

Large Language Models can accomplish Business Process Management Tasks

Michael Grohs*, Luka Abb*, Nourhan Elsayed*, and Jana-Rebecca Rehse

University of Mannheim

{michael.grohs,luka.abb,elsayed,rehse}@uni-mannheim.de

Abstract. Business Process Management (BPM) aims to improve organizational activities and their outcomes by managing the underlying processes. To achieve this, it is often necessary to consider information from various sources, including unstructured textual documents. Therefore, researchers have developed several BPM-specific solutions that extract information from textual documents using Natural Language Processing techniques. These solutions are specific to their respective tasks and cannot accomplish multiple process-related problems as a general-purpose instrument. However, in light of the recent emergence of Large Language Models (LLMs) with remarkable reasoning capabilities, such a general-purpose instrument with multiple applications now appears attainable. In this paper, we illustrate how LLMs can accomplish text-related BPM tasks by applying a specific LLM to three exemplary tasks: mining imperative process models from textual descriptions, mining declarative process models from textual descriptions, and assessing the suitability of process tasks from textual descriptions for robotic process automation. We show that, without extensive configuration or prompt engineering, LLMs perform comparably to or better than existing solutions and discuss implications for future BPM research as well as practical usage.

Keywords: Business Process Management · Natural Language Processing · Large Language Models · ChatGPT

1 Introduction

The objective of Business Process Management (BPM) is to understand and supervise the execution of work within an organization. This ensures consistent outcomes and allows for the identification of improvement opportunities [6]. To accomplish this, BPM researchers and practitioners make use of diverse sources of information pertaining to business processes. These sources range from well-structured process models and event logs to unstructured textual documents [18]. In the past decade, BPM researchers have increasingly turned to Natural Language Processing (NLP) techniques to automatically extract process-related information from the abundant textual data found in real-world organizations.

* Equal contribution

Many existing approaches utilize textual data for a wide range of BPM tasks. Examples of such tasks include the mining of imperative or declarative process models from textual process descriptions [8, 19], process redesign for classifying end-user feedback [11], identifying suitable tasks for robotic process automation (RPA) in textual process descriptions [10], assessing process complexity based on textual data [16], or extracting semantic process information from natural language [13]. Although a few approaches also incorporate machine learning methods, the majority rely on extensive rule sets.

Each existing approach is designed for a specific purpose, meaning that it can only be applied to one specific task. A versatile general-purpose model that comprehends process-related text and seamlessly integrates it into various BPM tasks does not yet exist. However, the recent emergence of pre-trained Large Language Models (LLMs), which have demonstrated remarkable reasoning abilities across diverse domains and tasks [17], offers promising prospects for developing such a system. Already, multiple research groups are actively exploring the potential of these models in the BPM field, for example by analyzing which opportunities and challenges LLMs pose for the individual stages of the BPM lifecycle [20], how LLMs input should look like such that the output supports BPM [5], or whether conversational process modeling is possible [9].

These recent publications and pre-prints mostly illustrate the potential and difficulties of LLMs on a high level, but they do not showcase concrete applications. In this paper, we take a more application-oriented approach by investigating whether an LLM can accomplish three text-related BPM tasks: (1) mining imperative process models from textual descriptions, (2) mining declarative process models from textual descriptions, and (3) assessing the suitability of process tasks for RPA from textual descriptions. We selected these tasks because they are practically relevant and have previously been addressed in research. We evaluate how well the LLM can perform these tasks by benchmarking them against existing approaches that were specifically developed for the respective task. Based on the results, we discuss implications for future research in the field of BPM and illustrate how LLMs can support practitioners in their daily work.

The paper is structured as follows: In Sect. 2, we introduce the general solution approach that we followed for all three tasks. The task-specific applications and results are described in Sect. 4, Sect. 3, and Sect. 5, respectively. Section 6 discusses the future usage of LLMs in practice as well as implications for future research, before we conclude the paper in Sect. 7.

