Enhancing Predictive Process Monitoring using Semantic Information

Jiaxin Yuan[1][0009−0005−2588−7845], Daniela Grigori[1][0000−0003−1741−8676], and Han van der Aa[2][0000−0002−4200−4937]

[1] Paris Dauphine University-PSL, Pl. du Maréchal de Lattre de Tassigny, 75016 Paris, France
jiaxin.yuan@dauphine.eu, daniela.grigori@lamsade.dauphine.fr
[2] University of Vienna, Universitätsring 1, 1010 Wien, Austria
han.van.der.aa@univie.ac.at

Abstract. Predictive Process Monitoring (PPM) leverages historical data to forecast information about ongoing business processes. Recent methods have utilized advanced deep learning and classical machine learning models. However, the role of semantic information that can be extracted from event logs has been underexplored, although such information has been demonstrated to have significant advantages for other process mining tasks, such as anomaly detection. Therefore, this paper proposes a novel mechanism that aims to exploit semantic information for PPM, particularly by extracting information regarding the status of business objects associated with process instances from event data. We evaluate this mechanism in outcome-oriented and next activity prediction tasks, using state-of-the-art large language models (LLMs) for semantic extraction. Our results show that integrating semantic information improves prediction performance across these tasks. This work demonstrates that utilizing semantic information in PPM has considerable potential, especially in combination with advanced language models.

Keywords: Process mining · Predictive process monitoring · Semantic information · Large language models.

1 Introduction

Predictive process monitoring (PPM) leverages historical data to forecast information about ongoing business process instances, thereby providing organizations with significant competitive advantages [13,18]. To address various tasks within this domain, numerous methods utilizing machine learning and deep learning models have been developed [8]. These methods employ a range of data preprocessing techniques and diverse prediction algorithms to tackle the challenges inherent in PPM [14].

Although few studies have explicitly explored the semantic information embedded within historical data, the potential benefits of incorporating such semantic information into predictive models are evident. For instance, in predicting the

outcome of a case, an *application* that is *complete* is more likely to succeed than an *incomplete* one. Similarly, in scenarios where the goal is to predict the next likely activity, if an *application* has been *rejected*, it is unlikely that it will proceed to *validation* again, thereby narrowing the potential search space. Relevant tasks in process mining such as anomaly detection have demonstrated significant advantages by leveraging semantic information [1,4]. These methods employ natural language processing (NLP) techniques to analyze the semantics of activity labels associated with events, enabling the identification of illogical behaviors. Furthermore, the increasing capabilities of large language models (LLMs) suggest that their integration into process mining is highly promising [2,9].

Therefore, this paper investigates integration of semantic information in PPM. Specifically, we propose a mechanism for PPM that utilizes semantic information by extracting and annotating the statuses of key business objects from event logs, through the use of LLMs. We tested this mechanism on both Outcome-Oriented Prediction (OOP) [13] and Next Activity Prediction (NAP) tasks [22], using real-life logs provided by a benchmark paper [23]. For NAP, we further validate our mechanism on additional logs. Our evaluation demonstrates that adopting our mechanism enhances prediction performance across both tasks. This underscores the potential effectiveness of discovering and leveraging semantic information in prediction tasks, indicating a promising future direction for this field.

The remainder is structured as follows. Section 2 reviews the related works in the field. Section 3 defines essential preliminaries. Section 4 describes the mechanism we introduced, which is further evaluated in Section 5. Finally, Section 6 concludes the paper by summarizing the findings, discussing their implications, and outlining potential directions for future research in this area.

2 Related Work

Research relevant to our work can be categorized into three areas: traditional methods and textual-aware solutions for PPM, semantic process mining, and the application of LLMs in the process domain.

