approaches and conducted a quantitative evaluation using real-world event logs. This extended version expands on our previous work by presenting additional experiments to gain deeper insights into the reliability of attention scores. Furthermore, we extend our evaluation to include synthetically generated event logs from process models, enabling a direct comparison between extracted explanation rules and corresponding process models. We also present an embodiment of this approach as a software tool and conduct a qualitative analysis of the extracted explanations.

The rest of the paper is organized as follows: Background section introduces basic terminology and the fundamentals of the transformer architecture. After positioning our work with respect to related research (Related work section), we conduct a series of experiments to investigate whether attention scores can serve as an explanation (Pre-study: the relevance of attention scores section). Building on these findings, we present the proposed global explainers (Explanation approaches section). In Evaluation section, we perform a qualitative and quantitative evaluation of our approach on both real-world event logs and synthetically generated event logs. Concluding remarks section discusses potential limitations and implications for theory and practice, while Conclusion and future work section provides directions for future research.

Background

In this section, we first introduce key concepts and notations from the field of process mining that are essential for understanding the remainder of the paper. Next, we explain the functionality of the transformer architecture, with a strong focus on its attention mechanism, which plays a pivotal role in the proposed explanation approaches.

Event logs and next activity prediction

A *business process* is a sequence of activities and decisions carried out to deliver a valuable outcome to the customer (Dumas et al. 2018). Each execution of such a business process is called a *process instance* or a *case* (van der Aalst 2016)). Modern IT systems record and store information about process executions in the form of event logs – sets of timestamped events that include various event attributes encapsulating information about the execution of activities (van der Aalst 2016)). In the following, we denote the set of activities of a business process as $\mathcal{A}$ and define an event formally as follows:

Definition 1 An event is as a tuple $e = (a, c, t, d_1, ..., d_m)$, where $a \in \mathcal{A}$ is the executed activity, c is a case identifier indicating the process instance to which the event belongs, t is the timestamp of execution, and $d_1, ..., d_m$ represent the data payload, i.e., optional event attributes related with the execution of activity a.

Thus, at a minimum, an event contains the following event attributes: a case identifier, the executed activity, and the timestamp of execution (van der Aalst 2016)). Accordingly, we use functions $\pi_a(e)$, $\pi_c(e)$, $\pi_t(e)$, and $\pi_{d_1}(e), \ldots, \pi_{d_m}(e)$ to access the activity, case identifier, timestamp, and the data payload of an event e (van der Aalst 2016)).

All events belonging to the same process instance can be temporally ordered by their timestamp into a so-called *trace* (van der Aalst 2016)):

Definition 2 A trace is a non-empty, finite sequence of events $\sigma = \langle e_1, \ldots, e_n \rangle$ such that, for $1 \leq i < j \leq n$, the following conditions hold:

- all events are ordered according to their timestamp (i.e., $\pi_t(e_j) \geq \pi_t(e_i)$) and
- all events belong to the same process instance (i.e., $\pi_c(e_j) = \pi_c(e_i)$).

The length of a trace, denoted by $|\sigma|$, refers to the number of events within that trace.

Based on this definition, we can define an event log as follows (van der Aalst 2016)):

Definition 3 An event log L is a set of traces of the same business process. The size of the event log, denoted by $|L|$, is defined as the number of traces contained in L.

To represent process instances at different points in time, we utilize the prefixes of a trace.

Definition 4 Let $\sigma = \langle e_1, \ldots, e_n \rangle$ be a trace and $r \in \{1, \ldots, n-1\}$. The prefix of a trace σ of length r is defined as a function hd that returns the first r events of σ, i.e., $hd(\sigma, r) = \langle e_1, \ldots, e_r \rangle$.

Each prefix thus consists of a sequence of consecutive events from the start of the trace up to a specific point, capturing the process execution up to that moment.

A next activity prediction model receives a record of a running process instance (i.e., a prefix) as input and predicts the most likely subsequent activity. We therefore define next activity prediction formally as a function (Rama-Maneiro et al. 2021):

Definition 5 Let $\sigma = \langle e_1, \ldots, e_r \rangle$ be a prefix of length r. Next activity prediction is defined as a function Ω that predicts for the activity of the next event e_{r+1}, which is not yet known.

Thus, a next activity prediction approach aims to train a prediction model that approximates this function Ω using a given event log. Most predictive business process monitoring approaches employ machine learning algorithms to learn the function Ω. In this paper, we do not define a new architecture but instead use the transformer architecture proposed by Bukhsh et al. (2021). Since this architecture considers solely the activity event attribute and ignores the remaining event attributes of a prefix, we can – without loss of generality – represent traces and prefixes simply as sequences of activities. Thus, we denote a trace or a prefix as follows: $\sigma = \langle \pi_a(e_1), \ldots, \pi_a(e_n) \rangle$.

Transformer architecture and attention mechanism

The transformer architecture was originally introduced in Vaswani et al. (2017) as a sequence-to-sequence model built on an encoder-decoder architecture equipped with a robust attention mechanism. *Sequence-to-sequence* modeling refers to tasks that convert an input sequence into an output sequence of potentially different lengths, such as language translation (Zhao et al. 2023). Since its introduction, the transformer architecture has also been adapted for sequence-to-vector tasks, where an input sequence is processed to output a fixed-size vector. Such adaptations are particularly relevant for classification or regression tasks, including next activity prediction or remaining time prediction (Bukhsh et al. 2021). This adaptation is achieved by omitting the decoder part of the architecture, which is why this variant is also called an *encoder-only transformer* (Devlin et al. 2019).

In this work, we focus on describing this modified transformer architecture as applied to next activity prediction. Specifically, we employ the process transformer architecture outlined in Bukhsh et al. (2021) (see Fig. 1). The following sections describe the key components of the architecture relevant to this study: the input, the attention mechanism, and the output of the transformer. For details on other aspects of the architecture, we refer the reader to Vaswani et al. (2017) and Bukhsh et al. (2021).

Input The transformer receives a sequence of activities $\sigma = (a_1, \ldots, a_n)$ as input. This input is then embedded into a high-dimensional space of dimension $d_m \in \mathbb{N}_{>0}$[1] by converting each element in the sequence into a so-called *embedding vector* (Vaswani et al. 2017). As a result, for each element a_i of the sequence, two embedding vectors of dimension d_m are generated: a vector representing the element itself (so-called *input embedding*) $x^i_{in} \in \mathbb{R}^{1 \times d_m}$ and a vector representing the position (so-called *position embedding*) $x^i_{pos} \in \mathbb{R}^{1 \times d_m}$. Following common practice in transformer implementations, these embedding vectors are represented as row vectors.

Because the transformer processes the entire input sequence in parallel and does not inherently capture positional information, position embeddings are essential to avoid losing the order of the elements. Both vectors are then summed component-wise,

[1] In the architecture used in this paper d_m is set to 36 (Bukhsh et al. 2021)

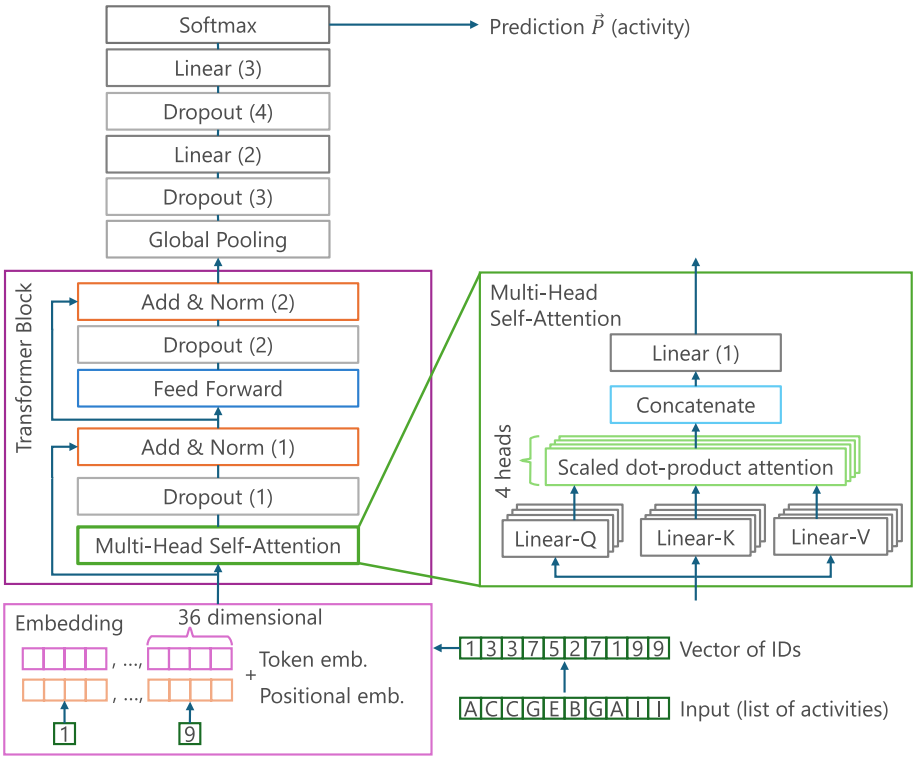

Fig. 1 Overview about the transformer architecture proposed in Bukhsh et al. (2021)

resulting in a continuous representation of the input element that captures positional, semantic, and syntactic properties (Vaswani et al. 2017). This resulting vector, denoted as $x_i \in \mathbb{R}^{1\times d_m}$, serves as input for the transformer block (Vaswani et al. 2017). Notably, these embeddings are automatically learned during training, allowing the model to dynamically adapt to the semantic and positional characteristics of the input data.

Transformer block The heart of the transformer architecture, and central to its capabilities, is the multi-head self-attention mechanism. This specialized form of attention significantly enhances the model's ability to capture relationships within sequences. In general, an *attention mechanism* allows deep learning models to dynamically focus on different parts of an input sequence when generating output. The core idea behind attention is to assign varying scores to different elements in the input, indicating how much focus the model should pay them when performing a task. Hence, it allows to prioritize most relevant information leading to more accurate and contextual output. Attention mechanism can be integrated into various deep learning architectures. In the transformer architecture introduced in Vaswani et al. (2017), a refined version of the attention mechanism, known as *self-attention* is proposed. This mechanism enables the model to attend to all elements within the input sequence simultaneously, enhancing its ability to capture both local and global dependencies.

Self-attention mechanism: In the self-attention mechanism, the embedding vector x_i is utilized in three distinct ways: as a *query vector* q_i, a *key vector* k_i, and a *value vector* v_i.

These vectors are obtained by multiplying the embedding vector x_i with the projections matrices $W_Q \in \mathbb{R}^{d_m \times d_k}$, $W_K \in \mathbb{R}^{d_m \times d_k}$, and $W_V \in \mathbb{R}^{d_m \times d_v}$:

$$q_i = x_i \cdot W_Q \in \mathbb{R}^{1 \times d_k}$$
$$k_i = x_i \cdot W_K \in \mathbb{R}^{1 \times d_k}$$
$$v_i = x_i \cdot W_V \in \mathbb{R}^{1 \times d_v}$$

Because W_Q, W_K, and W_V transform x_i in a lower-dimensional space, they are referred to as *projections*. These matrices are automatically learned during training. According to Vaswani et al. (2017) d_k and d_v are typically set to the same value. In order to enhance computational efficiency, these vectors are packed row-wise into matrices $Q \in \mathbb{R}^{n \times d_k}$, $K \in \mathbb{R}^{n \times d_k}$, and $V \in \mathbb{R}^{n \times d_v}$ (Vaswani et al. 2017).

We then employ Q, K, and V to calculate an attention score that reflects the relative importance of each element (query) in relation to the others in the input sequence. Specifically, each element in the input sequence is compared to the currently considered element by computing the matrix product $QK^T \in \mathbb{R}^{n \times n}$. The values of the resulting $n \times n$ matrix are scaled by dividing by $\sqrt{d_k}$ and then normalized by applying a row-wise softmax operation:

$$M_\sigma = softmax\left(\frac{QK^T}{\sqrt{d_k}}\right).$$

This normalization mitigates the risk of vanishing gradients and improves training efficiency (Vaswani et al. 2017). The resulting matrix M_σ is called the *attention score matrix* or simply the *attention matrix*[2]. Notably, the dimension of the attention score matrix depends on the length of the input sequence, resulting in attention score matrices of varying sizes for input sequences of different lengths.

Finally, each value vector v_i (contained in V) is multiplied with the attention scores:

$$Att_\sigma = M_\sigma \cdot V \in \mathbb{R}^{n \times d_v}.$$

This step is intended to reduce the impact of value vectors that receive very low attention scores. The resulting matrix Att_σ is the output of the self-attention mechanism.

Multi-Head Attention: To capture diverse relationships and patterns between the elements in the input sequence, we use h independent, parallel attention mechanisms, referred to as *attention heads* or *heads*. These heads are then combined into a so-called *multi-head attention* (Vaswani et al. 2017):

$$MultiHeadAttention_\sigma = concat(head_\sigma^1, \ldots, head_\sigma^h)W^O \in \mathbb{R}^{n \times d_m}$$

with $head_\sigma^j = Att_\sigma^j$.

In this formula, the attention score matrices from each attention head j are concatenated (i.e., the matrices are arranged side by side) resulting in a $n \times hd_v$ matrix. This concatenated matrix is then multiplied by a learned weight matrix $W^O \in \mathbb{R}^{hd_v \times d_m}$ to project the result back to the initial dimension d_m. In the result of the multi-head attention, each

[2] Please note that QK^T is often referred to as attention score matrix in the literature.

row represents an enriched context vector of an element of the input sequence, while each column represents a specific feature across all elements.

To ensure that each head does not require the full dimension d_m, the dimensions d_k and d_v are set to $\frac{d_m}{h}$. This adjustment keeps the total computational cost comparable to that of a single-head attention mechanism. The process transformer presented in Bukhsh et al. (2021) employs four attention heads, resulting in $d_k = d_v = 9$.

The core idea behind the multi-head attention mechanism is that each attention head can potentially learn different relationships or types of dependencies between the elements of the input sequence. Because each head operates independently with its own weight matrices, it can focus on distinct patterns and dependencies. By calculating multiple attention heads in parallel, the model can capture more diverse and complex relationships within the sequence than it would be possible with a single attention mechanism.