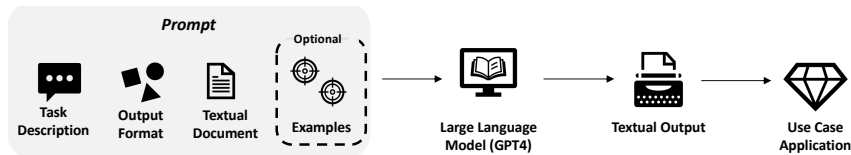


Fig. 1: Overview of our Approach

2 Approach

In this paper, we illustrate how LLMs can be utilized for three BPM tasks that require textual documents as input. For all tasks, we follow the same approach, illustrated in Fig. 1. We start by assembling a prompt with the following parts:

1. A general description of the BPM task that is to be accomplished.
2. A specification of a particular output format that the LLM should adhere to. This ensures that the generated text output has a certain level of consistency and that results are sufficiently standardized so that they can be further processed by, for example, parsing algorithms.
3. The natural language text that we want to abstract information from, e.g., a textual process description
4. Optionally, if suitable for a given task, few input-output pairs as examples

The complete prompt is then entered into the current state-of-the-art instruction-following LLM, ChatGPT with GPT4 backend [12] (henceforth referred to as GPT4). The textual output of GPT4 (i.e., the response to the prompt) is then evaluated with respect to its utility in solving the respective task and benchmarked against an existing approach. All parts of the prompt have not been specifically engineered but rather included such that the output is actually solving the tasks. The prompts were not optimized with respect to any metric.

In all applications, we provide the model with several prompts in order to check input robustness (i.e., how prompts from different authors influence the results) and output robustness (i.e., how the results change for different tries of the same prompt). By this, we aim to analyze whether GPT4 is able to accomplish specific tasks sufficiently well to be used by a diverse group of people and whether the results remain consistent despite the inherent randomness of the model output. For each task, we start with an ‘‘original’’ prompt written by one of the authors of this paper and enter this prompt three times (Tries 1 to 3). Two more prompts are then created by two other authors, who are given a general description of the task to be accomplished and the exact output format that they should specify, but who have not seen the original prompt. Finally, where appropriate, we also enter the original prompt without examples to evaluate the effect those have on the result. Each prompt is entered in a separate conversation window in the GPT4 web interface so that the model cannot draw on previous prompts as context.

All prompts, responses, and detailed evaluation results are available online¹.

3 Mining BPMN process models from natural language descriptions

3.1 Motivation

Process models are the predominant tool for representing organizational activities and are often the starting point for process analyses [19]. Constructing

¹ <https://gitlab.uni-mannheim.de/jpmac/llms-in-bpm>

such models requires knowledge of the process and proficiency in the creation of formal models [8]. However, the actors with process knowledge commonly are not experienced process modelers [8]. Therefore, modeling procedures can be very time-consuming and error-prone [15]. This holds true even though detailed textual descriptions of process requirements are often available in the form of policies, guidelines, or e-mail conversations, which can be considered relevant sources of information [8]. Approaches that extract process models from natural language can speed up the modeling and also enable managers to frequently update their process models without requiring extensive modeling experience.

A rule-based approach to extract Business Process Model and Notation (BPMN) process models from textual process descriptions was first proposed in [8]. This remains the only generally-applicable, end-to-end technique able to produce a full imperative process model from text input, though several other publications with a more narrow scope or a focus on mining partial models exist (see [2] for a short review). There are also papers that investigate the ability of LLMs to extract process entities and relations from textual descriptions [3, 9]. Though their approaches have some similarities to ours, neither ends up producing an actual process model from the text.

3.2 Evaluation

Following Fig. 1, we ask GPT4 to create a BPMN model for a process described in text. At the time of writing, the web interface version of GPT4 has a token output limit that prevents it from generating sequences at the length that would be required to generate BPMN models as XML files. We, therefore, prompt it to produce a model in a pre-specified intermediary notation as an output format that includes the main elements of BPMN and is straightforward to parse into a proper model representation. The template we provided in the prompt represents task nodes as natural language words, arcs between model elements as arrows (\rightarrow), and exclusive and parallel gateways as **XOR** and **AND**, respectively. We also specify that outgoing arcs of exclusive gateways can be labeled to represent decision criteria, e.g., **XOR (Proposal accepted) \rightarrow Task1**. Finally, we ask the model to provide an actor-to-activity mapping that can be used to construct lanes, in the format **actor: [activity1, ...]**. Other elements (e.g., messages) are not included. We also do not provide example pairs of text and corresponding full or partial models to the LLM to avoid bias towards a certain modeling style.

Figure 2 shows an example of a textual description of a computer repair process, an excerpt of the response that GPT4 gave when presented with this description, and a visualization of the derived BPMN model. The generated model accurately represents the process described in the text. It could, however, be made slightly simpler by combining the two separate *Test System Functionality* activities and the subsequent exclusive gateways into one each.

