

Enhancing Predictive Process Monitoring on Small-Scale Event Logs Using LLMs

Alessandro Padella, Paolo Fazzetto, Nicolò Navarin, and Massimiliano de Leoni

University of Padua, Padua, Italy

alessandro.padella@unipd.it, paolo.fazzetto@unipd.it
nnavarin@math.unipd.it deleoni@math.unipd.it

Abstract. Predictive Process Monitoring is a process-mining research direction that aims to predict the future of an uncompleted process execution. The vast majority of research work focuses on techniques that are “data greedy” and require a lot of event data to be sufficiently accurate. However, the recent development of Large Language Models presents significant opportunities and potential benefits across various industrial and research domains. They are capable of leveraging their pre-trained knowledge to understand and complete tasks effectively. This paper reports on the design and implementation of a Predictive Process Monitoring framework based on Large Language Models. Experiments on multiple event logs confirm our hypothesis that Large Language Models are capable of providing very accurate predictions, even with as few as 10 training traces.

Keywords: Predictive Process Monitoring · Large Language Models · Few-Shot Prompting · Small-Scale Event Log

1 Introduction

Predictive Process Monitoring (PPM) is a family of techniques that leverages event logs from business processes to generate predictions about the future states or properties of ongoing process instances [12]. PPM methods vary depending on the prediction target, which can include times [35], next activities [18], or process outcomes [31].

Literature has extensively explored Machine and Deep Learning models to enhance prediction quality [4]. However, these models typically require large amounts of data for effective training. When the available event log is limited in size, the applicability of such techniques becomes constrained, reducing the overall potential of PPM. As highlighted in [37], data availability remains one of the most significant challenges faced by researchers and practitioners in this domain.

This paper puts forward a framework for PPM that leverages the potentials of Large Language Models (LLMs) to estimate the duration of process executions. LLMs can leverage their embedded knowledge to generate accurate predictions while demonstrating robustness in handling noisy and unstructured data [23]. Their ability to operate effectively with limited training examples makes them a promising alternative for improving PPM in data-scarce environments [11]. Since LLMs rely on prompts to process and generate responses, designing an effective prompt is crucial for encoding process traces.

The prompt must structure the input data in a way that preserves the sequence, dependencies, and relevant attributes of events while remaining concise and interpretable for the LLM. To answer this question, we designed the framework with two possible encodings: *(i)* an encoding in which traces are provided as vectors in which each component refers to one activity and indicates the number of occurrences of that activity in the trace, as extensively employed in the literature [7,19,30,29] *(ii)* a second encoding in which the trace itself is given to the LLM in a novel, purpose-built function, which only maintains global attributes, activities, and times.

The framework was instantiated for Gemini¹: a freely-available LLM that performs the best, according to the recent standard benchmarks.² Experiments were conducted on five processes and event logs, which were temporally divided into training and test logs. Multiple, random sub-logs with 2, 10, and 100 traces were extracted from the training log. These sub-logs have been provided as input to Gemini in order to enable the predictions on the test traces.

Results show that our LLM-based framework is capable of generating accurate predictions with fewer than 100 trace examples. In some cases, accurate predictions were achieved with as few as two traces. By contrast, a Catboost-based benchmark, which is known to be among the best-performing PPM predictors [24,3,10], demonstrated low accuracy when trained on the same amount of traces. Notably, even if only employing a few traces, our LLM framework consistently outperformed the benchmark predictor trained on the complete dataset.

The remainder of this paper is organized as follows: Section 2 analyzes the relevant literature in LLMs and PPM. Section 3 introduces the preliminary knowledge needed for the development of the approach, outlined in Section 4. Experiments on five case studies are reported in Section 5, while Section 6 resumes the paper and highlights next potential research directions.

2 Related Works

In this Section, we report the literature relevant for PPM and the application of LLMs within business process management. Section 2.1 reports on the relevant literature in PPM, Section 2.2 deals with the significant literature of LLMs, while Section 2.3 provides an overview of the application of LLMs in the business process management field.

2.1 Predictive Process Monitoring

Over the years, PPM frameworks leveraged Machine learning, Deep Learning, and ensemble methods. Deep learning techniques, such as Long Short-Term Memory networks [1] and Process Graph Transforming models [8], have shown strong performance, with research also focusing on improving training efficiency [26] and robustness [27]. Despite their success, alternative methods, including Local Process Models [34], Support Vector Machines [18], and Random Forest [5], remain relevant, particularly for

¹Gemini Technical Report: <https://arxiv.org/abs/2312.11805>

²See <https://llm-stats.com> for an overview of the most popular LLMs

real-time applications due to their lower computational demands. However, boosting on decision trees, such as XGBoost and CatBoost, has emerged as a best approach, striking a balance between predictive effectiveness and efficiency [24,3,10].

2.2 Large Language Models

LLMs have gained prominence for their performance, versatility, and ability to generate modular responses through chat interfaces. Research shows they can effectively predict time series using text tokenization of numerical data [11], by leveraging contextual information to enhance predictions [32]. These approaches circumvent limitations of traditional forecasting by eliminating models' design and manual encoding of priors. Requeima et al. [23] demonstrated that zero-shot sequence completion can produce accurate forecasts while incorporating textual side information. Additional studies confirm that integrating contextual information in prompts significantly improves LLMs' forecasting capabilities [2].

2.3 Large Language Models in Business Process Management

LLMs are recently attracting growing focus in business process management as well [9]. They have proven to be significantly useful for many process mining tasks, such as process modeling [13], log extraction [6], anomaly detection [33] and they have also been used for assessing the validity of some traces for a given process [22]. Lashkevich et al. in [16] provide a state-of-the-art approach that leverages LLMs for enhancing the optimization of waiting times and, on the other hand, it relies on user-prompted feedback for recommending more effective re-design options. Rebmann et al. [22] present an approach for extracting knowledge from textual data, also providing some textual and synthetically generated datasets for extracting event logs trying to assess if there are missing activities in the extracted processes and eventually generating them. In the meanwhile, the work in [20] leverages LLMs to transform textual data into process representations, followed by training a text-encoding technique BERT-based deep learning model to predict the next-activity in a process. Finally, Kubrak et al. [14] developed an approach based on a chatbot for process analysis in which the LLM is used for explaining the recommendations provided by a model, enhancing their explainability. The method proposed in this study advances the field in two aspects: it introduces a general prompting framework that enables prediction using LLMs and shows that it maintains its predictive efficiency even under conditions of limited log training data.