Interpretation of the attention score matrices: Fig. 2 depicts an example of the attention score matrices for all heads obtained for a prefix of a real-world event log. Interpreting these attention score matrices is essential for understanding the explanation approaches and is therefore described in detail. All attention score matrices can be interpreted identically, regardless of the head. Each attention score matrix illustrates how the elements in the input sequence relate to one another. In these matrices, rows and columns correspond to elements of the input sequence. While the rows represent these elements as queries, the columns represent them as keys. By examining a row, we can assess the importance the transformer model assigns to each other element in the input sequence from the perspective of the element represented by that row. In contrast, inspecting a column reveals the level of attention other elements in the input sequence assign to the key element associated with that column. The entries of the matrix indicate

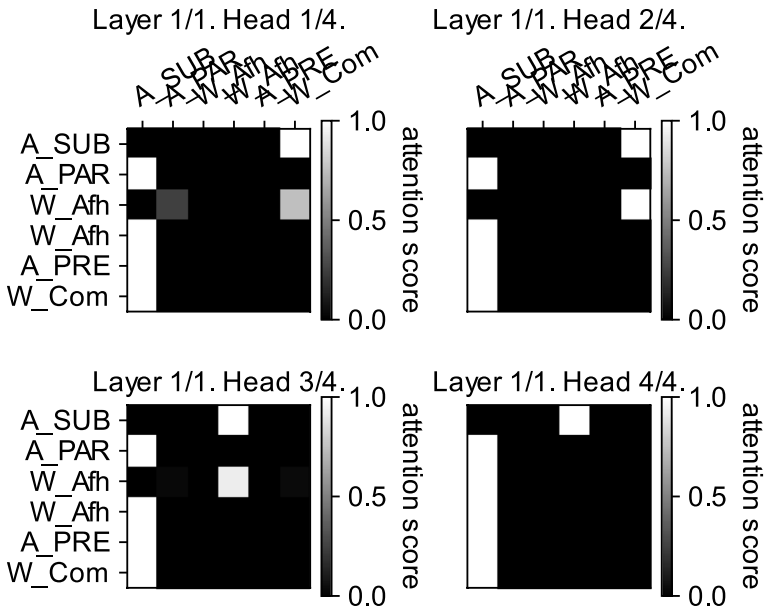

Fig. 2 Attention scores of all heads for the prefix ⟨A_SUB, A_PAR, W_Afh, W_Afh, A_PRE, W_Com⟩ of the BPIC12 event log

the strength of the relationship, i.e., the attention scores, between each pair of elements in the input sequence. In the visualization, higher attention scores are represented by lighter cells, while lower scores appear as darker cells.

Further processing of the attention output: The outputs of the multi-head attention mechanism are further processed and aggregated through different layers. First, a dropout layer follows the multi-head attention mechanism to prevent overfitting (Bukhsh et al. 2021). Layer normalization is applied next to stabilize the values of the activation functions, thereby helping to mitigate the risk of exploding or vanishing gradients. Following this, a position-wise feedforward network is applied, which operates independently of the position of elements in the input sequence. The purpose of this network is to increase the model's capacity by applying non-linear transformations that enable the network to capture complex interactions between the features encoded in the d_m-dimensional representation of each element. Thus, the feedforward network complements the overall architecture by playing a distinct role: while the attention mechanism models relationships between elements in the input sequence, the feedforward network focuses on refining and enhancing the internal representation of each individual element.

Output of the transformer The post-transformer block layers serve to prepare the model's output for the final prediction. Global pooling condenses the sequence information into a fixed-size vector. The linear layers then transform this vector, combining important features and adapting the model's internal representation to match the output format. Dropout reduces the risk of overfitting, and the softmax layer produces probabilities for each possible class. The softmax layer contains a neuron for each activity in the process, returning a probability distribution over the activities. Finally, the activity with the highest probability is selected as the predicted next activity using the *argmax* function (Bukhsh et al. 2021).

This probability distribution is called *prediction vector*. In the following, $\mathcal{M}$ denotes the trained transformer model and $p_\sigma = \mathcal{M}(\sigma)$ the softmax output for a given prefix σ. We call p_σ *prediction vector* and denote with $p_\sigma(a)$ the prediction score for activity $a \in \mathcal{A}$ in the prediction vector.

Related work

The inherent black box nature of deep learning models has raised significant concerns across various domains regarding their interpretability (El-Khawaga et al. 2022). Due to the shift in PBPM to deep learning models regardless of different prediction targets, this field of research is particularly affected by this problem. Explainable Artificial Intelligence (XAI) has emerged to address these challenges, evidenced by rapid growth and diverse contributions across application areas over the past decade (Nagahisarchoghaei et al. 2023). Exemplarily for this growth, we can see the survey of Rojat et al. (2021), which identifies 31 XAI approaches specifically for time series data. Remarkable here is that roughly one-third of these approaches utilize attention scores for generating explanations underpinning the potential of attention scores for XAI. Although it seems obvious that attention scores are suitable, there is an ongoing debate in literature, whether attention scores are trustworthy for explanation purposes, concluding that it must be

examined individually for each case (Wiegreffe and Pinter 2019; Jain and Wallace 2019; Serrano and Smith 2019). The works of Wiegreffe et al. (2019) and Serrano et al. (2019) propose a bunch of experiments that can be conducted to check the trustworthiness of attention scores. However, in the above-mentioned survey, none of the identified approaches include such evaluations.

In the following, we briefly discuss XAI approaches specific to the PBPM domain, with a strong focus on the next activity prediction. According to Carvalho et al. (2019) XAI approaches can be categorized as either *model-specific* or *model-agnostic*. Model-specific approaches are tailored to the inner workings of a particular model, whereas model-agnostic approaches can be applied to any machine learning model. Although such model-agnostic approaches are universally applicable they often provide less detailed insights into the decision-making process of the models.

The survey by Stierle et al. (2021) categorizes 20 XAI approaches within the PBPM domain and reveals that most are model-agnostic and limited to local explanations. Hence, there is a lack of global, model-specific explanation approaches in PBPM. When focusing only on model-specific explanation approaches for next-activity prediction, only three papers remain (Weinzierl et al. 2020; Sindhgatta et al. 2020; Hanga et al. 2020), but none of them employed the transformer architecture.

In Sindhgatta et al. (2020), an attention mechanism is used within an LSTM architecture. However, this mechanism differs from the self-attention mechanism used by the transformer architecture. Additionally, the approach only provides accumulated attention scores for each prefix position without further exploration. Also, they do not consider tests proposed by Wiegreffe and Pinter (2019); Serrano and Smith (2019) to verify the interpretability of the observed attention scores. Thus, the reliability of attention-based explanations in PBPM is still an open question, which we address in Pre-study: the relevance of attention scores section.

The *Explainable Next Activity Prediction (XNAP)* approach by Weinzierl et al. (2020) focuses on next activity prediction using LSTM models, assigning relevance scores to each activity in the input prefix via layer-wise relevance propagation. However, like Sindhgatta et al. (2020) their analysis is limited to a qualitative study, lacking a quantitative evaluation.

Hanga et al. (2020) also employ LSTMs but differ from the previously mentioned works by presenting explanations as directed and weighted graphs rather than assigning relevance scores. In these graphs, nodes represent activities, and the weighted edges denote the predicted likelihood of transitions between activities. Thus, this graph can be considered only as a Directly Follows Graph. While this approach highlights possible transitions according to the LSTM model, it is unclear whether this graph representation fully explains the model's decision-making process.

Beyond these directly related works, several other approaches (model-agnostic or local approaches as well as other prediction targets) offer valuable insights and inspiration for the approaches developed in this paper. For example, the LORE (Guidotti et al. 2018) approach approximates a black box model locally by generating instances close to a given input prefix and varying its features (e.g., continuous or categorical values). The LORELEY approach by Huang et al. (2022) extends LORE and tailors them to next-activity prediction by restricting the modifying operations to avoid violating the process

control-flow. However, as a model-agnostic approach, LORELEY differs from the objective of our approach and is not directly comparable.

Features used by a prediction model play a crucial role in XAI approaches. Several studies examine the impact of specific features on the model prediction. For example, Rizzi et al. (2020) identify features responsible for incorrect outcome predictions using model-agnostic, local explainers such as LIME and SHAP, and improve prediction accuracy by reducing the influence of these features. General guidelines for feature selection are further discussed by Stevens et al. (2022), who advocate for minimal, independent feature sets that do not compromise model fidelity in outcome prediction tasks. Similarly, Elkawaga et al. (2022) investigate to which extent explanations reflect data characteristics, showing that feature selection is crucial for both prediction quality and reliable explanations. However, the transformer architecture investigated in our study, exclusively utilizes the activity event attribute, making feature selection irrelevant for our purposes. Instead, we consider the events in the prefix as features and leverage attention scores to highlight parts of the input prefix deemed relevant by the model. While the aforementioned approaches primarily rely on input perturbations, Mehdiyev et al. (2021) propose a local explanation technique for outcome prediction that clusters the latent space of deep learning models and explains these clusters using decision trees.

Pre-study: the relevance of attention scores

At first glance, attention scores appear to offer an intuitive insight into the model's decision-making process. However, in the literature, it is controversially discussed whether attention scores can genuinely serve as explanations, coming to different conclusions depending on the considered task (Wiegreffe and Pinter 2019; Jain and Wallace 2019). Therefore, before using attention scores as a key component in an explanation approach for next activity prediction, we assess their reliability in this context, i.e., whether they indeed highlight activities decisive for a prediction.

To this end, we conduct a series of experiments. First, we assess whether the real importance of an activity (so-called *common feature importance*) correlates with the importance indicated by the attention scores. In this context, common feature importance estimates how crucial an activity is for the model's prediction by systematically removing or replacing it and observing the resulting impact on the model's output. This heuristic approach helps identify activities that are essential for determining the next step in the process (Experiment 1). Second, we investigate whether attention scores in general affect the prediction (Experiment 2). Finally, we examine the degree to which particular attention scores contribute to the prediction (Experiment 3).

Datasets For our study we use eight frequently used real-world event logs and four noise-free synthetically generated event logs (see Table 1). While synthetically generated event logs have the advantage that the degree of complexity can be controlled and the resulting explanations can be analyzed by referring to a process model, real-world event logs allow a more realistic evaluation of the effectiveness of the approach in practice even if the underlying process model is unknown (Klinkmüller et al. 2018). The

Table 1 Descriptive statistics of the event logs ($|\sigma|$ is the trace length)

Event Log	#Cases	#Act.	#Events	AVG. $\|\sigma\|$	Max. $\|\sigma\|$	#Variants
BPIC12	13087	24	262200	20.04	175	4366
BPIC12_O	5015	7	31244	6.23	30	168
BPIC12_W	9658	7	170107	17.61	156	2643
BPIC12_WC	9658	6	72413	7.50	74	2263
BPIC13_CP	1487	4	6660	4.48	35	183
BPIC13-I	7554	4	65533	8.68	123	1511
Helpdesk	4580	14	21348	4.66	15	226
Sepsis	1050	16	15214	14.49	185	846
Complex Model	5000	15	76535	15.31	77	4192
LDD	1000	10	7514	7.51	8	4
Looped And	1550	6	38016	24.53	94	1550
Sequence	1000	12	12000	12.00	12	1

We focused only on control-flow relevant characteristics since the transformer model only considers activities

real-world event logs are obtained from the *4 TU Center for Research Data*[3] and cover different domains:

- *BPIC12*: This event log contains records of three subprocesses of a loan application process in a Dutch financial institute. Beside the whole event log, we obtain three additional event logs (O, W, and WC) by different filtering operations. While BPIC12_O is derived from BPIC12 by only keeping events whose activities start with "O_", BPIC12_W only contains events whose activities begin with the prefix "W_". The BPIC12_WC is a refinement of BPIC12_W only keeping events whose lifecycle attribute is set to "complete".
- *BPIC13_CP* and *BPIC13 Incidents* are two event logs extracted from Volvo's IT incident and problem management system.
- *Helpdesk* is an event log from the IT service domain containing records of the helpdesk process of an Italian software company.
- *Sepsis*: This event log of a Dutch hospital is from the healthcare domain and contains anonymized records of the treatment of patients with symptoms of a sepsis condition.

The synthetic event logs were generated with the PURPLE (Burattin et al. 2022) event log generator out of four process models that can be find in Appendix 2.

General experiment setup All sub-experiments in this pre-study share a consistent setup. Each event log is randomly split into training and test data, with 70 % allocated for training and 30 % for testing. To ensure reproducibility, we set random seeds. For generating test samples, we extract all prefixes of at least length 1 from the traces in the test data. The transformer model is then trained on the training data using the standard hyperparameters (see Appendix 1) defined in Bukhsh et al. (2021). The resulting test

[3] https://data.4tu.nl/

samples are subsequently used to obtain predictions, enabling an analysis of the influence of attention scores on the model's performance.

To check whether the predictions for two prefixes σ and μ are close to each other, we compute the *cosine similarity* between the corresponding prediction vectors p_σ and p_μ to quantify their similarity (Manning et al. 2008):

$$cosineSim(p_\sigma, p_\mu) = 1 - \frac{p_\sigma \cdot p_\mu}{\|p_\sigma\| \cdot \|p_\mu\|}.$$

If $cosineSim(p_\sigma, p_\mu)$ is greater or equal to an a-priori defined threshold δ_{sim}, we consider both prediction as close together. We chose as default value for $\delta_{sim} = 0.8$.

Since each prediction vector can be understood as a probability distribution over the activities of the process, we quantify the difference between two prediction vectors, p_1 and p_2, using a common statistical distance metric known as the *Total Variation Distance* (TVD) (Jain and Wallace 2019):

$$TVD(p_1, p_2) = \frac{1}{2} \cdot \sum_{i=1}^{|A|} |p_1^i - p_2^i|, \text{ where } p^i \text{ denotes the } i\text{th component of } p.$$

For further analysis, we interpret the attention scores as a probability distribution over the elements in the input sequence. Following the notation of Transformer architecture and attention mechanism section, we denote with M_σ^j the attention score matrix of head j obtained for an input sequence σ. To access a particular attention score within M_σ^j, we write $M_\sigma^j(i,k)$ to identify the score in the i-th row and the k-th column. Building on this, we can define both the attention score distribution for a head and the attention score distribution across all heads, respectively:

Definition 6 Let M_σ^j be the attention score matrix of head j obtained for an input sequence σ and

$$\begin{aligned} v_\sigma^j = (&M_\sigma^j(1,1), M_\sigma^j(1,2), \ldots, M_\sigma^j(1,|\sigma|),\\ &M_\sigma^j(2,1), M_\sigma^j(2,2), \ldots, M_\sigma^j(2,|\sigma|),\\ &\ldots\\ &M_\sigma^j(|\sigma|,1), M_\sigma^j(|\sigma|,2) \ldots M_\sigma^j(|\sigma|,|\sigma|)) \end{aligned}$$

the vectorization of M_σ^j. The attribution score distribution of head j for σ is then defined as

$$\alpha_\sigma^j = \frac{v_\sigma^j}{\|v_\sigma^j\|},$$

where $\|\cdot\|$ denotes the L1-norm, i.e., the component-wise sum of the vector components.

The attention score distribution across all heads is then defined as follows:

Definition 7 Let σ be an input sequence, h the number of heads, $v_\sigma^1, \ldots, v_\sigma^h$ the vectorization of the heads' attention score matrices obtained for σ, and $v_{sigma} = (v_\sigma^1, \ldots, v_\sigma^h)$ the

flattened representation across all heads. The attention score distribution is then defined as

$$\alpha_\sigma = \frac{\nu_\sigma}{\|\nu_\sigma\|},$$

where $\|\cdot\|$ denotes the L1-norm.