For our evaluation, we use six process descriptions from [7] (1.1 - 1.4, 2.1, and 2.2). We selected these with the goal of applying our technique to a mix of short and simple as well as longer and more sophisticated textual descriptions. As ground truth, we use the annotations provided for these descriptions in the

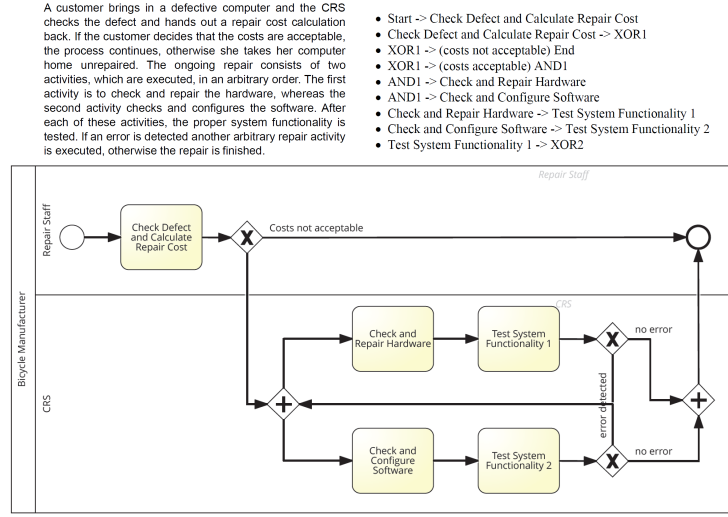


Fig. 2: Example of a textual process description (top left), an excerpt of the generated LLM response (top right, from Prompt 1 Try 1), and visualization of the corresponding BPMN diagram (bottom).

PET dataset [1]. Specifically, we evaluate the output of the LLM with regard to how many of the relations described in the textual description are correctly identified (i.e., recall). Note that this allows us to simultaneously evaluate how many entities (task names and actors) are correctly identified since a relation that involves an unknown entity will be counted as not identified. We do not evaluate the models with regard to how many superfluous entities or relations they produce (i.e., precision) as that would raise several conceptual questions that require answers (e.g., how to treat a task that is correctly identified but in the wrong position), which would go beyond the intended scope of this paper.

We further restrict our evaluation to *flow* and *actor performer* relations, i.e., those that are present in the intermediary notation we provide in the prompts. Since the ground truth annotation applies only to the textual descriptions, we manually establish a mapping between the entities identified in the dataset and the ones produced by GPT4. In some cases, the relations produced by the LLM do not exactly match the ground truth (e.g., **Write Report** and **Send Report** are combined to **Write and Send Report**). For these, we follow the same approach as [3], i.e., we evaluate them on a case-by-case basis and count them as correct if they are semantically correct. As a benchmark, we use the process models produced by [7], applying the same evaluation criteria as described above.

The results of our evaluation are shown in Tab. 1, subdivided by the evaluation of output robustness (OR) and input robustness (IR). Overall, regarding the proportion of relations (and entities) that are correctly extracted from the textual process description, the models generated by GPT4 are comparable to

Table 1: Recall for the Text-to-BPMN Task

		Text 1.1	Text 1.2	Text 1.3	Text 1.4	Text 2.1	Text 2.2	Overall
OR	Prompt 1 Try 1	0.42	0.58	0.46	0.50	0.57	0.45	0.50
	Prompt 1 Try 2	0.54	0.58	0.38	0.70	0.61	0.42	0.54
	Prompt 1 Try 3	0.54	0.58	0.50	0.60	0.53	0.53	0.54
IR	Other Author (1)	0.54	0.47	0.54	0.50	0.47	0.34	0.48
	Other Author (2)	0.46	0.42	0.35	0.47	0.43	0.39	0.42
Benchmark [19]		0.54	0.47	0.58	0.55	0.55	0.66	0.56

the ones produced by [8]. Note that the absolute numbers reported should be interpreted with caution, because the PET ground truth is very fine-granular and we weigh all relation types equally, so that, for example, a single missing exclusive gateway (with the gateway itself, two decision criteria on the outgoing arcs, and two subsequent activities) would be counted as five non-identified relations. Consequently, a recall value of 0.5 should not be understood to indicate that the model only includes half of the relevant process behavior described in the text. Furthermore, the models generated by GPT4 are very precise in the sense that they tend to include a minimal (often insufficient) set of tasks, whereas the rule-based approach of [8] tends to produce models with several superfluous activities (e.g., **Begin Process** following a start event). Since our evaluation does not include a notion of false positive relations, it could be argued that we somewhat underestimate the quality of the LLM output relative to the benchmark.