3 Preliminaries

The starting point for a process mining-based system is an *event log*. An event log is a multiset of *traces*. Each trace is a sequence of events, each describing a particular *process instance* (i.e., a *case*) in terms of the *activities* executed, the associated *timestamps* and other different domain-related *attributes*.

Definition 1 (Events). Let \mathcal{A} be the set of process activities. Let \mathcal{T} be the set of process timestamps. Let $\mathcal{V} = \mathcal{V}_1 \times \mathcal{V}_2 \times \dots \times \mathcal{V}_m$ be the Cartesian product of the data attribute sets. An event is a tuple $(a, t_{start}, t_{end}, \vec{v}) \in \mathcal{A} \times \mathcal{T}^2 \times \mathcal{V}$ where a is the event activity, t_{start} and t_{end} the associated timestamps, and \vec{v} the vector of associated attributes.

A trace is a sequence of events. The same event can occur in different traces. Namely, attributes may be given the same assignment in different traces. This means that the same trace can appear multiple times, although admittedly under extremely rare conditions, and motivates why an event log has to be defined as a multiset of traces:

Definition 2 (Traces & Event Logs). Let $\mathcal{E} = \mathcal{A} \times \mathcal{T}^2 \times \mathcal{V}$ be the universe of events. A trace σ is a sequence of events, i.e. $\sigma \in \mathcal{E}^*$.³ An event log \mathcal{L} is a multiset of traces, i.e. $\mathcal{L} \subset \mathbb{B}(\mathcal{E}^*)$.⁴

Given an event $e = (a, t_{start}, t_{end}, \vec{v})$, the remainder uses the following shortcuts: $activity(e) = a$, $start(e) = t_{start}$, $end(e) = t_{end}$, $duration(e) = t_{start} - t_{end}$, $attr(e) = \vec{v}$. Also, given a single attribute set \mathcal{V}_i it can be classified as *global* or *local*, depending on whether the values in it can vary or not in the same trace. We refer to this attributes as $global(\sigma) = \vec{g}$ and $local(\sigma) = \vec{l}$, and so the equation $global(\sigma) \oplus local(\sigma) = attr(\sigma)$ holds.⁵ Furthermore, given a trace $\sigma = \langle e_1, \dots, e_n \rangle$, $prefix(\sigma)$ denotes the set of all prefixes of σ , including σ , namely $prefix(\sigma) = \{\langle \rangle, \langle e_1 \rangle, \langle e_1, e_2 \rangle, \dots, \langle e_1, \dots, e_n \rangle\}$.

The goal of a time prediction framework is to forecast the total execution time of a running process instance that has not completed yet, namely a *running trace*. In this paper, the problem is modeled as the estimation of a **Total Time Function** $\mathcal{T} : \mathcal{X} \rightarrow \mathbb{N}_0$ that given a running trace $\sigma' = \langle e_1, \dots, e_k \rangle$ eventually completing as $\langle e_{k+1}, \dots, e_n \rangle$, returns the value $end(e_n) - start(e_1)$. The input of the Total Time function is a set \mathcal{X} , since not every approach shares the same encoding for event logs. For instance, in [1] the authors encoded traces in an LSTM compatible input, while in [25] the traces are encoded in a *comma-separated values* file suitable for a predictor based on a Decision Tree. This requires defining the **trace-to-instance encoding function** $\rho : \mathcal{E}^* \rightarrow \mathcal{X}$ with the goal of accurately translating every trace of the event log into an input suitable for the model. This function has proven to be significantly different based on the chosen predictive approach (cf. [28]).

Figure 1 depicts an example of a trace-to-instance encoding function. In it, the trace is preprocessed by adding the past activities in newly generated columns. For each activity in the trace, the number of previous occurrences of that activity is reported in a dedicated column, encoding the number of past executions of the activity. This encoding allows tracking the frequency of all past activities but does not maintain information about their sequential order, recording only the most recent one.

To formally define this encoding function we have to first define $\rho_{\mathcal{A}}^{hist}(\langle e_1, \dots, e_n \rangle)$. Here, for each activity $a \in \mathcal{A}$, one dimension exists in $\rho_{\mathcal{A}}^{hist}(\sigma) : \mathcal{E}^* \rightarrow (\mathbb{N})^{|\mathcal{A}|}$

³The operator $*$ refers to the Kleene star: given a set A , A^* contains all the possible finite sequences of elements belonging to A .

⁴ $\mathbb{B}(X)$ indicates the set of all multisets with the elements in set X .

⁵Considering \oplus as the concatenation of vectors e.g.

$[1, 3, 'request_created'] \oplus [2, True] = [1, 3, 'request_created', 2, True]$

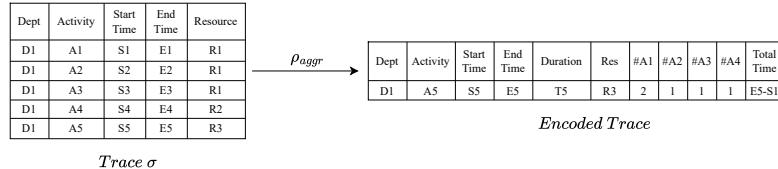


Fig. 1: Output example of a Aggregated History encoding function.

that takes on a value equal to the number of events $e_i \in \sigma$ such that $activity(e_i) = a$ for $i = 1, \dots, n$. The function ρ_{aggr} is then defined as: $\rho_{aggr}(\langle e_1, \dots, e_n \rangle) = global(\langle e_1, \dots, e_n \rangle) \oplus activity(e_n) \oplus start(e_n) \oplus end(e_n) \oplus duration(e_n) \oplus local(e_n) \oplus \rho_{\mathcal{A}}^{hist}(\langle e_1, \dots, e_n \rangle) \oplus end(e_n) - start(e_1)$. As reported in the Section 1, this function has been widely adopted in the literature [24,3,10]

Since this paper aims to leverage a LLM to estimate the total time of a running trace σ' , we introduce a general definition of it to establish the necessary conceptual framework:

Definition 3 (Large Language Model). Let Σ^* be the set of all finite strings over an alphabet Σ . A Large Language Model (LLM) is here modeled as a function $LLM : \Sigma^* \rightarrow \Sigma^*$ that, given an input $s \in \Sigma^*$, returns a string $LLM(s)$, which depends on the specific interpretation of the model.