For quantifying the divergence between two attention score distributions α_1 and α_2, we use the *Jensen-Shannon-Divergence* (JSD) (Jain and Wallace 2019)

$$JSD(\alpha_1, \alpha_2) = \frac{1}{2} KL\left(\alpha_1 \parallel \frac{\alpha_1 + \alpha_2}{2}\right) + \frac{1}{2} KL\left(\alpha_2 \parallel \frac{\alpha_1 + \alpha_2}{2}\right),$$

that is based on the *Kullback Leibler Divergence* (KL) defined in Kullback (1952):

$$KL(P \parallel Q) = \sum_{x \in X} P(x) \log\left(\frac{P(x)}{Q(x)}\right).$$

For comparing prediction vectors and attention score distributions, we use with TVD and JSD two different metrics. This choice is motivated by the distinct characteristics of TVD and JSD. The probability distribution in prediction vectors often focus heavily on one or few classes, meaning that a changed prediction typically goes in hand with strong shifts in probability mass. Because TVD effectively captures substantial shifts in a probability distribution (due to the absolute differences in the components of the prediction vectors), it is particularly suited for comparing prediction vectors.

JSD, on the other hand, is more appropriate for comparing attention score distributions because these scores tend to be more evenly distributed. JSD effectively captures subtle redistributions in focus, which may indicate a shift in the model's internal reasoning, even if the final prediction remains stable. By emphasizing structural changes in distributions, JSD reveals meaningful changes in attention patterns.

Using both metrics allows us to capture different aspects of the model's behavior: TVD identifies impactful changes in the model"s prediction, while JSD reveals internal shifts in the attention score distribution that may not directly affect the prediction but still indicate meaningful changes in the model's focus.

Experiment 1 – Feature Correlation In the first experiment, we assess whether the real importance of an activity (so-called *common feature importance*) correlates with the attention score feature importance, i.e., the feature importance indicated by the attention scores. The term feature is used in the context for an element of the input sequence.

Since, we cannot manually decide with a domain expert for each prediction, which activity (in this context feature) would be the decisive ones for a prediction, we rely on a heuristic to determine the common feature importance. The core idea behind this heuristic is to remove or perturb one element in the input sequence at a time, either by masking or replacing it with another activity and observing its effect on the models' prediction. Masking means that the element is replaced with a padding symbol (_), indicating to the model that there is no element.

Before calculating the common feature importance, we extract a directly follows graph (DFG) from the given event log. The DFG is then used to select activities that can be used to replace activities in the traces in a realistic way. Realistic means that it is likely that the modified trace represents a valid process execution.

Calculation of common feature importance: For calculating the common feature importance f_{com} for a trace σ, we proceed as follows: We first create a collection of modified versions of the original trace σ. Therefore, we iterate over all events in σ to generate for each event a set of modified traces in which the particular event e_i has either been masked out or its associated activity has been replaced by another activity (or sequence of activities). In the case of a replacement, we select the replacement activity in dependency of the position of e_i with the help of the DFG:

- If e_i is the first event of σ, we identify all possible start activities of the process in the DFG. Then we replace e_i with each possible start activity to get several modified traces.
- If e_i is the last event in the trace, we replace its activity with all activities that can succeed the preceding activity $\pi_a(e_{i-1})$ according to the DFG.
- For any other position (i.e, if e_i is neither the first nor the last event), we substitute e_i with all possible paths in the DFG that connect the activity of the preceding event, $\pi_a(e_{i-1})$, to the activity of the subsequent event, $\pi_a(e_{i+1})$.

The union L_M of all modified versions of σ obtained across all variations can then be used to compute common feature importance values per activity. Therefore, we initially set the common feature importance for each activity to zero. For each modified trace $\sigma_m \in L_M$, we first determine the replaced activity and compute then the cosine similarity of the prediction vector obtained for σ_m and the prediction vector for the original trace σ. This similarity value is then added to the common feature importance of the replaced activity. It is important to note that we add this score regardless whether the prediction for σ and σ_m differs, because we are interested in influence of the activity on the prediction. The larger the difference of the prediction vector, the higher is the impact. After evaluating all modified traces, we compute the mean of all similarity values belonging to the same replaced activity. Finally, we normalize the mean values so they sum up to 1.0 to obtain a probability distribution over the activities. Please note, that this feature importance is independent of the attention scores and only relates features and prediction.

Calculation of attention score feature importance: For calculating the attention score feature importance f_{att}, we rely on the same collection of modified traces L_M as before. If the prediction of the original trace and a modified trace are similar (i.e., their cosine similarity is lower than δ_{sim}), we determine the attention scores per activity from the transformers' attention score matrices. After inspecting all modified versions, we compute the mean of the cumulated attention scores per activity. It is crucial that we only consider modified traces with similar predictions to the original trace, because we want to identify, which activities are important for a particular prediction.

Finally, we calculate Kendall's tau rank correlation τ (Kendall 1938) between common feature importance and attention score feature importance. Concerning interpretability Kendall's tau is scaled into the continuous range $[-1, 1]$, where -1 indicates a perfect

negative correlation, 0 indicates no correlation, and +1 indicates a perfect positive correlation. Hence, a low correlation means, that the relative importance of the features is not sufficiently reflected in the attention scores. Thus, the explanations derived from the attention scores would not use the same reasoning as the transformer to be explained. Hence, for reliable attention scores large values are preferable.

The results of the correlation of common and attention score feature importance are shown in Fig. 3. For the majority of the event logs the median correlation is slightly larger than 0.5. Following the guidelines proposed in Akoğlu (2018) after which values $\tau < 0.3$, $0.3 \leq \tau \leq 0.5$, $\tau > 0.5$ indicate a weak, moderate, and strong correlation respectively, we observe a moderate to strong association. Nevertheless, for some of the event logs, the correlation is closer to zero, indicating no correlation. The latter applies to BPIC12, BPIC12-WC, and the Sepsis event log. In summary, the correlation between the common feature importance and the attention score feature importance is often significant, giving a first indication of reliable attention scores.

Experiment 2 – Attention Mechanism Parameter Manipulation The next experiment examines the necessity of the attention mechanism, adhering to the experimental setup outlined in Wiegreffe and Pinter (2019). To this end, we first train exactly one baseline model $\mathcal{M}_b$ without any modifications in the training process. Then we train multiple manipulated transformer models where we either randomly initialize the attention parameters (i.e., matrices W_Q, W_K, W_V, and W_O) or freeze the attention parameter to uniform values during training. The latter means that the attention score mechanism is completely eliminated, while the first means that the training process probably ends with different attention scores. Both modified training procedures are repeated five times to mitigate the influence of change, each initiated with unique random seeds to ensure variability across the trials.

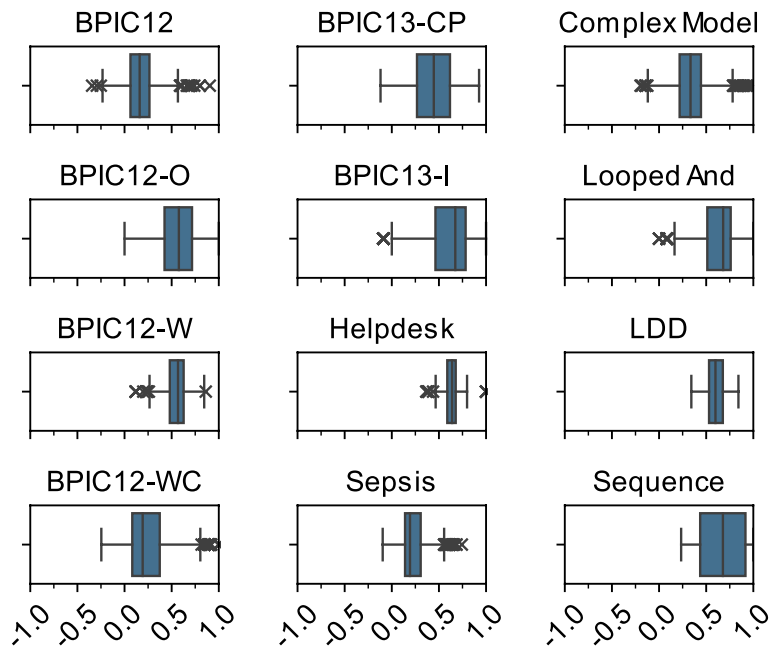

Fig. 3 Results for the Kendall Tau correlation of common and attention feature importance

Subsequently, each manipulated transformer $\mathcal{M}_m$ (both the randomly initialized and the frozen) is compared against the baseline $\mathcal{M}_b$. To this end, we send each test sample through all trained models (i.e., the baseline model and the manipulated transformers) and extract each time the corresponding attention score distribution and the prediction vector. The attention score distributions and prediction vectors derived from $\mathcal{M}_b$ are then compared to those obtained from the corresponding $\mathcal{M}_m$ using JSD and TVD, respectively. To aggregate the values, we compute the average JSD and TVD values across all test samples for each model.

The rationale behind this experimental design is to assess whether varying attention score distributions (indicated by higher JSD values) lead to invariant predictions (indicated by lower TVD values) (Wiegreffe and Pinter 2019; Jain and Wallace 2019). Thus, if identical predictions can be achieved with substantially different attention scores, this would imply that the attention mechanism barely affects the decision process of the model. Hence, the attention scores would have only minimal explanatory value.

To facilitate interpretation, we plot the mean JSD against the mean TVD achieved for each model (see Fig. 4). Models positioned further to the right and lower in the plots represent cases with less reliable attention scores. Notably, the majority of the event logs, regardless of whether they are real-world or synthetically generated, are positioned in regions indicating reliable attention scores. In cases where the models tend to be on the right side (Complex Model, Looped And, BPIC12, and BPIC12-W) the TVD values are significantly above zero. For Sequence and LDD, however, TVD values remain close to zero, suggesting that predictions are only minimally affected by changes in attention scores. Due to the simplicity of Sequence and LDD this observation is not surprising, since for most of the test samples the next activity is obvious, so that the model takes other options barely into account. At the same time, both Sequence and LDD are also on the left side, indicating only minor variations in the attention scores. Overall, these results suggest that the attention scores can be trusted.

To further investigate the stability of attention scores, we analyze the maximum attention value in a distribution, i.e., $max(\alpha_\sigma)$, in relation to JSD. The motivation for this analysis is the hypothesis that high maximum attention values indicate a model's strong focus on a particular input element. If this focus is consistent across different model initializations, it could suggest that strong peaks in the attention distribution align with stable and meaningful model behavior. Conversely, if high maximum attention values are associated with high JSD values, it would imply that even seemingly confident attention peaks may be unreliable and vary substantially across different model instances. By examining this relationship, we aim to assess whether strong attention peaks can serve as robust explanatory indicators.

Following Jain and Wallace (2019), we conduct a binning procedure in which all attention score distributions of the samples regardless of the model are categorized based on its maximum attention value. By combining results from multiple models, the analysis ensures that observed patterns are robust and not artifacts of specific model initializations. Specifically, each attention score distribution is assigned to one of four distinct bins, each defined by a numerical range for maximum attention values. Through the binning, we aim to examine how varying degrees (represented by the ranges of the bins) of selective focus correlate with shifts in attention distributions.

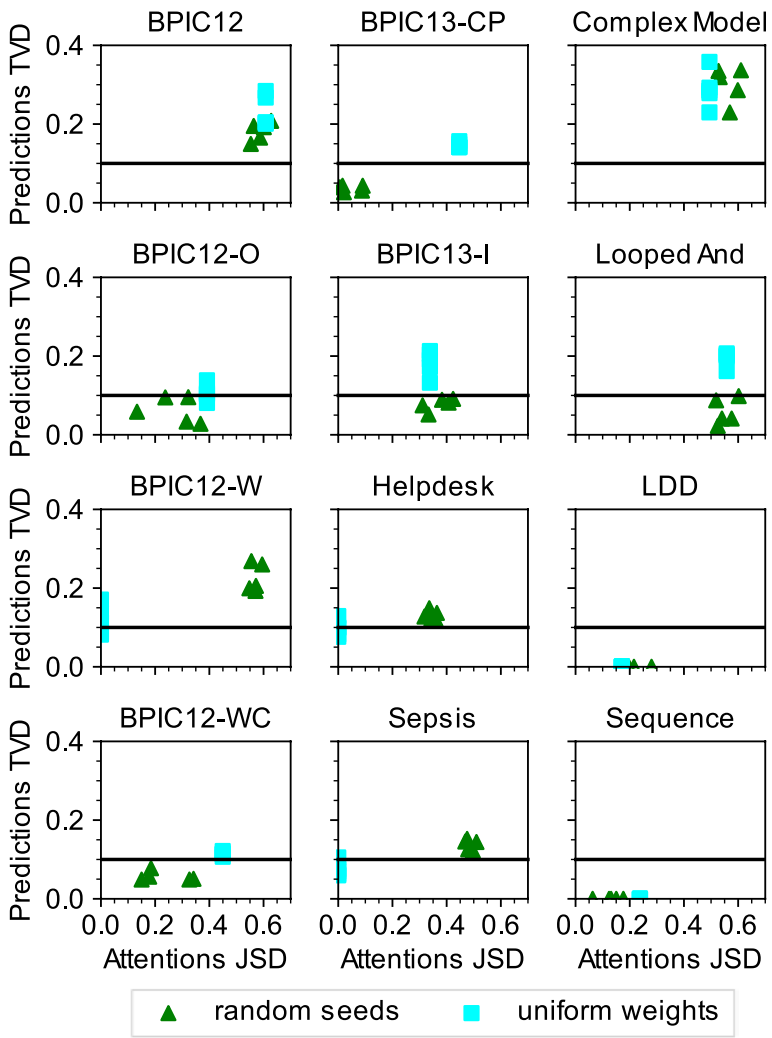

Fig. 4 JSD vs. TVD plots. Rectangles = models with frozen weights, triangles = seeded base models. JSD can only take values between 0 and $log(2) = 0.693$. The horizontal line at 0.1 serves as reference value to compare the process-specific results, with results obtained on general machine learning datasets. The value stems from Wiegreffe and Pinter (2019), where all TVD values were lower than 0.1

Figure 5 visualize the distribution of JSD divergences (compared to the base model) as a function of the maximum attention value across all models in form of a violin plot. The height of each violin is scaled according to the number of instances in each bin, indicating how often JSD values lie in certain ranges. Thus, a wide area in the violin means that many JSD values are concentrated in this area. The violins can be interpreted as follows: A right-weighted violin means that the JSD values in this bin tend to be high. This indicates that the attention values for this bin are susceptible to manipulation. In contrast, a left-weighted violin (wider left, with low JSD values) means that the JSD values in this bin are predominantly low. This indicates that the attention scores for this bin are robust and less manipulable.