Overall, an LLM-based text-to-BPMN technique produces reasonably good results. The model also produces consistent answers in the same intermediary notation when provided with the exact same description of the target template, so parsing its output into XML is possible. With prompt fine-tuning, and especially with subsequent prompting that asks the model to fix common issues, it is not unfeasible to create a reliable text-to-BPMN pipeline based on an LLM.

4 Mining declarative process models from natural language descriptions

4.1 Motivation

Not all business processes can be adequately captured by imperative modeling notations such as BPMN. For instance, knowledge-intensive processes have execution orders that cannot always be fully specified in advance [19]. These are better modeled using declarative process models, i.e., a set of formal constraints that do not rely on an explicit definition of allowed behavior [4]. They provide a flexible way of modeling processes, especially suitable in complex settings [4].

An approach that extracts declarative process models from natural language has been proposed in [19]. It uses the common declarative modeling language *Declare*, which is based on constraint templates grounded in Linear Temporal Logic (LTL) [4]. By applying rule-based NLP techniques to sentences, the

approach in [19] generates declarative constraints for five LTL templates: precedence, response, succession, initialization (init), and end. $Precedence(A, B)$ (formal as $\text{NOT}(B) \text{ U } A$) means that activity B should only occur after activity A. $Response(A, B)$ (formal as $A \rightarrow B$) means that B must follow whenever A occurs. $Succession(A, B)$ is the combination of $Precedence(A, B)$ and $Response(A, B)$. $Init(A)$ (formal as $\text{START} \rightarrow A$) prescribes that all process instances must start with A and $End(A)$ (formal as $\text{END} \rightarrow A$) indicates that they must end with A.

4.2 Evaluation

In our experiment, we recreate the set-up from [19], applying GPT4 on the same five LTL templates and 104 test sentences. Following Fig. 1, we create a prompt that asks GPT4 to create LTL formulas in the form of precedence, response, succession, init, and end. For each template, we provide the output format and an example. As a result, the LLM outputs one or more discovered constraints in the format prescribed by the prompt, as shown in the exemplary excerpt of the output in Tab. 2. This output can then be compiled and translated into declarative modeling languages like *Declare*.

In addition to the three identical prompts for output robustness, we use two other formulations by different authors and, as we use examples in the original prompt, also one prompt without examples for input robustness. Table 3 displays precision (Prec.), recall (Rec.), and F1-score (F1) as used by [19] for each of the five LTL templates of the six different prompts compared to the benchmark.² We only consider syntactically correct classifications as true positives.

Except for the response template, GPT4 outperforms the benchmark and has a high precision value of close to 1. Further, we see that precision does not vary significantly with respect to output robustness for all LTL templates. With respect to recall, we see lower values for precedence. This is because many precedence constraints are misclassified as a response, which also explains the lower precision for this template. For succession and end, we see a high variation in the recall. This is due to a few constraints of these types in the 104 sentences, meaning that few misclassifications have a high impact. With respect to input robustness, the evaluation metrics are worse if no examples for the LTL templates

² The corresponding confusion matrices can be found in our repository.

Table 2: Exemplary Output of GPT4 for the Text-to-LTL Task

Sentence Input	GPT4 Output	LTL-Template
A claim should be created before it can be approved.	NOT(approve claim) U create claim	Precedence
The process begins with the booking of the ticket.	START -> book ticket	Init
Every provided laundry service must be billed.	provide laundry service -> F(bill)	Response