We are aware that this is a simplification of the actual reality; however, as this paper seeks to propose a method for enhancing prediction quality through the use of an LLM, we provide a broad, simplified mathematical definition of LLMs, solely for the purpose of providing a reference function. This prediction method has been adapted from a more formal framework introduced in [23].

It is worthwhile pointing out that the remainder of this paper uses the term *training* to refer to both LLMs and Machine- and Deep-Learning models, to keep the discussion simple. However, we acknowledge that LLMs are already pre-trained: traces are provided to the LLMs as background, and are not formally used to train its internal parameters.

4 Approach For Small-Scaled Prediction Based on LLMs

This study seeks to leverage the potential of LLMs to develop a framework for PPM, particularly in scenarios where only a small amount of example traces are accessible. Leveraging their embedded knowledge, LLMs can extract and use additional information beyond the event data by incorporating the semantics of events, such as activity names, that traditional models cannot. Figure 2 depicts the proposed approach. Given an event log of completed traces \mathcal{L} and a running trace σ' , a trace-to-instance encoding function ρ is applied to transform them into a structured prompt. This prompt, composed of multiple components, is then used to enable the LLM to estimate the Total Time function \mathcal{T} .

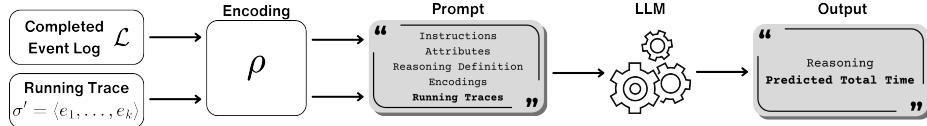


Fig. 2: Pipeline outlining the proposed method using LLMs for PPM.

In the remainder of this Section a new trace-to-instance encoding function ρ_{seq} suitable for LLM is introduced in Section 4.1, while Section 4.2 defines a context-based prompt suitable for employing an LLM for implementing the Total Time function defined in Section 3.

4.1 An Encoding Function for LLMs

Exploiting an LLM for developing a PPM framework is a topic that has not yet been explored in process mining (cf. Section 2). This section introduces a new LLM-suitable trace-to-instance encoding function $\rho_{seq} : \mathcal{E}^* \rightarrow \Sigma^*$ that is associated to an encoding that will be referred as **Sequential**.

The input of the ρ_{seq} function is a trace, while the output is a textual prompt that will be later enhanced to become suitable as input for the LLM function defined in Definition 3, that will be used as Total Time function \mathcal{T} . Specifically, in this case the generic input set of the Total Time function \mathcal{X} is equal to Σ^* . In ρ_{seq} , each trace $\sigma = \langle e_1, \dots, e_n \rangle$ is mapped into a string composed of three main elements:

- The values of the global attributes of the trace $global(e_n)$.
- A sequence of tuples $(activity(e_i), duration(e_i))$ for $i = 1, \dots, n$.
- The actual value of Total Time $end(e_n) - start(e_1)$.

Formally:

$$\rho_{seq}(\langle e_1, \dots, e_n \rangle) = global(\langle e_1, \dots, e_n \rangle) \oplus (activity(e_1), duration(e_1)) \oplus \dots \oplus (activity(e_n), duration(e_n)) \oplus end(e_n) - start(e_1)$$

The sets of local attributes have been intentionally excluded, as it has been demonstrated that LLMs are constrained by two primary factors: **technical** limitations and **methodological** considerations. From a technical perspective, an LLM can only process a certain number of characters; so it becomes necessary to reduce the size of the input to stay within this maximum quantity, namely the Context Length.⁶ Note that the Context Length of an LLM is not just a limitation per interaction (e.g., in a chatbot) but an inherent architectural constraint. Additionally, from a methodological standpoint, research demonstrates that the data in an LLM input does not equally impact the model's processing, and the significance of individual data points reduces as the input lengthens, even degrading its performance [15,17]. Therefore, we opted to omit local attributes. Conversely, global attributes were retained since they incorporate domain knowledge and have proved to retain more predictive power than local ones (cf. Galanti et al. [10]).

⁶See <https://llm-stats.com/> for an overview of Context Lengths of the latest models.

```
"{ Dept : D1, ActTimeSeq : { (A1, E1-S1), (A2, E2-S2), (A3, E3-S3), (A4, E4-S4), (A5, E5-S5) }, TotalTime : ES-S1 } "
```

Fig. 3: Output example of a Sequential trace-to-instance encoding function ρ_{seq} .

An example of application of the Sequential trace-to-instance encoding function ρ_{seq} is depicted in Figure 3, where the example trace reported in Figure 1 has been processed as a string. The result is a string-form Python object primarily composed of three keys: i) *Dept*, associated with the value of the corresponding global attribute; ii) *ActTimeSeq*, associated with the list of tuples where each activity and its duration are recorded; and iii) *TotalTime*, representing the total duration of the trace.

It is important to note that the Sequential is not the only suitable encoding for LLMs. In fact, the proposed approach also supports the *Aggregated History Encoding*, associated with the ρ_{aggr} trace-to-instance encoding function, as the LLM can predict future values independently of the specific encoding. This flexibility allows the framework to adapt to different encoding strategies.

4.2 Context-Based Prompting Technique for LLMs in Predictive Process Monitoring

In this paper, an LLM is defined as a function that takes as input a string and returns a string containing the predicted value (cf. Definition 3). This Section uses the traces that have been encoded using a trace-to-instance encoding function ρ and incorporates them generating an input suitable for an LLM. In essence we define a prompting technique that allows the model to generate total time predictions along with corresponding reasoning procedure, starting from the encoded traces. The prompting technique is divided into seven key parts, also reported in the Example 1:

- **Initial instruction and Header:** The LLM is introduced to the task with the prompt: *“You are an expert in process mining and machine learning. Your task is to predict the ‘Total Time’ of process instances based on event logs of activities, where each process instance is a sequence of activities.”* (Lines 1-2)
- **Attributes and Encoding description:** Contextual information specific to the process is provided, and the trace-to-instance encoding function is described (ρ_{seq} in the example). (Lines 4-12)
- **Output and Reasoning format specification:** The expected structure for predicted values is defined. (Lines 14-22 and 29-35)
- **Running Trace Format Specification:** The format for describing a running trace is specified to the model. (Lines 24-27)
- **Domain-specific background information:** Additional details about the process from which the data have been extracted. (Lines 37-38)
- **Example Data Provision:** Encoded data are provided as example to the model. (Lines 39-45)
- **Running Trace Provision:** The running trace is provided in the same format as the examples, with a custom last activity referred as “Running”, as described at lines 24-27. (Lines 47-50)