The plots reveal that most attention score distributions fall within either the first or last bin, with minimal representation in the second and third bins. This pattern suggests

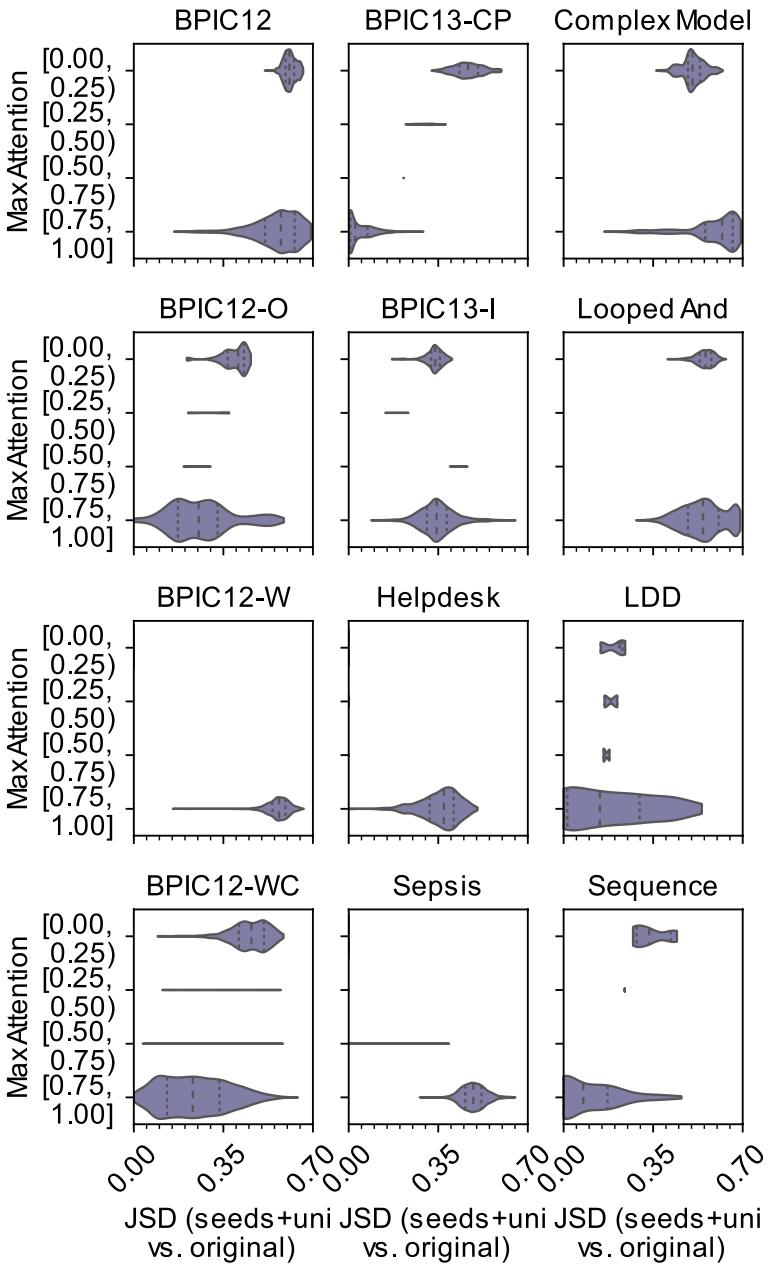

Fig. 5 All JSD vs. maximum value in the attention score distribution violin plots (scaled by count). The height is scaled by the count of instances per bin

that the model tends to react to certain elements either very strongly or hardly at all. Since the middle bins (second and third bins) are only minimally represented, the model rarely shows moderate attention to an element.

If attention score distributions fall into the last bin (0.00–0.25), there is often a rightward tendency. The first bin (0.75–1.00), on the other hand, shows a more heterogeneous picture, with some datasets tending to the left (BPIC13_CP, BPIC12_O, BPIC13-I, BPIC12_WC, Sequence) and two datasets (Complex, Looped And) showing a strong rightward tendency. The remaining logs are more centralized. Of particular interest are

cases where the last bin shows a rightward tendency while the first bin shows a leftward tendency (e.g., BPIC12_W, Sequence, BPIC13_CP). In these cases, instances with low attention scores are more susceptible to manipulation than those with higher attention scores.

Overall, this binning approach shows that the interpretability of attention scores can vary within a dataset and that particular maximum attention thresholds may serve as indicators of model robustness. The detailed analysis also supports the initial findings from the first part of the experiment that the attention scores are predominantly trustworthy, suggesting they are suitable for explanatory purposes.

Experiment 3 – Attention Score Masking In a third experiment, we investigate how particular attention values affect the prediction (see Fig. 6). In detail, we want to check whether the attention scores are closely linked with the information in the input sequence. Therefore, we examine whether removing an element from the input sequence has the same effect in terms of prediction as removing the element's attention scores.

To do this, we once mask elements in the input sequence and once the corresponding attention scores in the heads' attention score matrices and compare the model outputs via TVD. In the input sequence, we mask elements by replacing them with a padding symbol (_), indicating to the model that there is no element. In contrast, masking the attention scores is achieved by setting all values in the relevant rows and columns of the attention score matrices to zero. Figure 6 illustrates this conceptual difference between masking the input prefix and the attention scores of the heads' attention score matrices.

We perform this procedure for all test samples, whereby each element in the sample is masked one time. For each of the masked samples, we obtain two prediction vectors: a prediction vector p_m for the masked sample and a prediction vector p_{am} for the unmasked sample, where we masked the attention scores. We then compare p_m and p_{am} by calculating their TVD.

Low TVD values suggest that masking an element in the input sequence has a similar impact on the prediction as masking it within the attention score matrices. From this,

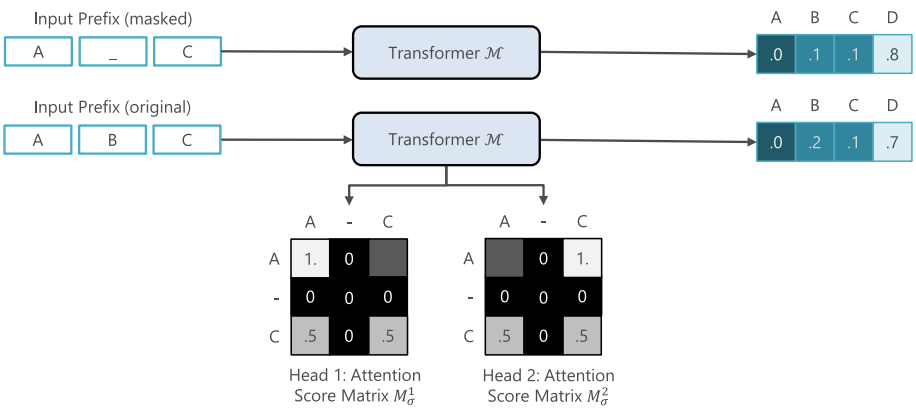

Fig. 6 The former masks events (here activity B) directly in the prefix. The latter only masks rows and columns corresponding to the event in the attention score matrices. Predictions may be different

it can be concluded that particular attention scores have relevance and can serve as an explanation. We observe that the histograms of TVD values (see Fig. 7) show a strong left skew, indicating that masking in the input prefix leads to almost identical predictions as masking in the attention score matrices. Minor deviations from this behavior appear only in the histograms for the Sequence and LDD event log.

Conclusion The overall result of this series of experiments is that the attention scores possess good reliability. Only very simple event logs such as Sequence or LDD (both

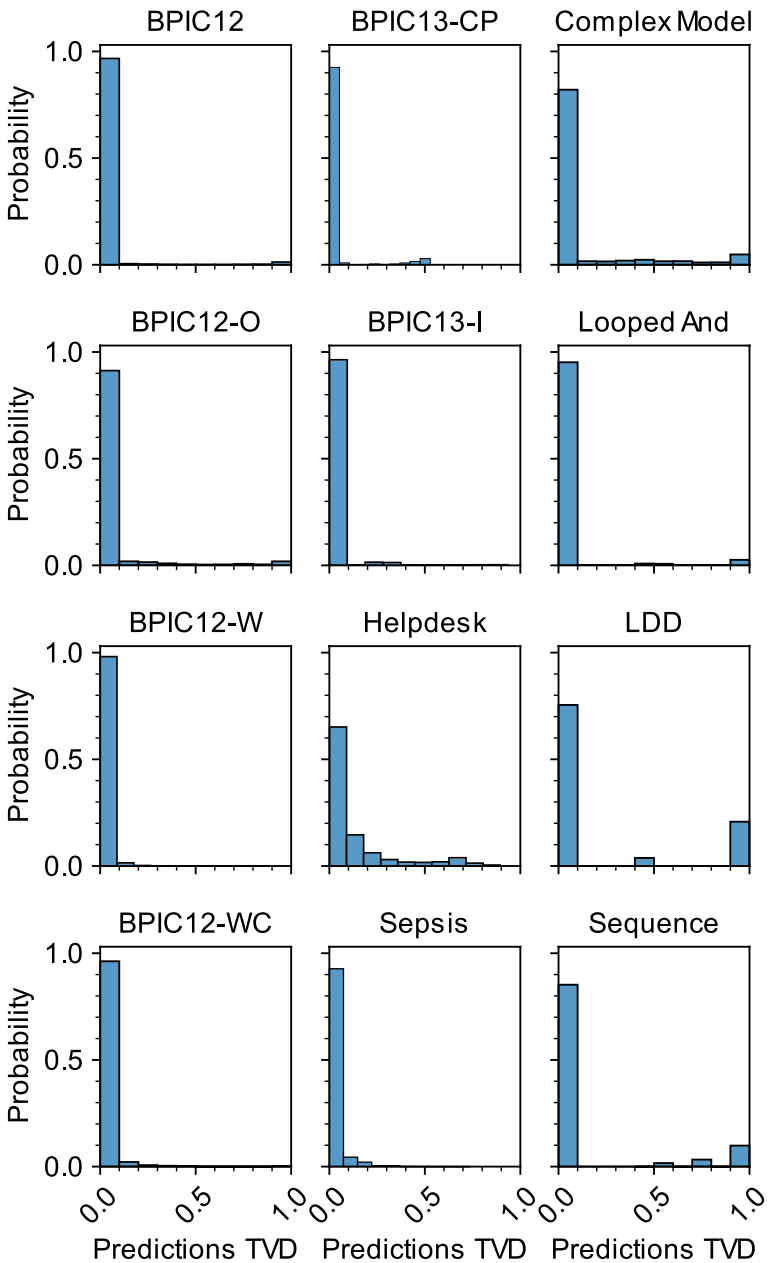

Fig. 7 Variation between predictions TVD between masked elements in the prefix and only masked attention scores matrix

synthetically generated) deviate slightly from this observation, although both still have sufficient trustworthiness. This finding that the attention scores appear less reliable for very simple datasets is also consistent with the results in the study of Wiegreffe and Pinter (2019), where the same observation was made for datasets from other domains.

The reason for this anomaly lies in the simplicity of the underlying task so that the model does not require the powerful attention mechanism at all. Based on the results obtained, it is justified to build explanation approaches with attention scores as a pivotal component.

Therefore, especially the results from Experiment 2 are beneficial, for getting deeper insights how strong the attention scores are pronounced. Most of the time the attention score distributions fall either in the first bin (maximum attention score very high) or the last bin (maximum attention score very low) but barely in the second or third bin. Hence, we have either at least a very high attention score (first bin) or a lot of small attention score (last bin). This observation helps us to define thresholds, when to consider an element in the input sequence as relevant or not.

Explanation approaches

In this section, we introduce two global transformer-specific explanation approaches. Each explainer receives as input a trained transformer prediction model and a set of prefixes L to be explained. Both explainers construct an interpretable directed graph that visualizes the control-flow of the process, serving as an indicator of the extent to which the prediction model captures the control-flow of the process. Because our preliminary study in Pre-study: the relevance of attention scores section provides strong evidence that attention scores are reliable for next activity prediction, our explanation approaches leverage them as a crucial component.

Explainer 1: backward explainer

The first explainer (so-called *Backward Explainer*) generates explanations for individual prefixes and subsequently integrates these into an overarching explanation for all prefixes in L. This procedure is explained in the remainder of this section and involves creating a local graph G_σ for each prefix $\sigma \in L$, which is then merged directly into the global graph G.

Creating a local graph An essential step in constructing a local graph G_σ for a prefix σ is to utilize the heads' attention scores to identify relevant activities that are decisive for the prediction p_σ. To do this we proceed as follows: An activity is considered relevant for the prediction p_σ if its aggregated attention score across multiple modified versions of σ is sufficiently high. To generate these modified versions of σ, we apply randomly modification operations. Each modified prefix σ_m is then passed to the transformer to obtain prediction vector p_{σ_m} and the attention score matrices $M^i_{\sigma_m}$ for a head i.

To check whether the prediction for σ_m is still close to the prediction of σ, we compute the cosine similarity between p_{σ_m} and p_σ and compare it to a predefined threshold δ_{sim}.

If σ_m satisfies this condition, the attention scores for each element j in σ_m are aggregated across the different heads as follows:

$$S_j = \sum_{n=1}^{|\sigma_m|} \left(\sum_{i=1}^{n} M^i_{\sigma_m} \right)_{nj}.$$

In this equation attention score matrices $M^i_{\sigma_m}$ from each head are first summed component-wise, and then the jth column is summed. This yields an aggregated attention score vector $\eta_{\sigma_m} = (S_1, ..., S_{|\sigma_m|})$ for each modification σ_m containing a total attention score for each event. This procedure is illustrated in Fig. 8.

Next, the total attention scores are further aggregated per activity to a vector ψ_σ by summing up the scores for all events belonging to a certain activity. The vector ψ_σ is then normalized to range within [0, 1]. We denote the total attention score for activity a as $\psi_\sigma(a)$. Finally, we filter activities based on a threshold δ_{attr}, keeping only activities with $\psi_\sigma(a) > \delta_{attr}$ to get only those activities with particularly high total attention scores. Thus, the set of relevant activities for input sequence σ is defined as $\mathcal{A}_r = \{a \in \mathcal{A} \mid \psi_\sigma(a) > \delta_{attr}\}$. As a default value we set δ_{attr} to 0.1. This filtering is essential to exclude irrelevant elements that may have low but non-zero attention scores. Without this step, nearly all activities would be classified as relevant.

Analogously, we identify the most likely next activities, denoted as $\mathcal{P}_r$, from the prediction vector p_σ, by selecting only those activities a with prediction scores exceeding an a-priori defined threshold δ_{pred}, i.e., $p_\sigma(a) > \delta_{pred}$. As a default value for δ_{pred} we used 0.1. We do not limit the selection to a fixed number of activities from p_σ, because depending on the process the confidence of the model varies: for some prefixes, it may indicate that only one activity is probable (e.g., in case of a strict sequence), while in others (e.g., at an AND split), multiple activities appear viable to be executed next. Building on this we can formally define local graph as follows:

Definition 8 Let σ be an input sequence, $\mathcal{A}_r$ the set of relevant activities identified in σ, and $\mathcal{P}_r$ the set of likely next activities for σ. A local graph for σ is a tuple $G_\sigma = (V, E)$ with

- $V = \mathcal{A}_r \cup \mathcal{P}_r$ being the set of nodes and
- $E = \{(s, t) \mid s \in \mathcal{A}_r, t \in \mathcal{P}_r\}$ being the set of edges.