Our work focuses on OOP and NAP, for which seminal benchmark articles have been published [19,23]. These articles suggest optimal configurations, including encoding, prefix generation, bucketing, and algorithm selection [23]. While extensive research in this area focuses on the discovery of more effective algorithms, ranging from Deep Learning (DL) approaches like transformers and LSTM to Machine Learning (ML) techniques such as SVM and RF [15,7,22,12], textual features remain largely underexplored. However, unstructured text from different process-related sources such as comment fields, emails, or documents is leveraged using traditional vectorization methods like bag of words (BoW) and Latent Dirichlet Allocation (LDA) [17]. More advanced encoding approaches, such as BERT [5], have been employed for textual features, aligning closely with recent studies on next activity prediction using LLMs, particularly from a control-flow perspective [21].

In other process mining tasks like anomaly detection, research has demonstrated significant success in utilizing semantic information, suggesting that integrating such information could enhance PPM tasks. Anomalous process behavior is identified by detecting semantically inconsistent execution patterns, with the benefits clearly demonstrated in real-world scenarios [1]. Combining machine learning methods with NLP techniques has led to significant improvements in precision and recall for semantic anomaly detection [6]. Tasks like the automatic semantic annotation of event data, which identifies specific semantic components from textual attributes, have inspired our approach to extracting business objects and their statuses [20].

Although our focus differs, our work also relates to recent research exploring the integration of LLMs into various aspects of process mining, such as process discovery and conformance checking [10,11]. Some research also delves into causal reasoning and explaining decision points [9]. Emerging benchmarks combining LLMs with process mining, such as those found in [2,3], mainly assess the quality of textual or coding answers provided by LLMs. However, these studies primarily explore how LLMs comprehend process mining artifacts or tasks rather than directly evaluating their performance on downstream tasks. More closely related to our task is a template extraction method performed using LLMs, followed by fine-tuning with a pre-trained language model, though their focus was solely on the next activity prediction task [16].

In our work, we combine these three streams by using LLMs to extract semantic information to be leveraged for different PPM tasks.

3 Preliminaries

In this section, we define preliminaries required for the remainder of the paper.
Event Data. To extract semantic information, our work takes an event log L as input, which is composed of traces. A trace $t = \langle e_1, e_2, \ldots, e_n \rangle \in L$ is a sequence of events associated with the same case. Each event in a trace can be represented as a tuple $(a, c, t, (d_1, v_1), \ldots, (d_m, v_m))$, where a denotes activity label, c is case id, t is the timestamp, and $(d_1, v_1), \ldots, (d_m, v_m)$ represents a number of data attributes and their corresponding values. PPM tasks are conducted over a set of prefixes of an event log. Therefore we define an event prefix of length k, noted as $hd^k(\sigma)$ is $hd^k(\sigma) = \langle e_1, \ldots, e_k \rangle$, where k is between 1 to $n-1$. Beyond these elements, from a broader perspective, the analysis that considers only the sequence of activities is referred to as control-flow analysis.

Classifier. In machine learning, a classifier is an algorithm that categorizes data into predefined classes. It learns from training data to map input features X to target labels y and predicts the class of new data. To be fed into a machine learning model, input features X are typically encoded as vectors, commonly referred to as feature vectors.

Outcome-Oriented Prediction. For OOP, the input X is an event prefix $hd^k(\sigma)$, while y is a finite set of categorical outcomes.

Next Activity Prediction. For NAP, the input X is the same as outcome-oriented tasks. We use a multi-class classifier $f : X \to \{a_1, a_2, \ldots, a_k\}$, where the label space is the set of unique activity labels $\{a_1, a_2, \ldots, a_k\}$.

4 Mechanism

This section presents our proposed semantic-aware status annotation mechanism for general PPM tasks. As shown in Fig. 1, our mechanism includes two steps specific to our pipeline (in red). *Semantic extraction* creates a static mapping from activity labels to their corresponding business objects and statuses. *Status annotation*, in turn, uses this mapping to annotate a prefix (for training) or an ongoing case (for inference) with information that captures the current status of each of its business objects. The rest of the PPM pipeline is in line with common practice, making our mechanism independent of the choice of a specific machine learning model or other configuration choices.