Although the proposed encoding is general and applicable to various use cases, certain information within these seven components must be specified by the process analyst and may be optionally removed.

```

1 You are an expert in process mining and machine learning. Your task is to predict the 'total time' of
2 process instances based on event logs, as each process instance is a sequence of activities.
3
4 A event log is a collection of traces, where each trace represents a process instance.
5 Each trace is mapped as a sequence of activities and integers representing the minutes since the start
6 of the process.
7 The log is represented as a python list containing one dictionary for each trace. Included in it are:
8 - the key "ApplicationType", representing the type of application
9 - the key "RequestedAmount", representing the total amount of euros requested in the loan application.
10 - the key "ActTimeSeq", which value is a list of [activity, cumulative elapsed minutes]
11 - The key "total_time", which value is the total execution time in minutes from the start of the activity,
12 that is the value to predict.
13
14 All interactions will be structured in the following way, with the appropriate values filled in.
15
16 [[ ## reasoning ## ]]
17 {your step-by-step reasoning}
18
19 [[ ## answer ## ]]
20 {your predicted total time as an integer}
21
22 [[ ## completed ## ]]
23
24 In adhering to this structure, your objective is to analyze the event log, and apply reasoning to predict
25 the total time for a new case. This case belongs to a not-yet-completed process instance, represented by the
26 label "Running" in "ActTimeSeq", indicating that more activities are expected before reaching the conclusion
27 of the process instance.
28
29 Ensure to articulate each step of your thought process in the reasoning field, detailing how you identify
30 relationships with past cases and leverage your intuition about the meaning of activities to arrive at the
31 solution. The answer should be the final prediction of the total time for the given process instance.
32 Respond with the corresponding output fields, starting with the field [[ ## reasoning ## ]],
33 then [[ ## answer ## ]], and then ending with the marker for [[ ## completed ## ]].
34
35 Your task is to learn from them and predict the 'total time' values for that traces.
36
37 The process deals with a loan application process from a Dutch financial institution. It has been provided
38 in the Business Process Intelligence (BPI) challenge in 2017.
39 The following list shows some completed example cases with their total times:
40
41 {"ApplicationType": "New credit", "RequestedAmount": 5000.0, "ActTimeSeq": [{"W_Complete application", 11},
42 {"W_Call after offers", 1464}, {"W_Call after offers", 7486}], "total_time": "7486"}
43 {"ApplicationType": "New credit", "RequestedAmount": 15000.0, "ActTimeSeq": [{"W_Complete application", 13},
44 {"W_Call after offers", 14}, {"W_Validate application", 4328}, {"W_Validate application", 8792}],
45 "total_time": "8792"}
46
47 Now predict the total time for this new uncompleted case, considering that the case is still running:
48
49 {"Application_1000386745": {"ApplicationType": "New credit", "RequestedAmount": 18000.0,
50 "ActTimeSeq": [{"W_Complete application", 2}, {"W_Call after offers", 8571}, {"Running"]}}
```

Example 1: Prompting technique example for a loan application process (Bpi17). Lines that have to be provided by the process analyst are marked in bold. In the example, only 2 training traces are provided due to space limitation.

In Example 1 they have been highlighted in bold. They are specifically, (i) the domain-specific background information (Lines 37-38) and (ii) the description of the global attributes (Lines 8-9), as they contain contextual details specific to the process under study. Notably, these informations are optional and can be excluded if necessary.

The remaining sections of the prompt are designed to be generic and can be applied to any event log without modification.

This modularity ensures that the framework can be associated with any encoding function and eventually be customized with details about the single process, while minimizing the effort needed for customization when applying the LLM to different process datasets.

5 Evaluation and Results

We assess the effectiveness of our approach in generating accurate predictions using a limited-size training set and comparing its performance against a state-of-the-art benchmark across five distinct case studies. To ensure a robust evaluation, we repeatedly sampled a limited number of traces, trained both a benchmark model and the LLMs on these samples, and conducted multiple experimental runs. The remainder of this section is organized as follows: The case studies are reported in Section 5.1, while Section 5.2 reports on the experimental setup, dealing with the implementation and the metrics used. Section 5.3 deals with the results and the associated comments. Finally, an example of the LLM’s output in terms of values and reasoning is reported in Section 5.4.

5.1 Case Studies

The validity of the approaches was assessed using five different processes, for which accordant event logs are available:

Bpi12: This process has been used by the BPI challenge in 2012⁷, it contains 8,616 traces, 6 different activities and 1 global attribute: *Requested_Amount*.

Bac: A process referring to a process of a Bank Institution that deals with the closures of bank accounts. It contains 32,429 completed traces, 15 different activities and 2 global attributes: *Closure_Type*, and *Closure_Reason*.⁸

Hospital: This process has been provided by an hospital emergency department. The log is made of 37,945 completed traces, contains 46 different activities and 3 global attributes, that are *Triage_Color*, *Triage_Access* and *Patient_Age*.

Purchasing: A process provided as part of the Fluxicon Disco tool and it is related to a purchase-to-pay (P2P) system, it is synthetic and generated from a model not available to the authors.⁹ The extracted event log has 608 traces. It contains 21 different activities and no global attributes.

Bpi17: The subprocess for the workflow-relevant in the 2017 BPI Challenge event data, and it is provided by the same financial institution that provides the log employed in *Bpi12*. It contains 30,276 completed traces, 8 different activities and 2 global attributes that are the *Loan_Goal*, and the *Requested_Amount*.

⁷<https://doi.org/10.4121/uuid:3926db30-f712-4394-aebc-75976070e91f>

⁸https://github.com/IBM/processmining/tree/main/Datasets_usecases

⁹<https://fluxicon.com/>

Table 1: Summary statistics of the considered case studies.