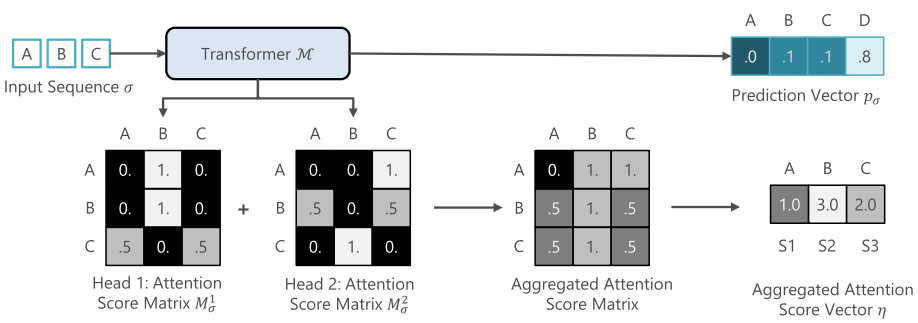

Fig. 8 Determining aggregated attention scores for each event in the prefix, assuming that $\mathcal{M}$ possesses two heads

Thus, for constructing the local graph G_σ, we use all elements in $\mathcal{A}_r$ and $\mathcal{P}_r$ as nodes and establish edges between all nodes from $\mathcal{A}_r$ to all nodes from $\mathcal{P}_r$.

Let us illustrate the procedure on the example depicted in Fig. 9. Given the prefix $\sigma_1 = \langle B, A, C, B, E \rangle$, the aggregated attention scores highlight activities B and C as relevant for the prediction, i.e., $\mathcal{A}_r = \{B, C\}$. Additionally, the prediction vector p_{σ_1} suggests that either activity B or D is likely to occur next, i.e., $\mathcal{P}_r = \{B, D\}$. According to Definition 8 we get nodes $V = \mathcal{A}_r \cup \mathcal{P}_r = \{B, C, D\}$ and edges $V = \{(C, B), (B, B), (B, D), (C, D)\}$. This graph can then be interpreted as follows: for executing activity D, it is crucial that activities B and C have been executed before.

Merging local graphs Each local graph G_σ is directly integrated into the global graph G. This integration involves adding the vertices and edges of G_σ to G if they are not yet included. To illustrate this step, suppose we have, in addition to the local graph G_{σ_1} from our example in Fig. 9, a second local graph $G_{\sigma_2} = (\{A, C\}, \{(A, C)\})$. The resulting graph after merging G_{σ_1} and G_{σ_2} would then be $G = (\{A, B, C, D\}, \{(C, B), (B, B), (B, D), (C, D), (A, C)\})$.

After merging a local graph into G, we apply a pruning step to remove undesired shortcuts that are unlikely to occur in the underlying process. Such shortcuts can arise if an activity in the front part of σ receives high attention for the prediction, but this activity is only indirectly required for the predicted activity, in that sense that this activity and the predicted activity are not directly sequential. To bring the global graph closer to a process model, we apply as heuristic, that we remove such shortcuts, that lead from an activity lying long back in the prefix to an activity that occurs later in the prefix. Formally, we remove edge $(u, v) \in E$ if there exist edges $(u, a_n) \in E$ and $(a_n, v) \in E$, for an activity a_n. Figure 10 illustrates this step on a short example. Please note, since we generally remove shortcuts there is a risk, that we may remove a desired shortcut. Since the idea is that we analyze all prefixes in an event log, the intention was that, direct connections are sufficient.

Explainer 2: attention exploration explainer

The Backward Explainer has a key limitation: it directly defines the edges of both the local and global graph when analyzing a prefix. Once an edge is inserted, it remains in

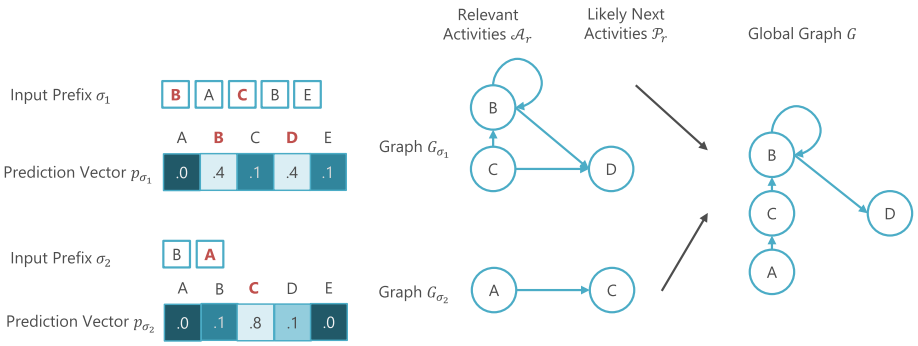

Fig. 9 Example for BackwardExplainer. Relevant activities and likely next activities are highlighted boldfaced in red color in the input prefix and prediction vector

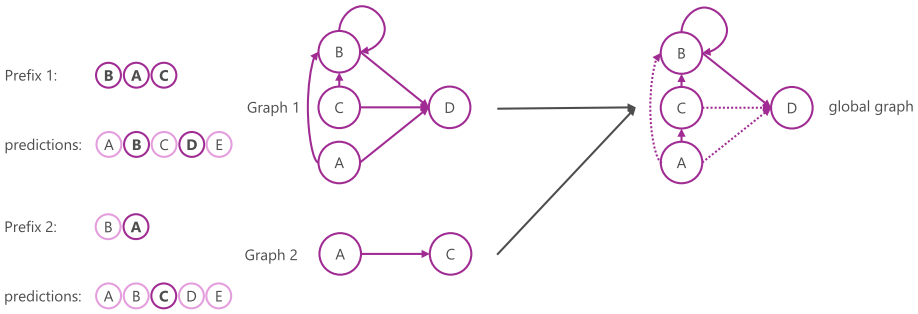

Fig. 10 Example for undesired shortcuts. In the prefixes we boldfaced the relevant activities. Also the relevant predictions are boldfaced. Graph 1 and 2 show the corresponding local graphs for the prefixes 1 and 2. In the global graph we marked shortcuts as dashed lines

the global graph unless it is identified as a shortcut and removed during the pruning step. Thus, existing edges are not updated with insights gained from subsequent prefixes. To address this limitation, we move away from local graphs and postpone the decision to insert an edge until all prefixes have been explored. Therefore, we introduce the so-called *relevance scores* for activities that are calculated across all prefixes, which then serve as the basis for the construction of the global graph. We call this advanced explainer the *Attention Exploration Explainer*.

Determining relevance For determining a relevance score of an activity across all prefixes, we cannot simply sum up its corresponding attention scores because that would lead to a continuous increase with each additional prefix. Hence, we need a scoring mechanism that increases the relevance score of an activity when it is relevant in a prefix and decreases it when the attention score for an activity is consistently low across prefixes.

However, we cannot use the attention values in our scoring mechanism directly, because in the transformer architecture attention values are inherently non-negative. Thus, we introduce negative attention scores to encode minimal impact on the prediction. In this way, a positive score for an activity in a prefix can be seen as a weighted vote for adding an edge, while a negative score as a vote against edge insertion in the global graph.

First, we identify relevant activities $\mathcal{A}_r$ in σ following the same method used in the Backward Explainer. Additionally, we store the positions of these relevant activities in σ in a set denoted as $\mathcal{I}_\sigma$. Next, we evaluate the individual and collective impact of activities on the prediction using all subsets of $\mathcal{I}_\sigma$. Thus, we consider the power set of $\mathcal{I}_\sigma$, i.e., the set of all subsets, and denote them with $P(\mathcal{I}_\sigma)$. For each subset $r \in P(\mathcal{I}_\sigma)$, we consider two scenarios: *(i) masking out most* and *(ii) masking out few*. In the "masking out most" scenario, we mask all positions not included in r. This has the effect that we exclude activities that are considered as irrelevant according to their attention scores (i.e., activities not contained in $\mathcal{A}_r$). In contrast, the "masking out a few" scenario masks positions within r, so that the relevant activities are excluded. Both masking scenarios are illustrated exemplarily in Fig. 11.

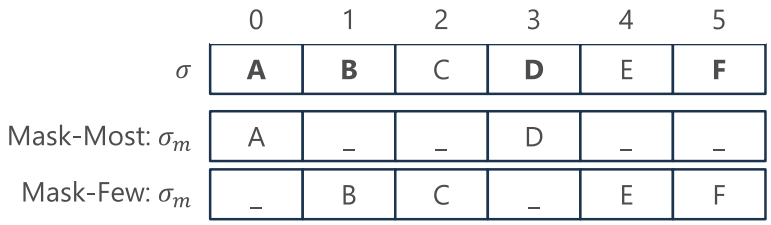

Fig. 11 Comparison between masking a few and masking most activities. The positions of relevant activities (boldfaced) are $l_\sigma = \{0, 1, 3, 5\}$, whereas the subset $r = \{0, 3\}$ is depicted

Next, we compare σ with all of its masked versions. In the following, we denote a masked version of σ with σ_m, regardless of the applied masking strategy. Then, both σ and σ_m are passed through the transformer to obtain prediction vectors p_σ and p_{σ_m}, respectively. Similarly, as in the Backward Explainer, we identify for prefix σ the most likely next activities $\mathcal{P}_r$ from p_σ and compute the aggregated attention score vectors per activity ψ_σ and ψ_{σ_m}.

Based on this, we calculate a *score matrix* K_{σ_m} for each masked version of a $\sigma \in L$. In this score matrix each activity $a \in \mathcal{A}$ has a corresponding row and column. Each entry in K_{σ_m} represents a vote for or against an edge between, the activities of the corresponding row and column.

The detailed procedure for computing the score matrix is outlined in Algorithm 1 and is independent of the used masking scenario. In this algorithm, we compute a relevance score for each activity in the masked prefix (Algorithm 1, l.2–15). Depending on whether the activity is masked or not in σ_m, the calculation differs:

- For masked activities: The score is calculated as the product of the prediction score and attention scores. If the prediction for a masked activity aligns with its prediction in the unmasked prefix, the score is negated, indicating reduced relevance due to the activity's negligible impact on the prediction. The intention behind multiplying $p_\sigma(a)$ with $\psi_\sigma(a_m)$ is to weight the probability that an activity a is executed next by the attention score of the masked activity. This weighting is meaningful because:
  - High $p_\sigma(a)$ and high $\psi_\sigma(a_m)$: Activity a is a likely prediction and a_m has strong impact on this prediction. Hence, there should be a strong vote for or against an edge between a and a_m.
  - High $p_\sigma(a)$ and low $\psi_\sigma(a_m)$: Activity a is a likely prediction, but a_m seems not to be decisive for this prediction. Thus, a moderate vote for or against an edge is required.
  - Low $p_\sigma(a)$ and high $\psi_\sigma(a_m)$: Activity a is unlikely to be executed next and a_m receives much attention from the model. Similar to the previous case, a moderate vote is needed.
  - Low $p_\sigma(a)$ and low $\psi_\sigma(a_m)$: It is unlikely that a is executed next and a_m receives barely attention. Thus, there is no indicator for a meaningful edge between a and a_m.

- For non-masked activities: The score is calculated differently. If an activity's prediction value remains unchanged under masking, the score is computed as the product of its attention score and its prediction score from the unmasked prefix. The intent here is identical to those in case of the masked activity. If the prediction changes, however, the score is calculated as the product of the absolute differences in both attention scores and predictions, reflecting the impact of the activity in the masked prefix. This ensures that an appropriately weighted vote for or against an edge between a_n and a is given between activities that result in strong changes in the attention scores and activities that show clear changes in the prediction score. In both cases, the multiplication ensures that the score reflects the actual importance of the activity: Important activities with a high level of attention and strong influence are given a higher score, while inconspicuous or less influential activities are given a correspondingly lower score.

The resulting score is then entered in the corresponding cell in the score matrix, where the row represents the predicted activity and the column corresponds to the masked activity.

All matrices K_{σ_m} within the same masking scenario are then summed to get two scenario-specific score matrices K_σ^{few} and K_σ^{most}. These matrices provide a comprehensive assessment of the relevance of each activity within σ, taking into account the influence of its presence or absence across different combinations.

Algorithm 1 *computeRelevanceScore*

Data: Prefix σ, Masked prefix σ_m, aggregated attention scores per activity ψ_σ and ψ_{σ_m}, prediction vectors p_σ and p_{σ_m}, most likely next activities $\mathcal{P}_r$
Result: Score matrix
$K \leftarrow (0)_{ij} \forall i, j \in \{1, ..., |\mathcal{A}|\}$
foreach $a \in \mathcal{P}_r$ **do**
  foreach *masked activity* $a_m \in \sigma_m$ **do**
    $s \leftarrow p_\sigma(a) \cdot \psi_\sigma(a_m)$
    if $p_\sigma(a) \sim p_{\sigma_m}(a)$ **then**
      $s \leftarrow -s$
    end
    $K(a, a_m) \leftarrow K_\sigma(a, a_m) + s$
  end
  foreach *non-masked activity* $a_n \in \sigma_m$ **do**
    if $p_\sigma(a) \sim p_{\sigma_m}(a)$ **then**
      $s \leftarrow \psi_{\sigma_m}(a_n) \cdot p_\sigma(a)$
    else
      $s \leftarrow |\psi_\sigma(a_n) - \psi_{\sigma_m}(a_n)| \cdot |p_\sigma(a) - p_{\sigma_m}(a)|$
    end
    $K(a, a_m) \leftarrow K_\sigma(a, a_m) + s$
  end
end
return K

Constructing the explanation graph To construct the explanation graph, we first aggregate the score matrices obtained for each prefix across both masking scenarios:

$$K^{few} = \sum_{\sigma \in L} K_{\sigma}^{few} \qquad K^{most} = \sum_{\sigma \in L} K_{\sigma}^{most}.$$

Next, we normalize each row in K^{few} and K^{most} to ensure that the sum of each row is one, thereby facilitating comparability across different activities. Afterwards, we convert each entry in both matrices to a boolean value, by comparing it to a predefined threshold δ_{edge} (set to 0.1 by default). Entries exceeding this threshold are marked as "true", indicating significant relevance between activities, while entries below the threshold are marked as "false", indicating no relevant relationship.

To integrate insights from both masking scenarios, we apply a component-wise logical OR operation between K^{few} and K^{most}. The resulting matrix K serves as an adjacency matrix, where a "true" value indicates a directed edge from the activity in the column to the activity in the row. In contrast, a "false" value indicates no edge. Finally, the global graph is constructed based on this adjacency matrix.

Evaluation

In this section, we evaluate our explainer from multiple angles. We begin by describing the implementation of our approach as a software tool and illustrating its practical application (see Implementation section). To evaluate the approach itself, i.e., both explainers, we employ quantitative and qualitative methods. Notably, qualitative evaluation in the context of XAI tends to introduce subjective biases, as they often favor explanations that appear intuitively plausible (Nauta et al. 2023). Hence, we prioritize a quantitative evaluation (see Quantitative evaluation section) relying on established objective metrics, followed by a concise qualitative assessment in Qualitative evaluation section. In the qualitative analysis, we interpret the generated explanations exemplarily for a synthetic event log, comparing them against the process model used to generate this log.