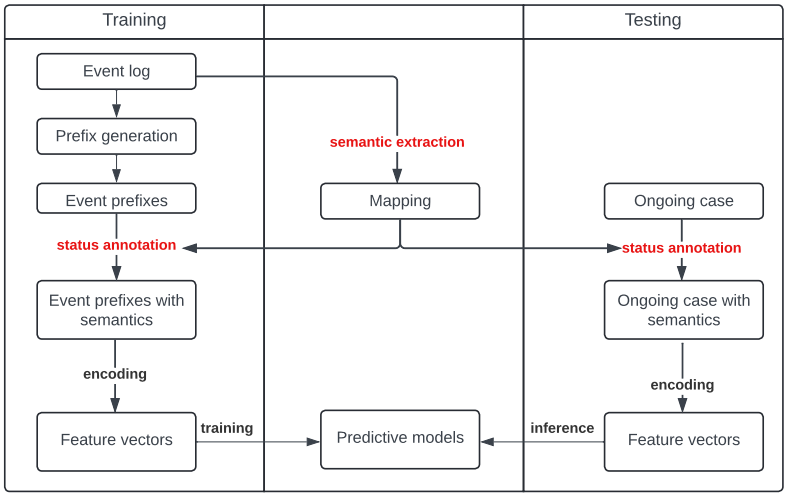

Fig. 1. Overview of our semantic-aware status annotation mechanism.

4.1 Semantic Extraction

The first step in our mechanism is semantic extraction, which creates a static mapping from each activity label in the label space of an event log to a collection of key-value pairs that identify business objects and their corresponding statuses. This mapping serves as a reference for generating semantic information for each event in the subsequent step. For example, from *Create fine*, we obtain *fine* as the business object and *created* as its status.

Semantic extraction can be conducted using LLMs. By providing a prompt with task-specific instructions, LLMs are able to comprehend the task and return the key-value pairs that identify business objects and their corresponding

statuses. Figure 2 illustrates an example of how we extract semantic information using Llama-3[3]. When a verb-substantive combination is present in the activity label, semantic extraction is straightforward. Otherwise, LLMs must rely on learned knowledge to infer meaning. For example, the activity label *Payment* provides no indication of status, but based on its pre-trained knowledge, the LLM assigns status *conducted* to the business object *payment*.

<|begin_of_text|><|start_header_id|>system<|end_header_id|>

You are an expert in process mining. Your task is to extract key business objects and their states from complex activity labels by mapping activity labels to a dictionary in the form of {activity label: {business object: status}}.
A single activity label can contain multiple essential business objects and states.
Business objects must be general and process-related.
Respond strictly in JSON format, providing fields for each activity. Ensure you understand the meaning and don't just extract directly. Output only the dictionary as the answer. Don't forget any of given activity labels.

Example activity labels:

{"Create Fine"}

Example answer:
{"Create Fine": {"fine": "created"}}

<|eot_id|><|start_header_id|>user<|end_header_id|>
{'Create Fine', 'Send Fine', 'Insert Fine Notification', 'Send Appeal to Prefecture', 'Receive Result Appeal from Prefecture', 'Appeal to Judge', 'Add penalty', 'Payment', 'Notify Result Appeal to Offender', 'Insert Date Appeal to Prefecture'}
<|eot_id|><|start_header_id|>expert<|end_header_id|>

Fig. 2. The prompt begins with role *system*, followed by a description of the task and constraints. It then presents examples of extractions, which may include multiple instances. The interaction continues in role *user*, listing activity labels for processing. Finally, the prompt concludes with role *expert*, where introduces the expected results.

4.2 Status Annotation

In this step, we annotate a prefix or ongoing case with information that captures the current status of all business objects, using the mapping derived in the first step. This is done by iteratively processing events that occurred, with each status being updated if a new event occurs that relates to that business object.