Case Study	Completed Traces	Activities	Global Attributes	Mean Total Time
Bpi12	8,616	6	Requested_Amount	19,680 min
Bac	32,429	15	Closure_Type, Closure_Reason	23,615 min
Hospital	37,945	46	Triage_Color, Triage_Access, Patient_Age	188 min
Purchasing	608	21	-	115,015 min
Bpi17	30,276	8	Loan_Goal, Requested_Amount	30,240 min

Note that the Hospital process is not publicly available, due to legal constraints. This also means that a pre-trained LLM cannot have seen them in any form for building a-priori knowledge, although it has likely seen event logs of similar processes, which are supposedly beneficial for generating predictions. This ensures that our model has been tested on both publicly available and non-publicly available event logs, preventing eventual data leakage scenarios.

A summary of case studies, associated with the mean total time of each trace, is reported in Table 1.

5.2 Experimental Setup

The whole approach has been implemented in Python and the code is publicly available.¹⁰ For developing the Total Time function, any choice of LLM is valid, and we resorted to **Gemini 2.0 Flash Thinking**: a state-of-the-art LLM developed by Google DeepMind¹. The model is built on a multimodal architecture designed for advanced natural language understanding and generation. This specific model has been chosen due to its status as the most powerful and freely available LLM at the time of submission. As the development of LLMs progresses, we anticipate that the performances of upcoming models using our method will likely see improvements. For the purpose of setting a benchmark for prediction, we employed Catboost [21], a state-of-the-art model predictor based on machine learning on decision trees, which has been shown to surpass existing prediction frameworks [25].

Consistently with standard supervised learning practices, we divided the event log \mathcal{L} into training and test, \mathcal{L}^{comp} and \mathcal{L}^{run} , respectively. To extract the training log we compute the earliest time t_{split} such that 80% of the identifiers related to traces of \mathcal{L} are completed. This allows us to define \mathcal{L}^{comp} as the set of traces of \mathcal{L} completed at time t_{split} , and consequently, define \mathcal{L}^{run} as $\mathcal{L} \setminus \mathcal{L}^{comp}$. The traces of the test log \mathcal{L}^{run} are truncated to a set \mathcal{L}^{trunc} , namely the set of prefixes, that is obtained from \mathcal{L}^{run} by removing every event with a timestamp larger than t_{split} : \mathcal{L}^{trunc} only contains the events that occurred before time t_{split} . This procedure tries to mimic the reality at time t_{split} and it is in line with the principles introduced in [36]. The system is trained on \mathcal{L}^{comp} , the predictions are produced for \mathcal{L}^{trunc} and tested using its completed form, \mathcal{L}^{run} . Furthermore, to ensure a robust generalization across various process instances and a more balanced comparison with the LLM, we randomly picked 10% of traces and used them as a validation set \mathcal{L}^{valid} to apply a Cross-Validation approach to optimize the following parameters of Catboost: *learning_rate*, *tree_depth*, *training_iterations*.

¹⁰https://github.com/Pado123/gui_xrecs_presc_analytics

The traces were encoded using two different trace-to-instance encoding functions:

- ρ_{aggr} , the Aggregated History encoding function that proven to be the best option for Catboost (cf. [10]) and introduced in Section 3.
- ρ_{seq} , the Sequential encoding function introduced in Section 4.1.

The results have been reported in terms of **Mean Absolute Error (MAE)**, defined as the mean of the absolute differences between the true values and the predicted values.

This metric is widely used because it allows more interpretability, retaining the original unit of measurement, that for this work is **minutes**. Furthermore, Section 5.1 presents each case study along with the mean total time, providing an indication of the typical trace length in relation to the MAE.

As highlighted in Section 4.1, the proposed prompting technique includes a component that the process analyst must configure for each case study. This is optional and can be excluded if desired. Therefore, *we report results for both scenarios: when this part that has to be provided by process analyst is included and when it is omitted*.

In addition, due to the fact that the proposed encoding function ρ_{seq} produces a string as output, we were unable to test Catboost using this encoding.

To test the model’s ability to maintain accuracy with a reduced training set, we evaluated its performance by progressively shrinking the training data. Specifically, we reduced the number of traces to 100, 10, and 2. This experiment allowed us to assess the model’s robustness and accuracy when exposed to increasingly sparse datasets, simulating real-world scenarios where training data may be limited or incomplete. To mitigate the effects of statistical variation due to sampling, experiments have been repeated 20 times, and the reported results represent the mean and standard deviation of these runs, in order to highlight not only the value of MAE but also the possible uncertainty related to the statistical sampling.

5.3 Predictions Results

Table 2 reports the results of the experiments for the different case studies, along with the results from the Catboost banchmark, for which the relevant hyperparameters have been optimized. Results are reported when the training sets are composed of 2, 10, and 100 traces. The column *Model* indicates the predictive model: the benchmark as well as the LLM using the encodings based on aggregated history ρ_{aggr} and the whole trace ρ_{seq} when both the context is and is not used. Recall that, by context, we mean the context-specific background information and the description of global attributes (cf. Section 4.2). For completeness, the table also reports on the result of the benchmark when the whole training log has been used, whose size is reported in column *# Train Traces* in terms of the amount of traces. Mean and median values of the each test log are also have been used naïve predictors with the aim to provide a baseline, and results are reported. Recall that, for each process, predictive model, and size of the training set (e.g. 100 traces), the experiments have been repeated 20 times and taken different samples of the same size (e.g. 100 traces), as discussed in Section 5.2. The cell numbers represent the mean across the 20 experiments, with the standard deviation indicated after the \pm symbol, except in the case of the benchmark on the full log, where the sampling procedure was not performed.

Table 2: Accuracy results in terms of Mean Absolute Error (minutes). For each case study and reduced train set, the lower MAE are highlighted in bold.