Implementation

In this section, we present the implementation of our proposed explainers as a software tool. The tool is developed as a Python-based application that supports both command-line usage and interaction through a graphical user interface (GUI). The GUI facilitates a more intuitive experience for process analysts and machine learning experts, allowing them to train models and apply our explainers seamlessly.

The software supports the entire workflow from pre-processing of event logs to training a transformer model and applying the explainers to the trained model. At each stage, users can customize various settings, which we briefly outline below:

- **Pre-processing stage:** Users can configure the splitting ratio for training and test datasets, as well as choose between various splitting methods (either random or time-based). If random splitting is selected, users can specify how many times the procedure should be repeated.

- **Training stage:** In this stage, users can adjust the hyperparameters of the transformer model (e.g., number of training epochs, batch size). By default, the parameters recommended in Bukhsh et al. (2021) are pre-set, ensuring a proper starting configuration. Users also have the option to specify whether the training should be conducted on GPU or CPU. While GPU training speeds up the training process, CPU training may offer more control and reproducibility through deterministic behavior.
- **Explanation stage:** Once the model is trained, users can select between the two explainers. Input fields are available to configure parameters specific to the explainers such as prediction thresholds and attention score thresholds. Users can also customize the masking scenarios according to their specific requirements. After running the explainers, the tool displays the explanation graphs and metrics.

Each step of the workflow generates artifacts (such as event logs, trained models, explanation graphs, and metrics), which are saved along with their respective configurations. This saving ensures transparency and reproducibility by enabling users to track and replay their experiment settings and results.

For more technically advanced users, all functionalities provided by the GUI can also be accessed via the command-line interface. This is particularly useful for automating large-scale experiments. The source code of our application as well as all results of our experiments is available in the GitHub repository associated with this paper[4].

Quantitative evaluation

In our experimental setup, we utilize all event logs from the pre-study. All models are trained in the same way as the baseline models in the pre-study, using a random split of 70% for training and 30% for testing as well as the default hyperparameters (see Appendix 1) proposed by Bukhsh et al. (2021). Additionally, test samples are extracted as in the pre-study, i.e., all prefixes with a minimum length of one.

Due to the absence of other transformer-specific global explainers, a fair comparison with other explanation approaches is not feasible. However, direct comparisons with existing local or model-agnostic approaches would be less meaningful, since local and global explainers pursue fundamentally different objectives, whereas local explainability is a much simpler task. Similarly, model-agnostic approaches are at a disadvantage as they cannot provide the same level of detail, as model-specific approaches utilizing model internals.

A general challenge in evaluating explanation approaches is that correct but implausible explanations for poorly performing black box models may be disregarded. This emphasizes the difficulty to distinguish between correct explanations for underperforming models from incorrect explanations for capable models (Nauta et al. 2023). In the case of our approach, an additional challenge arises: our explanation approaches are inherently tied to the attention mechanism of the model, requiring properly trained attention weights. We address these challenges by first evaluating the predictive performance of the trained models to verify that it is properly trained. This approach

[4] https://github.com/mkaep/transformer-explainability

minimizes both the risk that our explainers are unfairly penalized due to errors inherent to the models and mitigates the probability of generating meaningless explanations if the attention weights are poorly trained.

To evaluate the predictive performance of the transformer models, we utilize commonly used metrics: weighted F1-score, weighted precision, and accuracy (Bukhsh et al. 2021; Camargo et al. 2019; Pasquadibisceglie et al. 2019). Our models achieve a mean weighted F1-score of 0.74 across all real-world event logs (with minimum and maximum values of 0.46 and 0.93, respectively) and a mean weighted F1-score of 0.76 across synthetic event logs (minimum 0.51 and maximum 1.00). All values can be found in Table 2. Hence, these scores indicate that there are no no-skill models, suggesting that the attention weights have been properly trained. Achieving significantly higher values is prevented because the transformer solely focuses on the activity event attribute. In consequence, many cases remain ambiguous and are not decidable for the model.

For our purposes a certain number of incorrect predictions are unproblematic, as our primary purpose is to elucidate the model's decision-making rationale. Beside explaining correct predictions we also want explanations for wrong predictions. However, it is an essential and pivotal assumption that the model and in particular the attention weights are properly trained. Because our evaluation of the predictive performance shows that there are no no-skill models, we can assume that this assumption holds. This claim is also supported by the results of the pre-study, which indicate that these models have reliable attention scores.

Evaluation Metrics For our quantitative evaluation, we adopt the metrics introduced in Nauta et al. (2023). These metrics are based on explanations in the form of a set of implications rules. This necessitates us to transform the global graph $G = (V, E)$ into rule sets, with rules of the form $X \rightarrow Y$, with $X, Y \subseteq V$. Therefore, each node $v \in V$ in the graph is transformed into a rule that links it with its direct successors S_v. Thus, for each vertex $v \in V$, we obtain a rule of the form $v \rightarrow S_v$. These rules can be interpreted in such a way that the right-hand side encompasses all possible predictions if the currently executed activity is v.

Table 2 Predictive performance of the trained models

Event log	Weighted precision	Accuracy	Weighted F1-score
BPIC12	0.80	0.83	0.80
BPIC12_O	0.96	0.93	0.93
BPIC12_W	0.88	0.87	0.87
BPIC12_WC	0.75	0.78	0.75
BPIC13_CP	0.69	0.83	0.76
BPIC13-I	0.50	0.69	0.57
Helpdesk	0.74	0.79	0.76
Sepsis	0.49	0.51	0.46
Complex Model	0.52	0.57	0.51
LDD	0.88	0.92	0.89
Looped And	0.76	0.65	0.64
Sequence	1.00	1.00	1.00

Below, we provide a brief overview of the intent behind the metrics and how they can be computed. For further details we would like to refer to Nauta et al. (2023):

- **Correctness:** This metric evaluates the truthfulness of an explanation (Nauta et al. 2023). In the literature, the correctness is measured in two different ways: *(i)* by comparing the explanations to ground-truth explanations, or *(ii)* by calculating the correlation between the explainer's feature importance and the model's feature importance. Because ground-truth explanations are not available in our setting, we rely on the second option. The calculation of the feature importance of explainer and model is identical to those used in the first experiment of our pre-study. Thus, this metric ranges from $[-1, 1]$, with higher values indicating that the explainer primarily uses the correct feature for its explanations.
- **Completeness:** This metric measures how closely the explanations coincide with the predictions of the black box model (Nauta et al. 2023). To calculate completeness, we treat the most likely next activities $\mathcal{P}_r$ of the transformer as ground truth and the explanations (to be concise the right-hand sides) as predictions to be evaluated. Hence, we evaluate the explainer analogously to a machine learning model by comparing its predictions with ground truth data. Because the predictions are represented by one or more activities this results in a multi-label confusion matrix. From this matrix, we derive metrics such as precision, recall, and F1-score. Higher values indicate better completeness, meaning that the output of the explainer matches the predictions of the transformer.
- **Consistency:** The consistency metric judges the determinism of an explainer, i.e., whether an explainer provides the same explanations for two distinct models that yield similar outputs. To compute this metric, we first train an initial model M_1 and then five additional models. From these, we select the model with the greatest difference from M_1 based on the mean difference between their trainable weights. In the following, we denote this model with M_2.
  For each test sample σ, we calculate a consistency score c_σ if the predictions of M_1 and M_2 for σ are similar (evaluated using the cosine similarity) and if their feature importances align. The consistency score c_σ is computed as follows:

$$c_\sigma = 1.0 - |1 - cosineSim(p_1(\sigma), p_2(\sigma)) - Jacard(R_1, R_2)|, \tag{1}$$

  where R_1 and R_2 are the rule sets generated by the explainer for M_1 and M_2, respectively, and the Jaccard coefficient measures the similarity between these rule sets. The intention behind this formula is that the difference between the similarity of the prediction vectors (i.e., $|1 - cosineSim(p_1(\sigma), p_2(\sigma))$) and the Jaccard coefficient should be close to zero. Or in other words: If the predictions are not similar, the Jaccard coefficient should not be either and vice versa. The consistency scores is averaged across all test samples to lie in the interval [0, 1]. A value of 1 indicates optimal consistency, i.e., that the explainer provides identical explanations for models with the same outputs.
- **Continuity:** The continuity metric evaluates whether an explainer provides similar explanations for slightly altered inputs (Nauta et al. 2023). The intent behind this metric is, that if a model provides stable, similar explanations for slightly modified inputs, this suggests that the explainer could also provide meaningful and

consistent explanations for similar, previously unseen data points (Nauta et al. 2023). Thus, it can be used as an indicator of the robustness of explanations. By varying the input, we examine the extent to which the new explanation deviates (Nauta et al. 2023). Therefore, we create for each test sample σ a collection of modified versions of σ by applying masking and replacing as in the pre-study. For each modified sample σ_m, we compare the corresponding prediction vector p_{σ_m} with the prediction vector p_σ obtained for the original sample σ. If these vectors are similar (i.e., $cosineSim(p_{\sigma_m}, p_\sigma > \delta_{pred})$, we calculate a continuity score using Equation 1. Here, the rule sets R_1 and R_2, represent the explanations generated for σ and σ_m, respectively. The final continuity score is obtained by averaging the continuity scores across all modified samples. Similar to the consistency metric, this score takes values in the interval [0, 1], whereas 1 indicates perfect continuity (Nauta et al. 2023).

- **Contrastivity:** The contrastivity metric serves as the counterpart to the continuity metric. While the continuity metric focuses on modifications that result in similar predictions, contrastivity examines modifications that produce different predictions. Therefore, we only consider modified versions σ_m of σ, if $cosineSim(p_{\sigma_m}, p_\sigma) < \delta_{pred}$ (i.e., we invert the condition compared to the continuity metric). Apart from this modification, the metric is calculated in the same way as the continuity metric. Thus, it evaluates how distinct the explanations of an explainer are for dissimilar inputs (Nauta et al. 2023). This metric also ranges from [0, 1], with a value of 1 indicating perfect contrastivity.
- **Compactness:** The compactness metric measures how concise an explanation is. Therefore, two different aspects can be measured: the number of rules generated by an explainer and the average length of the rules' right-hand sides. Due to the limitations of human cognitive capacity, shorter rules are preferable, because they are easier to understand. Consequently, this metric is particularly relevant in practical applications. However, it is important to note that this metric favors brevity, which does not necessarily correlate with the quality of the explanations. An explanation with fewer or shorter rules may lack essential information and, thus, may not always be the most comprehensible or accurate. For this reason, compactness should only be evaluated in conjunction with other metrics.

To apply the metrics, a fixed percentage (80%) of prefixes is randomly selected from each event log to accommodate their varying sizes. Metrics are then computed for each prefix individually, followed by averaging these values. Table 3 reports these average values along with their standard deviations. In the following, we abbreviate Backward Explainer as BE and Attention Exploration Explainer as AE.

Discussion of the results We consistently observe high correctness values, indicating a clear correlation between high attention scores and the activities decisive for prediction. This observation holds for both real-world and synthetic event logs. Only in the cases of BPIC12_O and the synthetic Looped And and Sequence event logs do the values drop slightly. Since the correlation values, with exception of these logs, exceeds 0.5, we observe a moderate to strong correlation overall. By comparing AE and BE, we find that

the explainers generally behave similarly. However, in the cases of Sepsis, Helpdesk, and Sequence, BE achieves significantly higher values.

A more varied pattern emerges for the completeness metric when calculated using the F1-score. Depending on the event log, either relatively poor values (BPIC12, Complex Model, LDD, Sequence, Sepsis) or quite good values (BPIC13) are achieved. Further analysis shows that low F1-completeness is primarily due to low recall values (see row Completeness (Recall) in Table 3). Although the explanations are often correct (as indicated by significantly higher completeness precision), not all explanations are captured. In the resulting graphs (or rules), this means most of the edges are correct (high precision), but some edges are missing (low recall).

However, there are notable differences between the two explainers. The AE explainer consistently outperforms the BE in precision completeness. A similar trend is seen for recall completeness, where AE performs nearly identical to BE for BPIC12, BPIC12_0, BPIC12_WC, and Complex Model but outperforms BE in BPIC12_W, BPIC13_CP, BPIC13_I, Sepsis, and Looped And. Only for Helpdesk, Sequence, and LDD are small differences (3 to 5%) observed in favor of BE. These findings suggest that AE's explanations align more closely with the predictions of the model. These observations will also be visible in the qualitative evaluation in Qualitative evaluation section.

For the consistency metric, we observe extreme differences both between the event logs and between the explainers. While some event logs show nearly perfect consistency for each explainer (AE on BPIC13_CP and BE on Looped And), others exhibit very poor consistency close to zero (BPIC13_I for the BE and Looped And for AE). The extreme diametral behavior of BE and AE in the case of the Looped And is astonishing. An explanation for this behavior is that the attention scores of the two models selected for the calculation of the consistency metric are very different. Nevertheless, on some event logs the models come to the same prediction, since other parts of the architecture significant contribute to the prediction. However, the relevant activities ($\mathcal{A}_r$) derived from the attention scores differ fundamentally, which affects the structure of the global graph and thus the performance of the explainers. Since it is not a general pattern, it is a strong indicator that there are dataset specific reasons for that. Aside from these outliers, both explainers typically show mediocre consistency scores around 0.5 across most event logs.

For the continuity metric, the AE explainer nearly always achieves high values, typically above 0.85. Only for the synthetic Sequence and LDD event logs do we observe lower values (0.57 and 0.78, respectively). In contrast, the BE explainer performs worse, showing mostly lower values for BPIC12 (0.71), BPIC12_W (0.81), BPIC13_CP (0.49), BPIC13_I (0.19), Sepsis (0.57), Helpdesk (0.73), Complex Model (0.72), LDD (0.57), and Sequence (0.78). Overall, we can conclude that AE is a highly robust explainer, providing meaningful explanations even for previously unseen data. Although the BE cannot compete these values most of the time, it still achieves relatively high continuity scores around 0.7. When evaluating the counterpart, i.e., the contrastivity metric, we find consistently low values across all event logs, with both explainers performing similarly. The combination of high continuity values and relative low contrastivity scores suggests that the changes in the input have hardly any effect on the prediction.