For example, consider an entire trace with a control flow $\langle$*Create Fine*, *Send Fine*, *Insert Fine Notification*, *Add Penalty*$\rangle$, with Case id *A100*. The control flow of event prefix of length 2, denoted as $hd^2(\sigma_{A100})$, is $\langle$*Create Fine*, *Send Fine*$\rangle$. During status annotation, all business objects extracted from previous step are initialized as *unprocessed*, including *fine* and

[3] https://huggingface.co/meta-llama/Meta-Llama-3-8B

penalty, as well as *payment* from other traces. Then business object *fine* is updated to *created* and subsequently to *sent*, while business objects *penalty* and *payment* remain unaffected and retain its initial status. For the event prefix $hd^2(\sigma_{A100})$, the final statuses of business objects that appear in the trace are recorded as $\{fine : sent,\ penalty : unprocessed,\ payment : unprocessed\}$, as illustrated in Figure 3. The status annotation process for ongoing cases is similar to that for event prefixes, as both capture the sequence of events that have occurred up to that point.

Case id	Event number	Activity	Resource	Amount	Business objects			Label	
					fine	penalty	payment	NAP	OOP
A100	1	Create Fine	R561	35	created	unprocessed	unprocessed	Send fine	1
A100	2	Send Fine	R561	35	sent	unprocessed	unprocessed	Insert fine notification	1
A100	3	Insert Fine Notification	R561	35	notified	unprocessed	unprocessed	Add penalty	1
A100	4	Add penalty	R1	71	notified	added	unprocessed	End	1
A1	1	Create Fine	R537	36	created	unprocessed	unprocessed	Send fine	0
A1	2	Send Fine	R537	36	sent	unprocessed	unprocessed	Insert Fine Notification	0
A1	3	Insert Fine Notification	R537	36	notified	unprocessed	unprocessed	Add penalty	0
A1	4	Add penalty	R537	74	notified	added	unprocessed	Payment	0
A1	5	Payment	R537	74	notified	added	conducted	End	0

Fig. 3. Snapshot of a collection of an event log with semantics. All unique activities are listed in the column *activity*. The example discussed in the text is highlighted, with the event prefix with semantics of $hd^2(\sigma_{A100})$ in red, and the associated labels for NAP and OOP in green.

4.3 PPM pipeline

After obtaining the enriched inputs with semantics, we proceed with a standard process within the PPM pipeline. During the training phase, we utilize the collection of event prefixes and their associated semantics to train predictive models for various PPM tasks. In the testing phase, we use the trained models to make predictions for ongoing cases. Semantic information can be seamlessly integrated into downstream tasks, similar to other categorical features within the inputs, with business objects representing the feature names and their final statuses as the corresponding values.

We particularly apply our mechanism to outcome prediction and next activity prediction tasks from PPM, as these are well-explored in the research and represent distinct concerns—one focusing on immediate prediction ability and the other on cumulative prediction ability. Furthermore, more so than remaining time prediction, these tasks can benefit from the integration of semantic information, as their targets may be closely related to the underlying semantics.

5 Evaluation Experiments

This section reports on the experiments conducted in our study. The objective was to compare prediction results with and without semantic information. The selection of event logs is detailed in Section 5.1, and the experimental settings are described in Section 5.2. The results of both OOP and NAP task are presented in Section 5.3. We discuss the results and limitations of our experiment in Section 5.4. The implementation can be found in our repository.[4]

5.1 Log Selection

As OOP requires labeled datasets, we initially limited our selection to the OOP benchmark and generated NAP labels from these datasets [23]. We selected five datasets based on the rule that they must be in English with activity labels having clear linguistic meaning, not cryptic notations. Following this rule, we retained three logs from *BPIC2017*, as well as *production* and *traffic fines* logs. Since generating labels for next activity prediction is straightforward, we further expanded our selection to include additional real-world logs commonly used in research, applying the same rule. This resulted in the inclusion of five logs from *BPIC2020*, *BPIC2019*, and *helpdesk*. All logs are available in our repository.