Process	Predictive Model	Full log	100 Traces	10 Traces	2 Traces	Mean	Median	# Train Traces
Bpi12	Benchmark	6,846	9,394 \pm 114	10,811 \pm 111	11,594 \pm 3,819	9,373	9,374	6,892
	LLM ρ_{aggr} + Context	-	398 \pm 440	406 \pm 428	400 \pm 425			
	LLM ρ_{seq} + Context	-	7,258 \pm 878	8,450 \pm 1,402	7,549 \pm 2,049			
	LLM ρ_{aggr}	-	8,001 \pm 2,024	10,205 \pm 328	11,067 \pm 2,560			
	LLM ρ_{seq}	-	5,328 \pm 1,368	5,557 \pm 1,696	5,622 \pm 1,607			
Bac	Benchmark	2,647	6,393 \pm 387	8,181 \pm 1,633	9,634 \pm 1,929	6,172	5,998	25,901
	LLM ρ_{aggr} + Context	-	7,731 \pm 1,471	9,481 \pm 1,407	12,045 \pm 2,302			
	LLM ρ_{seq} + Context	-	2,510 \pm 471	6,066 \pm 920	6,741 \pm 2,692			
	LLM ρ_{aggr}	-	4,894 \pm 1,685	8,517 \pm 1,218	14,008 \pm 1,539			
	LLM ρ_{seq}	-	2,500 \pm 836	5,419 \pm 1,098	5,725 \pm 1,273			
Hospital	Benchmark	253	254 \pm 1	266 \pm 11	277 \pm 24	326	254	30,212
	LLM ρ_{aggr} + Context	-	433 \pm 418	406 \pm 428	400 \pm 425			
	LLM ρ_{seq} + Context	-	87 \pm 28	98 \pm 29	87 \pm 30			
	LLM ρ_{aggr}	-	401 \pm 416	410 \pm 431	389 \pm 389			
	LLM ρ_{seq}	-	86 \pm 29	92 \pm 23	91 \pm 32			
Purchasing	Benchmark	19,639	26,682 \pm 697	56,589 \pm 1,733	70,607 \pm 4,648	52,377	45,650	486
	LLM ρ_{aggr} + Context	-	46,902 \pm 8,299	50,774 \pm 12,891	56,277 \pm 4,205			
	LLM ρ_{seq} + Context	-	12,081 \pm 2,925	14,015 \pm 1,651	14,475 \pm 4,141			
	LLM ρ_{aggr}	-	37,899 \pm 3,657	42,225 \pm 3,237	56,127 \pm 6,740			
	LLM ρ_{seq}	-	11,767 \pm 2,339	12,593 \pm 2,722	12,071 \pm 3,466			
Bpi17	Benchmark	9,729	12,565 \pm 43	13,166 \pm 783	16,701 \pm 4,729	13,189	12,617	24,221
	LLM ρ_{aggr} + Context	-	13,638 \pm 3,302	12,732 \pm 3,189	14,425 \pm 4,011			
	LLM ρ_{seq} + Context	-	8,032 \pm 1,338	8,734 \pm 1,930	8,787 \pm 1,928			
	LLM ρ_{aggr}	-	11,185 \pm 3,005	11,043 \pm 2,736	14,722 \pm 2,958			
	LLM ρ_{seq}	-	6,931 \pm 2,605	6,993 \pm 2,491	7,656 \pm 3,107			

Our LLM-based framework for predictive process monitoring always outperforms the benchmark. Except for the Bpi12 case study, the ρ_{seq} encoding function outperforms the aggregated history ρ_{aggr} encoding function. This shows that, when Gemini is provided with more information for the predictive task, it is generally capable of discerning the information that is relevant for the specific case study, where an aggregated history imposes – generally wrongly – on the information that is relevant. This is far from being obvious: the feature engineering and the choice of the right aggregation is usually relevant for accuracy when predictive process monitoring is performed using Machine- and Deep-Learning predictors [28]. Remarkably, the context does not seem relevant in all case studies when the ρ_{seq} encoding function is employed, except for the Bpi12 process: the results with and without context are basically equivalent, namely the boundaries set by the standard deviations lie within the same intervals. This is likely due to the fact that the name of the global attributes and/or of the process activities allowed the LLM to derive the context from the pre-trained knowledge. Therefore, the results seem to suggest that the context is particularly relevant in non-conventional settings.

Interestingly, the Gemini predictive accuracy remains remarkably consistent whether it receives two or one hundred traces as input, with the exception of the Bac case study. The reason for this discrepancy in Bac is unclear. However, the general observation is that relying solely on two traces might not consistently yield a reliable predictive model. Also, our Gemini framework notably achieves higher accuracy with less than 100 traces, if compared with the Catboost benchmark trained on the whole event dataset. This su-

terior performance is observed across all case studies, though the results for Bac are less definitive.

5.4 Example of Reasoning to Achieve Predictions

The Gemini's output is not limited to returning the predicted value: Gemini also reports on the reasoning put in place to conclude the prediction. This is in fact a form of explanation of the predictions, although user studies would be necessary to assess whether they are indeed comprehensible by non-LLM and non-AI experts.

```

1  [[ ## reasoning ## ]]
2  The new case is for a 'New credit' application with a 'RequestedAmount' of 18000. The process instance
3  has executed activities 'W_Complete application' and 'W_Call after offers'.
4  The process is currently 'Running'.
5  To predict the total time, we will look at historical cases with similar characteristics.
6
7  1. Filter for similar Application Type and Requested Amount:
8      We focus on cases with 'ApplicationType' as 'New credit' and 'RequestedAmount'
9      in a range around 18000.0 (e.g., 10000.0 to 25000.0) to find comparable instances.
10
11 2. Identify relevant historical cases:
12      From the provided log, we select the cases that match 'New credit' application
13      type and have 'RequestedAmount' within the range of 10000.0 to 25000.0.
14      We list their total times:
15      10000.0: 9713, 10000.0: 18803, 11500.0: 4771, 12000.0: 18924, 12000.0: 8102,
16      13000.0: 6645, 13000.0: 21550, 14000.0: 9713, 15000.0: 8049, 15000.0: 17011
17
18 3. Calculate the median total time:
19      To get a robust estimate and reduce the impact of outliers,
20      we calculate the median
21      of these total times.
22      Sorted total times: 4771, 5299, 5975, 5975, 6645, 6645, 7432, 8049, 8102, 9713, 9713,
23      12991, 13056, 17011, 17434, 18803, 18924, 19826, 21550, 71762.
24      Since there are 20 values, the median is the average of the 10th and 11th sorted values,
25      which are 9713 and 9713. Median = (9713 + 9713) / 2 = 9713.
26
27 4. Predict total time:
28      We use the median total time of the similar historical cases as our prediction
29      for the new running case.
30
31 [[ ## answer ## ]]
32 9713
33 [[ ## completed ## ]]

```

Example 2: LLM's output and reasoning returned after the prompting technique example for a loan application process.

Nonetheless, it is a valuable addition to this paper to report and comment on an instance of the prediction's reasoning given as output by Gemini. To do so, Example 2 reports the output returned by the LLM for Example 1.