Table 3: Precision, Recall, and F1-Score for the Text-to-LTL Task

			Precedence	Response	Succession	Init	End	Overall
Output Robustness	Prompt 1 Try 1	Prec.	0.96	0.68	1.00	1.00	1.00	0.84
		Rec.	0.53	0.96	1.00	0.82	0.17	0.76
		F1	0.68	0.80	1.00	0.90	0.29	0.79
	Prompt 1 Try 2	Prec.	0.91	0.65	1.00	1.00	1.00	0.80
		Rec.	0.57	0.75	0.33	1.00	0.50	0.68
		F1	0.70	0.70	0.50	1.00	0.67	0.74
	Prompt 1 Try 3	Prec.	0.94	0.68	1.00	1.00	1.00	0.83
		Rec.	0.61	0.88	0.25	1.00	0.60	0.76
		F1	0.74	0.77	0.40	1.00	0.75	0.79
Input Robustness	No Examples	Prec.	0.57	0.51	0.33	0.83	1.00	0.58
		Rec.	0.08	0.79	0.67	0.91	0.50	0.49
		F1	0.14	0.62	0.44	0.87	0.67	0.53
	Other Author (1)	Prec.	0.94	0.72	1.00	1.00	1.00	0.86
		Rec.	0.65	0.88	1.00	0.82	0.67	0.77
		F1	0.77	0.79	1.00	0.90	0.80	0.81
	Other Author (2)	Prec.	0.91	0.71	0.75	0.90	1.00	0.83
		Rec.	0.61	0.77	1.00	0.82	0.83	0.72
		F1	0.73	0.74	0.86	0.86	0.91	0.77
Benchmark [19]	Prec.	0.78	0.8	0.68	0.75	0.88	0.77	
	Rec.	0.71	0.77	0.68	0.82	0.88	0.72	
	F1	0.74	0.75	0.68	0.78	0.88	0.74	

are provided. This is especially visible for the precedence template. In contrast to that, different formulations from other authors do not have a significant impact on the metrics. Rather, stability across different prompts is visible.

The F1-score shows that all prompts with examples for the LTL templates yield equal or higher scores than the benchmark. This illustrates that GPT4 outperforms the specific approach from [19] if it is provided with proper examples. This is an important finding as it indicates that prompts yield different results based on their fit to the task. Further, for tasks like this with short input text to be classified and a few classification targets, we recommend that the prompt should include examples. It should be noted that other prompts for example with additional information or the repetition of instructions could yield even better results. Further, the output of GPT4 has to be parsed into declarative process models using for example *Declare* to allow complete usage. This is possible in an automatic manner given the consistent output format for all 104 sentences.

5 Assessing RPA suitability of process tasks from natural language descriptions

5.1 Motivation

RPA is a technology that aims to automate routine and repetitive tasks in business environments. To do so, software robots that work on the user interface

of software systems are developed to perform these tasks the same way human actors would do, thus increasing operational efficiency [14].

Various process information can be used to identify tasks that are suitable for RPA. This includes textual process descriptions, which are commonly used to document processes [18]. The approach proposed in [10] identifies suitable tasks for RPA by measuring the degree of the automation of process tasks using supervised machine learning techniques from the textual descriptions of business processes. From this textual data, the approach classifies the process tasks into manual, automated, or user tasks. Manual tasks are the tasks performed by a human actor without any use of an information system, user tasks consist of humans interacting with an information system, and automated tasks are performed automatically on an information system without any human involvement. Tasks classified as user tasks are suitable RPA candidate tasks as they can be automated by replicating human interactions by means of RPA agents. This increases the efficiency of identifying suitable RPA tasks in comparison to a manual analysis that takes a long time and effort, especially if there exists a large number of such documents or a large number of processes to be analyzed [10].

5.2 Evaluation

Following the approach from Fig. 1, GPT4 is used to replicate the experiment of [10]. The task is described in the prompt by asking the LLM to classify process tasks into one of three output formats: manual, user, or automated task. Possible features that might affect the task classifications (e.g., verb feature, object feature, resource type (human/non-human), and IT domain) are included in the task description. The output format as well as an example of tasks' classification for tasks of a given process description is also provided. We use the same dataset as [10], consisting of 33 textual process descriptions obtained from [7]. These descriptions consist of 424 process tasks to be classified. See Tab. 4 for an example of an input process description and the output generated by GPT4.

We did three identical prompts, we use two other prompts by different authors with an example in each. We also did one prompt without an example. Table 5 displays precision (Prec.), recall (Rec.), and F1-score (F1) for each of the six prompts compared to the benchmark from [10]. For the overall results, we apply the same micro-averaging approach as the benchmark, i.e., the number of tasks belonging to a class was used to weigh the respective precision and recall values.