Data Preprocessing. We filter out traces containing fewer than three events, as they generally provide insufficient information at early stages. On the other hand, training with long prefixes is time-consuming, and the OOP becomes trivial at late stages [23]. Thus, we vary the prefix length from 3 to 20 during both training and testing.

Train-test split. Within each log, we use 80% of the traces for training and the remaining 20% for testing. To prevent overfitting, we ensure that the testing traces are completely excluded from the training set.

Characteristics. Table 1 gives a summary of the characteristic of the 12 datasets. Among all the datasets for OOP tasks, the labels are either positive or negative, denoting whether the outcomes deviate from or not to the defined rules [23]. In the table, *Pos rate* indicates the proportion of positive labels.

5.2 Experimental Setting

Implementation and Environment. Our implementation comprises two components: semantic extraction with LLMs on 2 Nvidia RTX 2080 Ti GPUs and 32GB RAM, and prediction using XGBoost in Python 3.8 on an Apple M2 CPU with 16GB RAM. The prediction phase was executed using a CPU, as XGBoost shows minimal GPU benefit from acceleration.

Configurations and baseline. We compare results obtained for three settings:

- *w/o semantics*: This represents a baseline that only captures control-flow information. We specifically use aggregation encoding on the *Activity attribute*.

[4] https://github.com/jiaxin-yuan/semantic_status_annotation

Table 1. Characteristics of the chosen logs.

Dataset	Traces	Trace length		Variants	Events	Activity	OOP:
		Avg.	Max			Classes	Pos rate
bpic17_a	31,413	45.45	180	15,846	1,198,366	26	0.41
bpic17_c	31,413	45.45	180	15,846	1,198,366	26	0.47
bpic17_r	31,413	45.45	180	15,846	1,198,366	26	0.12
production	220	20.34	78	203	2,489	26	0.53
traffic	129,615	4.07	20	200	460,556	10	0.46
bpic19	251,734	33.23	990	11,973	1,595,923	42	–
bpic20i	6,449	11.19	27	753	72,151	34	–
bpic20d	10,500	5.37	24	99	56,437	17	–
bpic20permit	7,065	14.80	90	1,478	86,581	51	–
bpic20prepaid	2,099	9.27	21	202	18,246	29	–
bpic20request	6,886	5.74	20	89	36,796	19	–
helpdesk	4,580	4.66	15	226	21,348	14	–

For example, an event prefix of length 2 in the trace $\langle a, b, c, d\rangle$ is encoded as $\{a : 1, b : 1, c : 0, d : 0\}$. The aggregation method was chosen for encoding due to its relatively superior performance, as reported in the OOP benchmark [23].

- *w sem(LLM)*: This configuration adds semantic information as obtained through our mechanism to the feature vector. All other configuration options are the same as for the baseline.
- *w sem(hard)*: This configuration is the same as the previous one, with the exception that we use a manually defined (i.e., hardcoded) mapping as the output for Step 1. This configuration allows us to test if extraction mistakes by the LLM impact the prediction accuracy.

In our LLM-based configuration, we employed in-context learning with LLaMa-3 for extraction. Experimentation with various prompting settings revealed that few-shot prompting with more examples typically yielded more stable and accurate results. Consequently, we used six examples per log in our experiments.

Performance Metrics. For OOP, we adopted the Area Under the Curve (AUC), indicating the likelihood of ranking a positive instance higher than a negative one. For NAP, we employed overall accuracy and Macro-Averaged F1 Score as metrics [19]. Accuracy measures correct classifications, while the F1 Score balances precision and recall. The Macro-Averaged F1 Score averages F1 Scores across classes equally, making it suitable for imbalanced distributions.