Table 3 Results for Attention Exploration Explainer (AE) and Backward Explainer (BE); cells: mean (first), std. deviation (second); boldfaced = best values; *NaN* = no value could be computed (in case of std there was sometimes no variance due to identical results, in case of the correctness value it occurs if rule sets for a prefix and its modifications are identical)

Metric	Explainer	BPIC12	BPIC12_O	BPIC12_W	BPIC12_WC	BPIC13_CP	BPIC13_I	Sepsis	Helpdesk	Complex Model	LDD	Looped And	Sequence
Correctness	AE	**0.65±0.15**	0.48±0.22	0.65±0.21	0.56±0.18	0.60±0.28	**0.57±0.25**	0.59±0.17	0.73±0.18	0.51±0.12	0.68±0.22	0.31±0.36	0.17±0.25
	BE	0.63±0.20	0.48±0.22	0.65±0.21	**0.62±0.18**	0.75±0.18	NaN±NaN	**0.86±0.13**	**0.91±0.08**	0.59±0.15	0.68±0.21	NaN±NaN	**0.32±0.18**
Completeness (Precision)	AE	0.18±0.08	**0.25±0.07**	**0.49±0.14**	**0.68±0.35**	**0.87±0.14**	**0.82±0.06**	**0.54±0.13**	0.50±0.15	**0.30±0.07**	0.10±0.00	**0.31±0.01**	0.11±Nan
	BE	**0.20±0.09**	0.11±0.03	0.21±0.23	0.42±0.25	0.74±0.23	0.35±0.05	0.50±0.19	**0.54±0.15**	0.19±0.03	0.30±0.00	0.00±0.00	**0.32±NaN**
Completeness (Recall)	AE	**0.05±0.00**	**0.20±0.01**	**0.22±0.03**	0.23±0.06	**0.79±0.15**	**0.77±0.08**	**0.11±0.01**	0.25±0.03	0.10±0.00	0.14±0.00	**0.25±0.01**	0.11± NaN
	BE	0.04±0.01	0.18±0.03	0.09±0.05	**0.25±0.06**	0.46±0.11	0.26±0.04	0.04±0.02	**0.30±0.04**	**0.11±0.01**	**0.17±0.00**	0.05±0.02	**0.14±NaN**
Completeness (F1-score)	AE	**0.07±0.01**	**0.21±0.02**	**0.27±0.04**	0.28±0.09	**0.79±0.15**	**0.77±0.08**	**0.15±0.02**	0.30±0.04	**0.12±0.01**	0.12±0.00	**0.27±0.01**	0.10± NaN
	BE	0.06±0.01	0.13±0.02	0.10±0.08	**0.29±0.10**	0.50±0.14	0.29±0.04	0.04±0.02	**0.36±0.05**	0.10±0.01	**0.18±0.00**	0.01±0.00	**0.15±NaN**
Consistency	AE	0.19±0.06	0.31±0.05	**0.73±0.08**	**0.54±0.07**	**0.99±0.09**	**0.18±0.03**	0.10±0.02	0.09±0.05	**0.25±0.06**	0.27±0.15	0.01±0.01	0.42±0.21
	BE	**0.46±0.07**	**0.65±0.07**	0.66±0.13	0.39±0.08	0.59±0.18	0.01±0.02	**0.13±0.03**	**0.11±0.09**	0.02±0.03	**0.45±0.22**	**0.99±0.01**	**0.53±0.20**
Continuity	AE	**0.93±0.05**	0.95±0.07	**0.92±0.06**	0.89±0.05	**0.98±0.10**	**0.99±0.02**	**0.92±0.13**	**0.85±0.19**	**0.88±0.21**	0.57±0.31	0.93±0.24	0.78±0.25
	BE	0.71±0.05	0.95±0.07	0.81±0.13	**0.90±0.06**	0.49±0.13	0.19±0.24	0.57±0.09	0.73±0.15	0.72±0.19	0.57±0.32	0.93±0.24	0.78±0.25
Contrastivity	AE	0.26±0.25	0.07±0.18	0.16±0.26	0.02±0.09	0.05±0.19	0.03±0.12	0.13±0.27	**0.20±0.30**	0.30±0.15	0.42±0.31	0.40±0.21	0.32±0.34
	BE	**0.45±0.33**	0.07±0.18	**0.19±0.30**	**0.03±0.13**	0.05±0.20	**0.04±0.18**	**0.17±0.33**	0.08±0.22	**0.58±0.17**	0.42±0.31	0.40±0.21	**0.33±0.30**
Compactness (#Rules)	AE	**6.32±1.11**	4.61±0.55	4.05±0.78	**3.19±0.60**	**1.88±0.34**	1.99±0.09	**4.55±0.85**	**2.45±0.79**	**4.19±0.98**	**3.75±1.91**	**2.00±0.00**	**5.08±2.02**
	BE	13.04±2.03	4.61±0.55	**3.28±0.57**	3.97±0.89	2.16±0.53	**1.38±0.49**	7.72±1.42	3.32±0.94	7.47±2.03	3.94±2.17	4.00±0.00	7.08±3.34
Compactness (Mean length)	AE	4.12±0.77	**2.21±0.09**	1.86±0.25	1.89±0.15	2.09±0.35	2.00±0.09	2.76±0.42	2.55±0.62	3.60±0.64	**1.89±1.13**	4.00±0.00	2.06±0.40
	BE	**1.66±0.20**	2.56±0.06	**0.90±0.14**	**1.48±0.09**	**0.60±0.20**	**0.19±0.24**	**0.83±0.16**	**1.99±0.39**	**2.01±0.53**	2.76±0.74	**1.00±0.00**	**1.96±0.65**

This is essentially due to the fact that the transformer primarily learns the most frequent execution paths, as it is limited to the event activity attribute. As a result, it tends to provide "standard" predictions even in cases of significant modifications in the input prefix. This observation is also in line with prior experiments conducted in Käppel et al. (2021), which indicate that prediction models usually focus on frequent execution variants and treat exceptional cases as standard variants.

With regard to the compactness metric, we observe that the BE explainer tends to generate more rules, although they usually remain within a range that does not hinder interpretability (fewer than five rules). Exceptions can be noted for BPIC12 (13.04), Sepsis (7.72), Complex Model (7.47), and Sequence (7.08), where BE produces more rules compared to AE. When examining the mean length of the right-hand sides of the rules, we observe that the AE explainer consistently produces longer rules. However, these lengths remain below 5, which still enables human interpretability. In contrast, the Backward Explainer explainer typically generates rules with mean lengths around 1. When combined with its performance in other metrics, particularly completeness and continuity, we see this as an indicator that BE's rules are potentially too short to adequately capture the process behavior.

In summary, the performance of both explainers is nearly identical when comparing real-world and synthetic event logs. However, notable differences exist between the two explainers. Based on the most relevant metrics for explainer quality – correct and completeness – we consider the AE explainer to be the superior choice.

Qualitative evaluation

To provide a clearer and intuitive understanding of the explainers' performance and weaknesses, we discuss exemplarily their output for the synthetic Complex Model event log (cf. Fig. 12). Because it is the most complicated process model we used for the generation of the synthetic event logs it offers the best insights into the effectiveness of the explainers. For creating the explanation graphs, we used all prefixes from the test log and start with a minimum prefix length of one. Including all prefixes is essential to capture the full process behavior, as this allows us to trace the step-by-step development of a process instance.

For better comparability, the nodes in both explanation graphs were manually aligned similarly to the activities in the process model. At first glance, the BE explanation graph shows significantly more edges than the AE graph, reflecting the results of the compactness metric. Although there are significantly more edges in the graph of the BE, the completeness recall values of both explainers are nearly identical (0.10 vs. 0.11). However, the AE outperforms the BE with regard to completeness precision (0.30 vs. 0.19). Hence, this example confirms the finding of the quantitative evaluation that the AE finds fewer incorrect edges. Nevertheless, both graphs contain erroneous transitions. For example, the AE graph contains a reflexive edge at the node for activity A and an unwanted shortcut from A to J. The latter is also contained in the BE graph.

Nonetheless, similarities to the process model are evident in both explanation graphs. The initial (A-B-C) and the subsequent AND gateway after activity D are clearly recognizable. While both explainers correctly identify that activities H, E, and P can follow D, the AE graph does not cover all possible follow-up sequences, whereas the BE allows

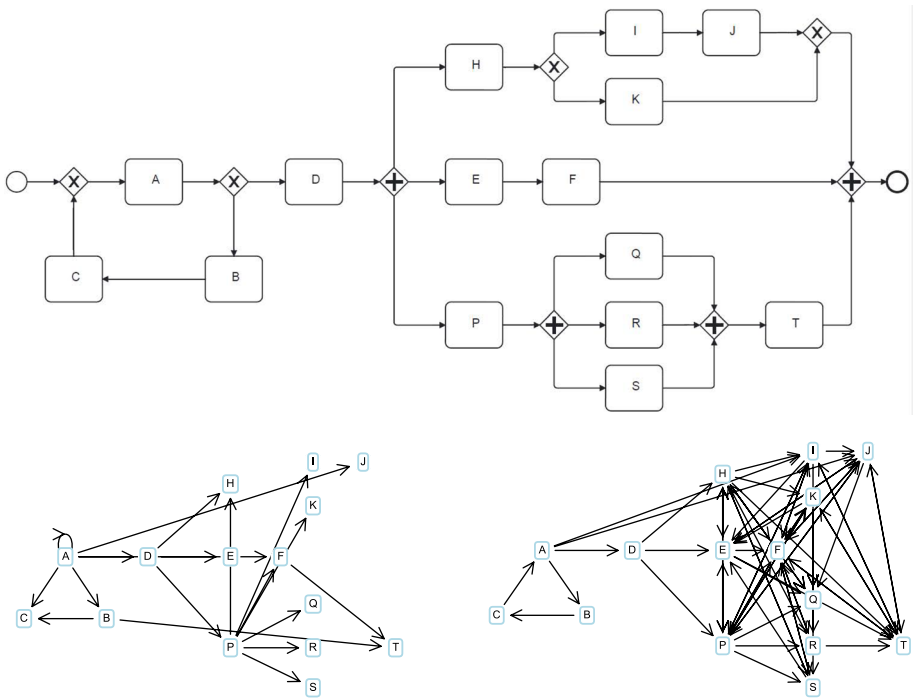

Fig. 12 Resulting explanation graphs for Complex Model Event Log. Top: The underlying real process model. Left: Attention Explorer generated graph. Right: Backward Explainer generated graph

more flexible process executions. A similar pattern occurs at the second AND gateway (after activity B), where the AE graph identifies various options but ends in dead ends, while the BE graph identifies significantly more transitions.

Concluding remarks

Based on the results of the quantitative and qualitative evaluation, we can answer our research question as follows: the transformer's attention scores are effective for checking and visualizing its process understanding. Although the transformer model produces some incorrect predictions – primarily because it relies solely on the activity event attribute for its predictions – certain patterns in the control flow, such as loops or XOR-gateways, cannot always be predicted correctly. Nevertheless, we were able to observe these patterns in the explanation graph. The reason for that is, that the transformer considers the different options related with these patterns, paying them a certain probability, although he cannot predict the correct one due to its limited information.

In the rest of this section, we conclude the work with some theoretical and practical implications. Additionally, we discuss potential limitations of our approach.

Theoretical and Practical Implications Our approach offers valuable opportunities for advancing transformer-based predictions of the next activity. In the following, we outline the primary implications of our work for both theory and practice in the field of process mining and especially for the subfield of next activity prediction.

Our work has three main theoretical implications. First, our study enhances the theoretical understanding of how transformer models capture and interpret process structures, highlighting the relationship between model internals (i.e., attention scores) and recorded process behavior in form of traces. From a more general point of view, it delivers insights into how transformer models encapsulate temporal and sequential dependencies in prediction tasks. Second, the global explanatory capabilities of the proposed explainers contribute to assessing the overall process understanding of the prediction model and ensure consistency in explanations. Unlike many local explainers, which may yield contradictory explanations across different process instances, our approach provides a holistic view of the process logic, fostering coherence across explanations. Third, our approach has the potential to lay the foundation for a new class of process discovery techniques, which extract process workflows based on predictive logic. As a consequence, a prediction model would become an active component in process analysis and (re)design.

From a practical standpoint, our work has two main implications. First, our approach enhances transparency and trust in prediction models. By making the decision-making process of the model more transparent and interpretable, process participants gain a clearer understanding of the predictions and have more confidence in the model's predictions. This is particularly important for processes, where the predictions directly impact business operations or strategic decisions as well as for processes in critical contexts, where a prediction may have drastic consequences (e.g., healthcare systems). Second, the insights gained through the explainers can serve as a promising starting point for model improvement and debugging. Especially in cases of erroneous predictions, the explainers provide useful insights that can guide the model improvement, either through retraining or by providing additional training data to address situations where the model currently struggles. Moreover, by analyzing the outputted graph structure, practitioners can identify potential weaknesses or areas for refinement. This also allows to some degree to anticipate potential future behavior of the prediction model, even for situations that were not observed until now.

Limitations There are some limitations in our work. First, our experiments in both pre-study and evaluation are limited to eight real-world event logs and four synthetically generated ones. Although the real-world event logs stem from diverse domains and have heterogeneous characteristics, it is possible that the obtained results would vary with different event logs. Nonetheless, our findings provide encouraging evidence that our explanation approach can handle common data quality issues in real-world logs, such as noise or inconsistencies.

In our explanation approaches, we aggregate the attention scores across all four attention heads, offering a holistic perspective on the entire attention mechanism but without distinguishing the individual contributions of individual heads. Thus, it remains unexplored what particular heads learn or how their interpretations might differ. However, as the objective of our study was to explore, whether attention scores as a whole give insights into what the prediction model has learned, we considered it appropriate to first focus on the aggregated attention mechanism. While aggregating may lead to losing fine-grained information, this choice primarily affects the detail level of the explanation rather than the overall usefulness of the explainers.

Another limitation is that our explanation approach relies solely on attention scores as model internals. Although these scores play a crucial role in the architecture, as verified in the pre-study, other components and layers of the transformer architecture also influence the prediction. Currently, the impact of these components is widely unexamined. However, this flaw is mitigated by the fact that the subsequent layers primarily process the attention scores further, so that they can be still considered as the central element in the model's decision-making process.

Our explainers employ various thresholds, such as those determining the prediction similarity or the relevance of an attention score. These thresholds affect the explanation quality because they control the addition or removal of edges in the explanation graph. For the attention score thresholds, this issue is somewhat alleviated, as the pre-study reveals that attention scores are typically either very high or very low, allowing setting a threshold with minimal information loss. However, configuring the prediction thresholds is more challenging, as the ideal threshold can vary depending on the particular process and its execution state. In our study, we chose a universal threshold expected to perform reasonably well across different processes. However, the threshold would be ideally adapted to the specific process characteristics. For example, a lower threshold is suitable for flexible processes with numerous decision points and alternative execution paths, while a higher threshold is appropriate for more sequential processes. Given that in practice, processes are often a mixture of flexible and largely sequential parts, a dynamic adjustment of thresholds depending on the current execution state would provide the best result.

Conclusion and future work

This paper investigates whether a trained transformer model for next activity prediction, based on the architecture proposed in Bukhsh et al. (2021), has gained an understanding of the control-flow of the underlying process. Given that attention scores are at the heart of the transformer architecture and a key factor for its effectiveness, we thoroughly investigated their reliability for explanation purposes. Our experiments provide strong evidence that attention scores offer valuable insights into the decision-making of the prediction model, effectively supporting the assessment and visualization of its process understanding. To this end, we developed two transformer-specific global explainable artificial intelligence approaches based entirely on attention scores, which create graph structures resembling process models. These graphs illustrate how attention scores for particular activities are linked to predictions. We evaluated both explainers using established quantitative metrics, revealing that attention score based explainers hold substantial potential. However, because the prediction models only incorporate control-flow information, the process cannot be fully learned. In future research, we plan to improve the explainers by analyzing the distinct relationships learned by individual attention heads more in detail. Moreover, we aim to replace the current masking and modification operations for determining the relevance of events with more sophisticated techniques tailored to the specific needs of process data. Additionally, we plan to extend our explanation approach to transformer architectures that incorporate additional features, such as resource or time information.