Table 4: Exemplary Output of GPT4 for the RPA Classification Task

Task Input	GPT4 Output
register a claim performed by claims officer	User task
examine a claim performed by claims officer	Manual task
write a settlement recommendation performed by claims officer	Manual task
send the claim back to the claims performed by claims officer	User task

Table 5: Precision, Recall, and F1-Score for the RPA Task

			Manual	User	Automated	Overall
Output Robustness	Prompt 1 Try 1	Prec.	0.68	0.88	0.15	0.79
		Rec.	0.83	0.6	0.75	0.67
		F1	0.75	0.74	0.45	0.73
	Prompt 1 Try 2	Prec.	0.65	0.84	0.73	0.78
		Rec.	0.69	0.83	0.69	0.78
		F1	0.67	0.84	0.71	0.78
	Prompt 1 Try 3	Prec.	0.84	0.84	0.32	0.82
		Rec.	0.65	0.83	0.93	0.78
		F1	0.75	0.84	0.63	0.8
Input Robustness	No Examples	Prec.	0.85	0.77	0.34	0.78
		Rec.	0.42	0.88	0.88	0.74
		F1	0.64	0.83	0.61	0.76
	Other Author (2)	Prec.	0.44	0.71	0.29	0.61
		Rec.	0.42	0.74	0.13	0.62
		F1	0.43	0.73	0.21	0.62
	Other Author (2)	Prec.	0.5	0.87	0.36	0.74
		Rec.	0.82	0.55	0.8	0.64
		F1	0.66	0.71	0.58	0.69
Benchmark [10]		Prec.	0.81	0.8	0.92	0.81
		Rec.	0.9	0.7	0.52	0.8
		F1	0.85	0.75	0.66	0.81

GPT4 outperforms the benchmark for 4 out of 6 prompts for user tasks. For the automated tasks, precision results are below the benchmark because many tasks were classified by GPT4 as automated although they are not. However, the recall for this class outperforms the benchmark in almost all the prompts. For the F1-score, it is similar to the benchmark for the classes except for the user class where the F1-score results were higher than the benchmark. Overall, as indicated by the F1-score, GPT4 performs similarly to the benchmark for all six prompts. We also saw the performance of GPT4 deteriorate over time. We suspect that this is caused by the limited context window of GPT4, combined with the large number of tasks to be classified (424). In such cases, reminding the LLM of the task description between inputs could yield better results.

6 Discussion

After illustrating that out-of-the-box GPT4 performs similarly or even better than specialized approaches for our three exemplary tasks, we now want to discuss the usage of LLMs in practice and provide guidelines for users.

Prompt Recommendations. In our experiments, we found that including different contents in the prompt increase the performance of GPT4. For example, the output should be clearly defined instigating the task. Further, for the text-to-LTL task, examples led to better results. We can therefore recommend specifying

the output format and to try using examples if feasible. In general, different prompts should be used and compared to maximize the benefits of using GPT4. **Non-deterministic output.** In order to produce more natural-sounding text, generative LLMs typically have *temperature* parameter that adds some variability to the output. Because of this, responses given by GPT4 may change even if the input remains constant. At the same time, if the input is varied slightly (e.g., by phrasing the same instruction in a different way), the model may make significant alterations to its response. In our experiments, we attempted to account for this by establishing a certain level of input and output consistency. We found that, although results are overall relatively consistent, there is still considerable variation in how well each response reflects individual aspects of the provided text, for example, whether a particular task has been correctly identified and categorized. We, therefore, argue that future research into the behavior of LLMs and their reaction to different inputs is needed. In particular, the non-deterministic nature of LLM’s output has implications for evaluation design: in our opinion, a basic sensitivity analysis as applied in this paper is always required in order to perform a meaningful evaluation of performance.

File Generation. When using it in practice, as illustrated with the three tasks, GPT4 does not generate files but rather text. Therefore, in order to use it in the first two exemplary tasks, further translation into formalized languages was necessary. This can be done via a compiler that generates *Declare* constraints or BPMN models based on the output. Nevertheless, it poses a limitation of current LLMs, especially considering output variability. It should be noted that this limitation is specific to present-day LLMs such as GPT4, which are not capable of file generation and may be overcome by future iterations of the models.