5.3 Results

Result of OOP. Table 2 presents the results for the outcome-oriented prediction task, measured by AUC among all testing set. Across almost all datasets, settings that incorporate semantics extracted either through hard coding or LLMs consistently outperform those that do not utilize semantics. Though, an exception is observed in the *production* dataset, where the performance of LLM-extracted semantics is lower compared to non-semantics, whereas semantics extracted man-

ually still perform better than non-semantics. In most cases, manually extracted features perform similarly to those extracted by LLMs.

Table 2. Overall AUC for OOP tasks obtained on the test set, using XGBoost. Bold numbers indicate the best score for that dataset.

method	**dataset**				
	bpic17_a	**bpic17_c**	**bpic17_r**	**production**	**traffic**
w/o semantics	0.6624	0.6377	0.5743	0.6440	0.6225
w sem(hard)	**0.6636**	**0.6385**	0.5751	**0.6449**	**0.6228**
w sem(LLMs)	**0.6636**	0.6380	**0.5758**	0.6396	**0.6228**

Result of NAP. Table 3 presents the results for the next activity prediction task, evaluated using overall accuracy and Macro-Averaged F1 Score across the test set. The *BPIC2017* subsets *accepted*, *cancelled*, and *refused* originate from the same process, resulting in identical values. In most datasets (7/10), both metrics consistently show performance improvements with the incorporation of semantics. Inconsistencies between the metrics are observed in datasets with limited samples, such as *Production*, and in those with over 40 activity labels. Only the *traffic* dataset consistently underperforms when semantics are incorporated. In most cases, extraction by LLMs outperforms human extraction for this task.

Table 3. Overall accuracy (first row) and Macro-Averaged F1 Score (second row) for the NAP tasks on the test set, using XGBoost. Bold values represent the highest score for each dataset within the same evaluation metric.

method	**dataset**				
	bpic17	**production**	**traffic**	**bpic20d**	**bpic20i**
w/o semantics	0.8680	0.0518	**0.8182**	0.3042	0.7041
	0.7204	0.0289	**0.5506**	0.1191	0.3530
w sem(hard)	0.8959	0.0621	0.8181	0.3045	0.7042
	0.7251	**0.0407**	0.5505	0.1193	0.3540
w sem(LLMs)	**0.8960**	**0.0647**	0.8181	**0.3271**	**0.7098**
	0.7252	0.0397	0.5505	**0.1324**	**0.3759**
	bpic20permit	**bpic20prepaid**	**bpic20request**	**helpdesk**	**bpic19**
w/o semantics	0.2025	0.6541	0.4937	0.2406	0.0093
	0.2370	0.3913	0.1533	0.1132	**0.0097**
w sem(hard)	**0.2040**	0.6546	0.4941	**0.2427**	**0.0097**
	0.2358	0.3938	0.1556	**0.1146**	0.0093
w sem(LLMs)	0.2034	**0.6610**	**0.5043**	0.2411	**0.0097**
	0.2388	**0.4446**	**0.1702**	0.1143	0.0093

Earliness. For the OOP task, we assess earliness by calculating AUC at each prefix length, as illustrated in Figure 4. As the three logs from *BPIC2017* exhibit similar trends, only one is presented here. The performance gains from using semantics are more pronounced in the earlier stages, as demonstrated in

three datasets from the *BPIC2017* log. The AUC score without semantics is notably poorer compared to two settings with semantics, particularly for prefix lengths between 3 and 5. Despite the performance gains from using semantics in the *traffic fines* figure being less visually apparent, the numerical results show that semantics outperform non-semantics for prefix lengths 5, 6, and 7, with improvements of 0.2%, 0.03%, and 0.009%, respectively, while remaining consistent across other prefix lengths. The *Production* dataset shows no clear pattern, but the inclusion of semantics leads to superior performance for prefix lengths 5 to 10.