The output is returned at the end (see line 32 of Example 2). The reasoning of this example is as follows: Gemini has focused on the traces of the provided training log in which for which the 'ApplicationType' is of type '*New credit*' and the request amount is in the range of 10000 to 25000 (see lines 7-9). This leads to 20 traces (see lines 15-16), for which the median value is computed (see lines 24-25), which is used as predicted value (see explanations given at lines 28 and 29).

This reasoning certainly provides a valid explanation of the reasoning behind why this prediction is provided. However, this is only an example of the reasoning procedure performed by Gemini, and we observed that the output prompt can vary on the basis of the context and the number of training traces provided. Effort is necessary to homogenize the output prompt, and to steer towards a solution that is more explainable for process stakeholders. The latter still requires a research investigation that involves users and multiple analysis. We aim to move towards this direction as future work.

The given reasoning explains the prediction, but it is only an example of the reasoning procedure that is performed by Gemini, which changes with context and data. To improve explainability for process stakeholders, a more consistent and standardize output is necessary. This requires in-depth research involving user studies and varied analytical methods, a direction we intend to explore as future work.

6 Conclusions

This research introduces a novel Predictive Process Monitoring (PPM) framework that leverages on the capabilities of Large Language Models (LLMs) to overcome the challenge of limited data. Traditional PPM techniques rely on Machine- and Deep-Learning, which notoriously struggle with small datasets. LLMs have the capability to use the pre-trained knowledge to generalize even when small-scale event logs are provided.

With the premises above, this paper reports on our contribution to design a PPM framework that is based on LLMs. Two alternative prompts have been leveraged to encode the training sets and the contextual information. The framework has been implemented in Python, using Gemini as LLM, which has shown to be the best performing on different benchmarks among those freely available. Experiments have shown that our LLM-based framework enables making accurate predictions with small-scale event logs that are composed by less than 100 traces, with significant accuracy improvements with respect to standard methods from the PPM literature.

Section 5.4 has already reported that our future work will certainly focus on the output prompt generated by LLMs, aiming at its standardization and at its consequent use as explanation method. However, this requires user studies to evaluate alternatives and assess their benefits. While this research focuses on predictive process monitoring, a natural extension is to move toward recommender systems where not only are predictions given but also corrective actions are provided to recover the executions that are predicted to not achieve a satisfactory outcome. In this paper, the outcome is only defined in terms of duration of process executions, but KPIs can generally be of different natures (costs, customer satisfaction, etc.): a natural extension is indeed to extend our predictive framework towards KPIs that are others than execution duration. Finally, an interesting direction of future work is also related to investigating zero-prompting techniques, where the predictive model is provided with no specific examples, in line with the goal of this work to provide a more adaptable and scalable framework.

7 Acknowledgment

We acknowledge the support of the project “Future AI Research (FAIR) - Spoke 2 Integrative AI - Symbolic conditioning of Graph Generative Models (SymboliG)” funded by the European Union under the National Recovery and Resilience Plan (NRRP), Mission 4 Component 2 Investment 1.3 - Call for tender No. 341 of March 15, 2022 of Italian Ministry of University and Research – NextGenerationEU, Code PE0000013, Concession Decree No. 1555 of October 11, 2022 CUP C63C22000770006.

References

1. Ali, M.A., Dumas, M., Milani, F.: Enhancing the accuracy of predictors of activity sequences of business processes. In: *Research Challenges in Information Science*. Springer Nature Switzerland (2024)
2. Ashok, A., Williams, A.R., Marcotte, É., Zantedeschi, V., Subramanian, J., Riachi, R., Requeima, J., Lacoste, A., Rish, I., Chapados, N., et al.: Context is key: A benchmark for forecasting with essential textual information. In: *NeurIPS Workshop on Time Series in the Age of Large Models* (2024)
3. Buliga, A., Vazifehdoostirani, M., Genga, L., Lu, X., Dijkman, R.M., Francescomarino, C.D., Ghidini, C., Reijers, H.A.: Uncovering patterns for local explanations in outcome-based predictive process monitoring. In: *Business Process Management - 22nd International Conference, BPM 2024, Krakow, Poland, September 1-6, 2024, Proceedings*. Lecture Notes in Computer Science, vol. 14940, pp. 363–380. Springer (2024)
4. Ceravolo, P., Comuzzi, M., De Weerd, J., et al.: Predictive process monitoring: Concepts, challenges, and future research directions. *Process Science* **1**(2), 2 (2024)
5. Cunzolo, M.D., Ronzani, M., Aringhieri, R., Francescomarino, C.D., Ghidini, C., Guastalla, A., Sulis, E.: Robust solutions via optimisation and predictive process monitoring for the scheduling of the interventional radiology procedures. *Int. Trans. Oper. Res.* (2025)
6. Dani, V.S., Dees, M., Leopold, H., Busch, K., Beerepoot, I., van der Werf, J.M.E.M., Reijers, H.A.: Event log extraction for process mining using large language models. In: *Cooperative Information Systems - 30th International Conference, CoopIS 2024, Porto, Portugal*. Lecture Notes in Computer Science, Springer (2024)
7. Di Francescomarino, C., Ghidini, C.: *Predictive Process Monitoring*, pp. 320–346. Springer International Publishing, Cham (2022)
8. Elyasi, K.A., van der Aa, H., Stuckenschmidt, H.: Pgtnet: A process graph transformer network for remaining time prediction of business process instances. In: *Advanced Information Systems Engineering - 36th International Conference, CAiSE 2024, Limassol, Cyprus, 2024, Proceedings*. Lecture Notes in Computer Science, vol. 14663, pp. 124–140. Springer (2024)
9. Estrada-Torres, B., del Río-Ortega, A., Resinas, M.: Mapping the landscape: Exploring large language model applications in business process management. In: *Enterprise, Business-Process and Information Systems Modeling*. Springer Nature Switzerland (2024)
10. Galanti, R., de Leoni, M., Monaro, M., Navarín, N., Marazzi, A., Stasi, B.D., Maldera, S.: An explainable decision support system for predictive process analytics. *Eng. Appl. Artif. Intell.* **120**, 105904 (2023)
11. Gruver, N., Finzi, M., Qiu, S., Wilson, A.G.: Large language models are zero-shot time series forecasters. *Advances in Neural Information Processing Systems* **36**, 19622–19635 (2023)
12. Kim, J., Comuzzi, M., Dumas, M., Maggi, F.M., Teinemaa, I.: Encoding resource experience for predictive process monitoring. *Decision Support Systems* **153** (2022)