7 Conclusion

In this paper, we developed and applied an approach that utilizes the LLM GPT4 for diverse BPM tasks. The approach itself is simple and leverages the capabilities of GPT4 by instructing it to accomplish the task at hand. We selected three BPM tasks to illustrate that GPT4 is indeed able to accomplish them: mining imperative process models from the textual description, mining declarative process models from the textual description, and assessing RPA suitability of process tasks from textual descriptions. For all three tasks, GPT4 performs similarly to or better than the benchmark, i.e., specific applications for the respective task. We analyzed the input and output robustness of the approach and found that the output is relatively insensitive to different executions of the same prompt, even if different authors formulated them. Further, we found that some prompts should include examples to help the LLM. Future research could assess whether LLMs are also applicable to other tasks from different phases of the BPM lifecycle. All in all, this paper illustrates and evaluates three practical applications of GPT4 and provides implications for future research and usage.

References

1. Bellan, P., van der Aa, H., Dragoni, M., Ghidini, C., Ponzetto, S.P.: Pet: An annotated dataset for process extraction from natural language text tasks. In: BPM Workshops. pp. 315–321. Springer (2023)
2. Bellan, P., Dragoni, M., Ghidini, C.: A qualitative analysis of the state of the art in process extraction from text. In: DP@AI*IA (2020)
3. Bellan, P., Dragoni, M., Ghidini, C.: Extracting business process entities and relations from text using pre-trained language models and in-context learning. In: Enterprise Design, Operations, and Computing. pp. 182–199. Springer (2022)
4. Burattin, A., Maggi, F.M., Sperduti, A.: Conformance checking based on multi-perspective declarative process models. *Expert Syst Appl* **65**, 194–211 (2016)
5. Busch, K., Rochlitzer, A., Sola, D., Leopold, H.: Just tell me: Prompt engineering in business process management. preprint arXiv:2304.07183 (2023)
6. Dumas, M., La Rosa, M., Mendling, J., Reijers, H.: Introduction to Business Process Management. In: Fundamentals of Business Process Management. Springer (2018)
7. Friedrich, F.: Automated generation of business process models from natural language input (Master thesis), <https://frapu.de/pdf/friedrich2010.pdf>
8. Friedrich, F., Mendling, J., Puhlmann, F.: Process Model Generation from Natural Language Text. In: CAiSE. pp. 482–496. Springer (2011)
9. Klievtsova, N., Benzin, J.V., Kampik, T., Mangler, J., Rinderle-Ma, S.: Conversational process modelling: State of the art, applications, and implications in practice. preprint arXiv:2304.11065 (2023)
10. Leopold, H., van der Aa, H., Reijers, H.A.: Identifying candidate tasks for robotic process automation in textual process descriptions. In: BPMDS. pp. 67–81. Springer (2018)
11. Mustansir, A., Shahzad, K., Malik, M.K.: Towards automatic business process redesign: an NLP based approach to extract redesign suggestions. *Autom Softw Eng* **29**, 1–24 (2022)
12. OpenAI: GPT-4 Technical Report. preprint arXiv:2304.04309 (2023)
13. Rebmann, A., van der Aa, H.: Extracting semantic process information from the natural language in event logs. In: CAiSE. pp. 57–74. Springer (2021)
14. Reddy, K.N., Harichandana, U., Alekhya, T., Rajesh, S.: A study of robotic process automation among artificial intelligence. *Int J Sci Res* **9**(2), 392–397 (2019)
15. Reijers, H.A., Limam, S., van der Aalst, W.: Product-Based Workflow Design. *JMIS* **20**(1), 229–262 (2003)
16. Rizun, N., Revina, A., Meister, V.G.: Assessing business process complexity based on textual data: Evidence from ITIL IT ticket processing. *BPMJ* **27**(7), 1966–1998 (2021)
17. Teubner, T., Flath, C.M., Weinhardt, C., van der Aalst, W., Hinz, O.: Welcome to the era of ChatGPT et al. The prospects of large language models. *BISE* **65**, 95–101 (2023)
18. van der Aa, H., Carmona, J., Leopold, H., Mendling, J., Padró, L.: Challenges and Opportunities of Applying Natural Language Processing in Business Process Management. In: COLING. pp. 2791–2801 (2018)
19. van der Aa, H., Di Ciccio, C., Leopold, H., Reijers, H.A.: Extracting declarative process models from natural language. In: CAiSE. pp. 365–382. Springer (2019)
20. Vidgof, M., Bachhofner, S., Mendling, J.: Large language models for business process management: Opportunities and challenges. preprint arXiv:2304.04309 (2023)