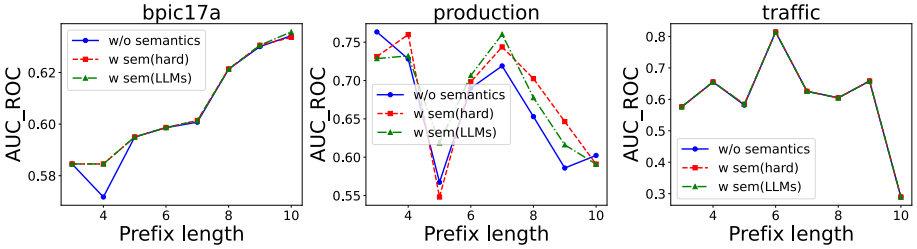

Fig. 4. Earliness of the OOP task across different datasets, limited to prefix lengths below 10.

5.4 Discussion

Overall, we demonstrate that semantics can enhance PPM performance, as indicated by the general performance improvements achieved with our mechanism. However, these improvements vary a lot across datasets and settings. By summarizing the prediction performances from both the OOP and NAP tasks, as well as the earliness of the OOP task, we derive the following key insights.

Task comparison. Performance gains are generally more evident in NAP than in OOP, indicating that semantics are more beneficial for immediate predictions than for cumulative prediction. On the other hand, performance varies considerably across different datasets, both in terms of overall performance and the magnitude of performance and of performance improvements. This variability suggests that the benefits of our mechanism differ among datasets, which can be attributed to the extent of semantic information contained in the activity labels.

Extraction method comparison. Overall, semantic extraction using LLMs outperforms manual extraction in both tasks. Excluding datasets with inconsistent results across both metrics, exceptions are observed in *Helpdesk*. Despite having a limited number of activity labels, similar to *traffic fines* and *BPIC20d*, *Helpdesk* lacks business objects within its activity labels, making it difficult for the LLM to accurately conduct semantic extraction from the context.

Earliness. The consistent performance gains observed in most datasets highlight promising potential in prediction earliness, which is crucial for the OOP task. The oscillations observed in *Production* likely stem from the limited sample size at each prefix length, with the highest number of examples per group being below 40, making the results sensitive to individual samples.

Limitations. Due to the probabilistic nature of LLMs, limitations such as the randomness in optimal extractions are inherent. Additionally, time constraints restricted us from comparing different prefix encoding methods, employing more robust techniques like cross-validation, or exploring the selection of optimal predictive algorithms.

6 Conclusion

This paper addresses the underexplored role of semantic information in PPM. By introducing a novel mechanism for extracting and annotating the statuses of business objects from event logs, we demonstrated that incorporating semantic information can enhance predictive accuracy for outcome-oriented and next activity prediction.

However, our work has some limitations. Currently, our mechanism focuses sorely on statuses associated with individual business objects and does not account for semantic interactions between business objects using linguistic knowledge, such as synonyms, near-synonyms, and antonyms. For example, the status *cancelled* of the business object *booking* may trigger status updates in other related objects, such as changing the status of *bills* from *unpaid* to *cancelled*.

In future work, we plan to explore the role of semantic information in prescriptive process monitoring, incorporating the additional linguistic knowledge mentioned earlier. Building on the foundations of predictive process monitoring, we hope this will help identifying effective intervention actions for different scenarios in advance, thereby offering actionable recommendations for improvement in process monitoring.

Disclosure of Interests. The authors have no competing interests to declare that are relevant to the content of this article.

References

1. van der Aa, H., Rebmann, A., Leopold, H.: Natural language-based detection of semantic execution anomalies in event logs. Information Systems **102**, 101824 (2021)
2. Berti, A., Kourani, H., van der Aalst, W.M.: Pm-llm-benchmark: Evaluating large language models on process mining tasks. arXiv preprint arXiv:2407.13244 (2024)
3. Berti, A., Qafari, M.S.: Leveraging large language models (llms) for process mining (technical report). arXiv preprint arXiv:2307.12701 (2023)
4. Busch, K., Kampik, T., Leopold, H.: xsemad: Explainable semantic anomaly detection in event logs using sequence-to-sequence models. arXiv preprint arXiv:2406.19763 (2024)
5. Cabrera, L., Weinzierl, S., Zilker, S., Matzner, M.: Text-aware predictive process monitoring with contextualized word embeddings. In: International Conference on Business Process Management. pp. 303–314. Springer (2022)
6. Caspary, J., Rebmann, A., van der Aa, H.: Does this make sense? machine learning-based detection of semantic anomalies in business processes. In: International Conference on Business Process Management. pp. 163–179. Springer (2023)