13. Kourani, H., Berti, A., Schuster, D., van der Aalst, W.M.P.: Process modeling with large language models. In: Enterprise, Business-Process and Information Systems Modeling. pp. 229–244. Springer Nature Switzerland, Cham (2024)
14. Kubrak, K., Botchorishvili, L., Milani, F., Nolte, A., Dumas, M.: Explanatory capabilities of large language models in prescriptive process monitoring. In: Business Process Management - 22nd International Conference, BPM 2024, Krakow , 2024, Proceedings. Lecture Notes in Computer Science, vol. 14940, pp. 403–420. Springer (2024)
15. Kuratov, Y., Bulatov, A., Anokhin, P., Rodkin, I., Sorokin, D., Sorokin, A., Burtsev, M.: Babilong: Testing the limits of llms with long context reasoning-in-a-haystack. Advances in Neural Information Processing Systems **37**, 106519–106554 (2024)
16. Lashkevich, K., Milani, F., Avramenko, M., Dumas, M.: Llm-assisted optimization of waiting time in business processes: A prompting method. In: Business Process Management - 22nd International Conference, BPM 2024, Krakow , 2024, Proceedings. Lecture Notes in Computer Science, vol. 14940, pp. 474–492. Springer (2024)
17. Li, T., Zhang, G., Do, Q.D., Yue, X., Chen, W.: Long-context LLMs struggle with long in-context learning. Transactions on Machine Learning Research (2024)
18. Oyamada, R., Tavares, G., Barbon Junior, S., Ceravolo, P.: Enhancing Predictive Process Monitoring with Time-Related Feature Engineering. Springer Nature Switzerland (2024)
19. Padella, A., de Leoni, M., Dogan, O., Galanti, R.: Explainable process prescriptive analytics. In: 2022 4th International Conference on Process Mining (ICPM). pp. 16–23 (2022)
20. Pasquadibisceglie, V., Appice, A., Malerba, D.: Lupin: A llm approach for activity suffix prediction in business process event logs. In: 2024 6th International Conference on Process Mining (ICPM). pp. 1–8 (2024)
21. Prokhorenkova, L.O., Gusev, G., Vorobev, A., Dorogush, A.V., Gulin, A.: Catboost: unbiased boosting with categorical features. In: NeurIPS (2018)
22. Rebmann, A., Schmidt, F.D., Glavas, G., van der Aa, H.: Evaluating the ability of llms to solve semantics-aware process mining tasks. In: 6th International Conference on Process Mining, ICPM 2024, Kgs. Lyngby, Denmark, October 14–18, 2024. pp. 9–16. IEEE (2024)
23. Requeima, J., Bronskill, J., Choi, D., Turner, R.E., Duvenaud, D.K.: LLM processes: Numerical predictive distributions conditioned on natural language. In: Advances in Neural Information Processing Systems (2024)
24. Rizzi, W., Simonetto, L., Di Francescomarino, C., Ghidini, C., Kasekamp, T., Maggi, F.M.: Nirdizati 2.0: New features and redesigned backend. In: Proceedings of the Dissertation Award, Doctoral Consortium, and Demonstration Track at BPM 2019. vol. 4220, pp. 154–158. ceur-ws.org (2019)
25. Shoush, M., Dumas, M.: White box specification of intervention policies for prescriptive process monitoring. Data Knowl. Eng. **155**, 102379 (2025)
26. Snoeck, M., Verbruggen, C., Smedt, J.D., Weerdt, J.D.: Supporting data-aware processes with MERODE. Softw. Syst. Model. **22**(6), 1779–1802 (2023)
27. Stevens, A., De Smedt, J., Peeperkorn, J., De Weerdt, J.: Assessing the robustness in predictive process monitoring through adversarial attacks. In: 2022 4th International Conference on Process Mining (ICPM). pp. 56–63 (2022)
28. Tavares, G.M., Oyamada, R.S., Barbon, S., Ceravolo, P.: Trace encoding in process mining: A survey and benchmarking. Eng. Appl. Artif. Intell. **126**(Part D), 107028 (2023)
29. Tax, N., Verenich, I., La Rosa, M., Dumas, M.: Predictive business process monitoring with lstm neural networks. In: Dubois, E., Pohl, K. (eds.) Advanced Information Systems Engineering. Springer International Publishing (2017)
30. Taymouri, F., Rosa, M.L., Erfani, S., Bozorgi, Z.D., Verenich, I.: Predictive business process monitoring via generative adversarial nets: The case of next event prediction. In: Fahland, D., Ghidini, C., Becker, J., Dumas, M. (eds.) Business Process Management. pp. 237–256. Springer International Publishing, Cham (2020)

31. Teinemaa, I., Dumas, M., La Rosa, M., Maggi, F.: Outcome-oriented predictive process monitoring: Review and benchmark. *ACM Transactions on Knowledge Discovery from Data* **13** (07 2017)
32. Vacareanu, R., Negru, V.A., Suciu, V., Surdeanu, M.: From words to numbers: Your large language model is secretly a capable regressor when given in-context examples. In: *First Conference on Language Modeling* (2024)
33. van der Aa, H., Rebmann, A., Leopold, H.: Natural language-based detection of semantic execution anomalies in event logs. *Information Systems* **102**, 101824 (2021)
34. Vazifehdoostirani, M., Genga, L., Dijkman, R.: Encoding high-level control-flow construct information for process outcome prediction. In: *2022 4th International Conference on Process Mining (ICPM)*. pp. 48–55 (2022)
35. Verenich, I., Dumas, M., La Rosa, M., Maggi, F., Teinemaa, I.: Survey and cross-benchmark comparison of remaining time prediction methods in business process monitoring. *ACM Transactions on Intelligent Systems and Technology* **10** (07 2019)
36. Weytjens, H., Weerdt, J.D.: Creating unbiased public benchmark datasets with data leakage prevention for predictive process monitoring. In: *Business Process Management Workshops - BPM 2021 International Workshops, Rome, Italy, 2021, Revised Selected Papers. Lecture Notes in Business Information Processing*, vol. 436, pp. 18–29. Springer (2021)
37. Zimmermann, L., Zerbato, F., Weber, B.: What makes life for process mining analysts difficult? a reflection of challenges. *Software and Systems Modeling* pp. 1–29 (11 2023)