Appendix 1: Hyperparameter configuration

The following table list the hyperparameter configuration used for the training of the transformer models.

Hyperparameter	Value
Epochs	10
Batch size	12
Learning rate	0.001
GPU usage	True
Number of heads	4
Embedding Dimension	36
Dimension of the feed forward network	64

Appendix 2: Process models

In the following the process models used for generating the synthetic event logs are depicted:

Complex Model

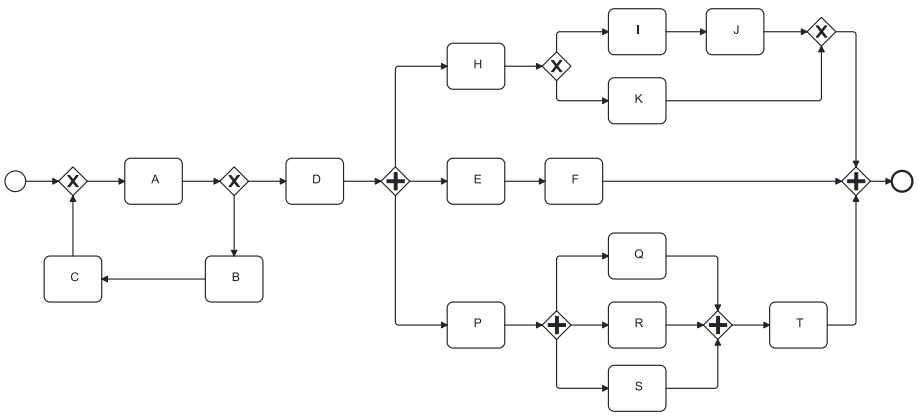

Fig. 13 BPMN model for the Complex Model event log

LDD

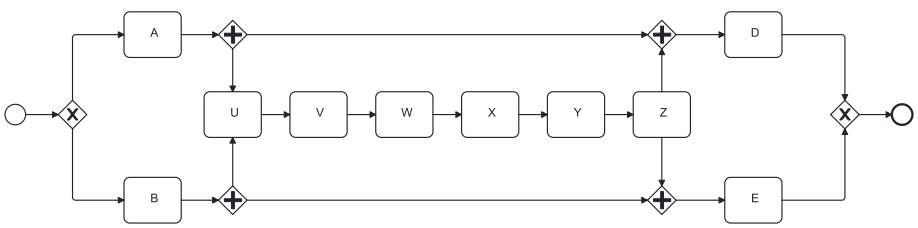

Fig. 14 BPMN model for the Long Distance Dependency event log

Looped And

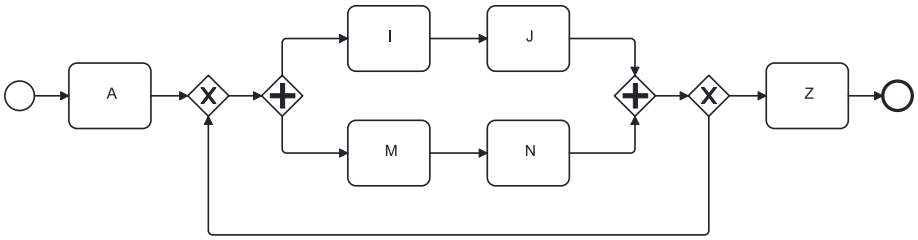

Fig. 15 BPMN model for the Looped And event log

Sequence

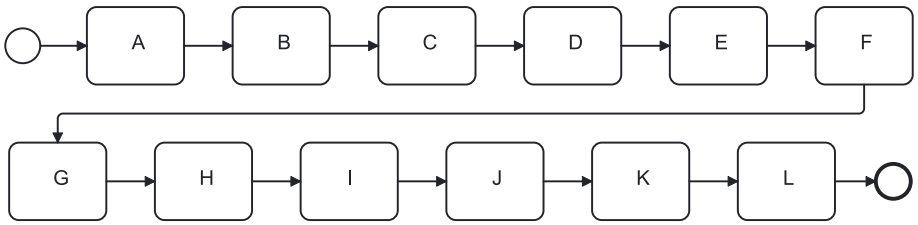

Fig. 16 BPMN model for the Sequence event log

Authors' contributions
All authors contributed to the conception and the writing of the prior BPM paper. For this extended version M.K., L.A., and S.J. wrote the main manuscript text. M.K., L.A., and S.H. prepared the visualizations. M.K. and S.H. contributed to implementation of the approach's prototype. M.K., L.A., and S.H. conducted experiments (including those of the prior BPM paper). M.K., L.A, and S.J. reviewed the manuscript.

Funding
No funding was received.

Data availability
No datasets were generated or analysed during the current study.

Materials availability
Our code and detailed results can be found in the supplementary material at https://github.com/mkaep/transformer-explainability.

Declarations

Ethics approval and consent to paticipate
Not applicable

Competing interests
The authors declare no competing interests.

Received: 9 November 2024 Accepted: 12 May 2025
Published: 4 June 2025

References

Akoğlu H (2018) User's guide to correlation coefficients. Turk J Emerg Med 18:91–93

Bukhsh ZA, Saeed A, Dijkman RM (2021) Processtransformer: Predictive business process monitoring with transformer network. arXiv. https://doi.org/10.48550/arXiv.2104.00721

Burattin A, Re B, Rossi L et al (2022) Purple: a purpose-guided log generator. In: Proceedings of the ICPM Doctoral Consortium and Demo Track 2022. CEUR-WS, CEUR Workshop Proceedings. iCPM 2022 Demo Track; Conference date: 23-10-2022 Through 28-10-2022. https://icpmconference.org/2022/. Access Dates: 17.05.2025

Camargo M, Dumas M, González-Rojas O (2019) Learning accurate lstm models of business processes. In: Proc. of BPM Conf. Springer, Cham, pp 286–302

Carvalho DV, Pereira EM, Cardoso JS (2019) Machine Learning Interpretability: A Survey on Methods and Metrics. Electronics 8(8):832. https://doi.org/10.3390/electronics8080832

Devlin J, Chang MW, Lee K et al (2019) BERT: Pre-training of deep bidirectional transformers for language understanding. In: Burstein J, Doran C, Solorio T (eds) Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers). Association for Computational Linguistics, Minneapolis, pp 4171–4186. https://doi.org/10.18653/v1/N19-1423. https://aclanthology.org/N19-1423/

Di Francescomarino C, Ghidini C (2022) Predictive Process Monitoring. Springer, Cham, pp 320–346

Dumas M, Rosa ML, Mendling J et al (2018) Fundamentals of Business Process Management, 2nd edn. Springer Publishing Company, Incorporated, Berlin.

Elkhawaga G, Abu-Elkheir M, Reichert M (2022) Explainability of Predictive Process Monitoring Results: Can You See My Data Issues? Appl Sci 12:8192. https://doi.org/10.3390/app12168192

Elkhawaga G, Abu-Elkheir M, Reichert M (2022) Explainability of predictive process monitoring results: Can you see my data issues? Appl Sci 12(16)

Evermann J, Rehse JR, Fettke P (2017) Predicting process behaviour using deep learning. Decis Support Syst 100:129–140

Grigori D, Casati F, Castellanos M et al (2004) Business process intelligence. Comput Ind 53(3):321–343

Guidotti, R., Monreale, A., Ruggieri, S., Pedreschi, D., Turini, F. and Giannotti, F. (2018) Local Rule-Based Explanations of Black Box Decision Systems. arXiv Preprint arXiv: 1805.10820. https://doi.org/10.48550/arXiv.1805.10820

Hanga KM, Kovalchuk Y, Gaber MM (2020) A graph-based approach to interpreting recurrent neural networks in process mining. IEEE Access 8:172923–172938

Huang TH, Metzger A, Pohl K (2022) Counterfactual explanations for predictive business process monitoring. In: Information Systems. Springer, Cham

Jain S, Wallace BC (2019) Attention is not Explanation. In: Proc. of 2019 Conf. of the North American Chapter of the ACL). ACL, Minneapolis

Käppel M, Ackermann L, Jablonski S et al (2024) Attention please: What transformer models really learn for process prediction. In: Marrella A, Resinas M, Jans M et al (eds) Business Process Management. Springer Nature Switzerland, Cham, pp 203–220

Käppel, M., Jablonski, S., Schönig, S. (2021). Evaluating Predictive Business Process Monitoring Approaches on Small Event Logs. In: Paiva, A.C.R., Cavalli, A.R., Ventura Martins, P., Pérez-Castillo, R. (eds) Quality of Information and Communications Technology. QUATIC 2021. Communications in Computer and Information Science, Springer, Cham, vol 1439:167–182. https://doi.org/10.1007/978-3-030-85347-1_13

Kendall MG (1938) A new measure of rank correlation. Biometrika 30(1/2):81–93

Klinkmüller C, van Beest NRTP, Weber I (2018) Towards reliable predictive process monitoring. In: Information Systems in the Big Data Era. Springer, Cham 317:163–181. https://doi.org/10.1007/978-3-319-92901-9_15

Kratsch W, Manderscheid J, Röglinger M et al (2021) Machine learning in business process monitoring: A comparison of deep learning and classical approaches used for outcome prediction. BISE 63(3):261–276

Kullback S (1952) An application of information theory to multivariate analysis. Ann Math Stat 23(1):88–102

Maggi FM, Di Francescomarino C, Dumas M et al (2014) Predictive monitoring of business processes. In: Proceedings of the 26th International Conference on Advanced Information Systems Engineering. Springer, Cham, vol 8484:pp 457–472. https://doi.org/10.1007/978-3-319-07881-6_31

Manning C, Raghavan P, Schütze H (2008) Introduction to Information Retrieval. Cambridge University Press, Cambridge.

Márquez-Chamorro AE, Resinas M, Ruiz-Cortés A (2017) Predictive monitoring of business processes: A survey. IEEE Trans Serv Comput 11(6):962–977

Mehdiyev N, Evermann J, Fettke P (2020) A novel business process prediction model using a deep learning method. Bus Inf Syst Eng 62:143–157

Mehdiyev, N., Fettke, P. (2021). Explainable Artificial Intelligence for Process Mining: A General Overview and Application of a Novel Local Explanation Approach for Predictive Process Monitoring. In: Pedrycz, W., Chen, SM. (eds) Interpretable Artificial Intelligence: A Perspective of Granular Computing. Studies in Computational Intelligence, Springer, Cham, vol 937:1–28. https://doi.org/10.1007/978-3-030-64949-4_1

Nagahisarchoghaei M, Nur N, Cummins L et al (2023) An empirical survey on explainable ai technologies: Recent trends, use-cases, and categories from technical and application perspectives. Electronics 12(5). https://doi.org/10.3390/electronics12051092

Nauta M, Trienes J, Pathak S et al (2023) From anecdotal evidence to quantitative evaluation methods: A systematic review on evaluating explainable ai. ACM 55(13s)

OpenAI, Achiam J, Adler S et al (2024) Gpt-4 technical report. https://arxiv.org/abs/2303.08774

Pasquadibisceglie V, Appice A, Castellano G et al (2019) Using convolutional neural networks for predictive process analytics. In: 2019 International Conference on Process Mining (ICPM), IEEE, 129–136. https://doi.org/10.1109/ICPM.2019.00028

Peeperkorn J, Sv Broucke, De Weerdt J (2023) Can recurrent neural networks learn process model structure? J Intell Inf Syst 61(1):27–51

Rama-Maneiro E, Vidal JC, Lama M (2021) Deep learning for predictive business process monitoring: Review and benchmark. IEEE Trans Serv Comput 16(1):739–756

Rizzi W, Di Francescomarino C, Maggi FM (2020) Explainability in predictive process monitoring: When understanding helps improving. BPM Forum. Springer, Cham, pp 141–158
Rojat T, Puget R, Filliat D et al (2021) Explainable artificial intelligence (xai) on timeseries data: A survey. https://arxiv.org/abs/2104.00950
Senderovich A, Di Francescomarino C, Maggi FM (2019) From knowledge-driven to data-driven inter-case feature encoding in predictive process monitoring. Inf Syst 84:255–264
Serrano S, Smith NA (2019) Is attention interpretable? In: Proc. of the 57th Annual Meeting of the ACL. ACL, Florence, pp 2931–2951
Sindhgatta R, Moreira C, Ouyang C et al (2020) Exploring interpretable predictive models for business processes. Springer
Stevens A, De Smedt J, Peeperkorn J (2022) Quantifying explainability in outcome-oriented predictive process monitoring. Process Mining Workshops. Springer, Cham, pp 194–206
Stierle, Matthias; Brunk, Jens; Weinzierl, Sven; Zilker, Sandra; Matzner, Martin; and Becker, Jörg, Bringing Light Into the Darkness - A Systematic Literature Review on Explainable Predictive Business Process Monitoring Techniques (2021). ECIS 2021 Research-in-Progress Papers, 8. https://aisel.aisnet.org/ecis2021_rip/8
van der Aalst WMP (2016) Process Mining. Springer, Berlin. https://doi.org/10.1007/978-3-662-49851-4
Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, u. & Polosukhin, I. (2017). Attention is All you Need. In I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan & R. Garnett (eds.), Advances in Neural Information Processing Systems, : Curran Associates, Inc 5998–6008. http://arxiv.org/abs/1706.03762
Verenich I, Dumas M, Rosa ML et al (2019) Survey and cross-benchmark comparison of remaining time prediction methods in business process monitoring. ACM Trans Intell Syst Technol 10(4):1–34. https://doi.org/10.1145/3331449
Weinzierl S, Zilker S, Dunzer S et al (2024) Machine learning in business process management: A systematic literature review. Exp Syst Appl 253:124181. https://doi.org/10.1016/j.eswa.2024.124181
Weinzierl, S., Zilker, S., Brunk, J., Revoredo, K., Matzner, M., Becker, J. (2020). XNAP: Making LSTM-Based Next Activity Predictions Explainable by Using LRP. In: Del Río Ortega, A., Leopold, H., Santoro, F.M. (eds) Business Process Management Workshops. BPM 2020. Lecture Notes in Business Information Processing, Springer, Cham. vol 397: 129–141. https://doi.org/10.1007/978-3-030-66498-5_10
Wiegreffe S, Pinter Y (2019) Attention is not not explanation. Inui K, Jiang J, Ng V and Wan X (eds),In: Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP), Association for Computational Linguistics, Hong Kong, 11–20, https://doi.org/10.18653/v1/D19-1002
Zhao WX, Zhou K, Li J et al (2025) A survey of large language models. https://arxiv.org/abs/2303.18223

Publisher's Note