7. Chen, H., Fang, X., Fang, H.: Multi-task prediction method of business process based on bert and transfer learning. Knowledge-Based Systems **254**, 109603 (2022)
8. Evermann, J., Rehse, J.R., Fettke, P.: Predicting process behaviour using deep learning. Decision Support Systems **100**, 129–140 (2017)
9. Fahland, D., Fournier, F., Limonad, L., Skarbovsky, I., Swevels, A.J.: How well can large language models explain business processes? arXiv preprint arXiv:2401.12846 (2024)
10. Grohs, M., Abb, L., Elsayed, N., Rehse, J.R.: Large language models can accomplish business process management tasks. In: International Conference on Business Process Management. pp. 453–465. Springer (2023)
11. Kourani, H., Berti, A., Schuster, D., van der Aalst, W.M.: Process modeling with large language models. In: International Conference on Business Process Modeling, Development and Support. pp. 229–244. Springer (2024)
12. Kratsch, W., Manderscheid, J., Röglinger, M., Seyfried, J.: Machine learning in business process monitoring: a comparison of deep learning and classical approaches used for outcome prediction. Business & Information Systems Engineering **63**, 261–276 (2021)
13. Maggi, F.M., Di Francescomarino, C., Dumas, M., Ghidini, C.: Predictive monitoring of business processes. In: CAiSE. pp. 457–472 (2014)
14. Neu, D.A., Lahann, J., Fettke, P.: A systematic literature review on state-of-the-art deep learning methods for process prediction. Artificial Intelligence Review **55**(2), 801–827 (2022)
15. Ni, W., Zhao, G., Liu, T., Zeng, Q., Xu, X.: Predictive business process monitoring approach based on hierarchical transformer. Electronics **12**(6), 1273 (2023)
16. Oved, A., Shlomov, S., Zeltyn, S., Mashkif, N., Yaeli, A.: Snap: Semantic stories for next activity prediction. arXiv preprint arXiv:2401.15621 (2024)
17. Pegoraro, M., Uysal, M.S., Georgi, D.B., van der Aalst, W.M.: Text-aware predictive monitoring of business processes. arXiv preprint arXiv:2104.09962 (2021)
18. Philipp, P., Jacob, R., Robert, S., Beyerer, J.: Predictive analysis of business processes using neural networks with attention mechanism. In: 2020 International conference on artificial intelligence in information and communication (ICAIIC). pp. 225–230. IEEE (2020)
19. Rama-Maneiro, E., Vidal, J.C., Lama, M.: Deep learning for predictive business process monitoring: Review and benchmark. IEEE Transactions on Services Computing **16**(1), 739–756 (2021)
20. Rebmann, A., van der Aa, H.: Enabling semantics-aware process mining through the automatic annotation of event logs. Information Systems **110**, 102111 (2022)
21. Rebmann, A., Schmidt, F.D., Glavaš, G., van der Aa, H.: Evaluating the ability of llms to solve semantics-aware process mining tasks. arXiv preprint arXiv:2407.02310 (2024)
22. Tax, N., Verenich, I., La Rosa, M., Dumas, M.: Predictive business process monitoring with lstm neural networks. In: International Conference on Advanced Information Systems Engineering. pp. 477–492. Springer (2017)
23. Teinemaa, I., Dumas, M., Rosa, M.L., Maggi, F.M.: Outcome-oriented predictive process monitoring: Review and benchmark. ACM Transactions on Knowledge Discovery from Data (TKDD) **13**(2), 1–57 (2019)