Local Large Language Models for Business Process Modeling

Kaan Apaydin[1](✉) and Yorck Zisgen[2]

[1] Department of Computer Science, Kiel University, Kiel, Germany
kap@informatik.uni-kiel.de
[2] University of Bayreuth, Bayreuth, Germany
yorck.zisgen@uni-bayreuth.de

Abstract. Large language models (LLMs) are capable of efficiently understanding natural language by processing large volumes of text data. Natural language is also used in process descriptions, thus LLMs appear to be a suitable candidate to significantly improve business process modeling. Although plenty of third-party LLMs exist, they raise the risk of privacy disclosure, untrustworthiness, and generalizability of the results. This paper proposes a pipeline to use a local and fine-tuned LLM that expects a textual process description as input and finally generates a visual process tree representation. We instantiate our pipeline with Llama3 8B and fine-tune the LLM with a training set of 120 self-generated examples. Initial evaluation results of our LLM-based approach for automated business process modeling promise usefulness of the approach in terms of process model quality while preserving data privacy.

Keywords: Generative AI · Large Language Models · Process Modeling · Fine-Tuning · Pipeline · Process Descriptions · Dataset

1 Introduction

Large Language Models (LLMs) have gained significant attention for their ability to process text data, making them a suitable candidate for business process management tasks, particularly for business process modeling [4]. Existing LLM-based approaches for business process modeling rely on cloud-based services to generate process models from textual descriptions [1,3,7]. However, sharing sensitive data with third parties hampers data privacy and security.

This paper proposes an LLM-based processing pipeline for automatic business process modeling. Compared to existing works, the pipeline relies on fine-tuning and local deployment of LLMs. By fine-tuning an LLM, it can demonstrate competitive performance against commercial vendors like ChatGPT with GPT-4 on specific tasks [2,9], while offering an improved control over data security when employed locally.

We use prompt engineering techniques and input from process experts to generate a representative data set consisting of process trees [8] and textual process descriptions. The data set is then used to fine-tune a local LLM.

We demonstrate the feasibility of the pipeline by implementing a tool for generating training data with scripts for fine-tuning LLMs[1] relying on our dataset[2], and providing the first fine-tuned LLM for modeling process trees when prompted with process descriptions[3].

The remainder of the paper is structured as follows. Section 2 presents a comparative analysis of related works. The LLM-based pipeline is presented in Sect. 3. The paper concludes with Sect. 4.

2 Related Work

Existing LLM-based approaches primarily use the latest versions of ChatGPT [1,3,4] and, to a lesser extent, Gemini for modeling processes based on textual descriptions [7]. Common across the approaches is the integration of definitions for process modeling languages, ensuring that the LLM response uses the expected process model notation. Additionally, few-shot learning techniques have been found to be advantageous in improving process model quality. Kourani et al. [7] further report improved results through role injection and negative prompting. However, to the best of our knowledge, there remains a gap in the literature regarding techniques that utilize locally executed LLMs and fine-tuning. Even though other authors considered fine-tuning, it is often reported as unfeasible due to the extensive datasets that are required but not available.

3 Pipeline for Process Modeling Using Local LLMs

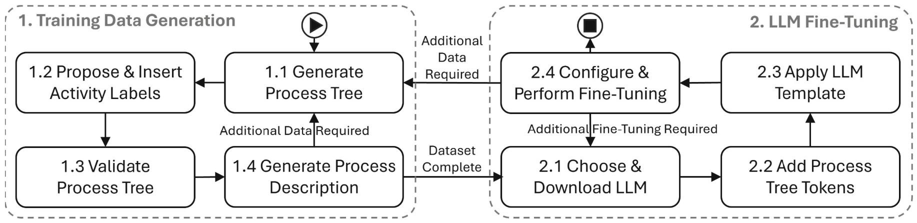
1. Training Data Generation
1.1 Generate Process Tree
1.2 Propose & Insert Activity Labels
1.3 Validate Process Tree
1.4 Generate Process Description
Additional Data Required
Additional Data Required
Dataset Complete
2. LLM Fine-Tuning
2.1 Choose & Download LLM
2.2 Add Process Tree Tokens
2.3 Apply LLM Template
2.4 Configure & Perform Fine-Tuning
Additional Fine-Tuning Required

Fig. 1. Pipeline for generating training data and fine-tuning an LLM.

Figure 1 shows our pipeline, which consists of two phases: *(1) Training Data Generation* and *(2) LLM Fine-Tuning*. In the first phase, labeled process trees and their corresponding textual descriptions are generated to serve as a training data set for fine-tuning in phase 2. The first phase consists of four steps and is performed iteratively while additional data is required.

[1] https://github.com/ApaydinK/local-LLMs-for-process-modelling.
[2] https://huggingface.co/datasets/ApaydinK/process_trees_w_descriptions.
[3] https://huggingface.co/ApaydinK/lora_model_process_tree_generator.

1.1. Generate Process Tree: In this step the user generates a random process tree using e.g. the implementation of the framework described by [6] in PM4PY. We generated random process trees in increasing complexity. Starting with three activities and going up to nine activities while iteratively restarting the first phase. The weights for choosing operators randomly was set to sequence 0.5, choice 0.2, parallel 0.2, and loop 0.1. In this step, the activities have dummy labels. **1.2 Propose & Insert Activity Labels**: In this step, the LLM proposes concrete process activity labels to replace the dummy labels based on the process tree's control flow. Subsequently, in step **1.3 Validate Process Tree** the user validates and revises the meaningfulness of the process tree and thus can shift or rename activities and operators. Next, in step **1.4 Generate Process Description** the user manually generates a textual process description corresponding to the process tree of step 1.3. In our case, three process experts leveraged our tool (see footnote 1) to view the process tree while generating a corresponding process description.

In the second phase, the process trees and corresponding process descriptions are used to fine-tune a local LLM in four steps. **2.1 Choose & Download LLM:** First, a publicly available LLM is chosen and downloaded. We chose Llama 3 8B Instruct[4] and downloaded the LLM without quantization as it has high rankings across benchmarks on the HuggingFace LLM Leaderboard[5]. **2.2 Add Process Tree Tokens:** The string representation of the process tree's operators are then added to the tokenizer of the LLM. **2.3 Apply LLM Template:** Then, the LLM-specific template for prompts and responses is applied. This is done to ensure that the training data will be delivered to the LLM in the format that it requires, e.g. providing a system message labeling the textual description as input and defining the process tree as the desired output. **2.4 Configure & Perform Fine-Tuning:** Finally, the LLM is fine-tuned. This step can involve testing different parameter configurations. We used LoRA [5] with rank 16 for efficient fine-tuning and bfloat 16 to improve accuracy over 10 epochs.

The performance of the fine-tuned LLM was evaluated iteratively after performing both phases of the pipeline three times. For evaluation, we used one process description from the PET Dataset[6] and modeled a ground truth process tree for comparison with sequence, choice, parallel, and loop operators. We observed that after fine-tuning with 40 examples, the LLM modeled sound process trees in 11 of 15 cases. After fine-tuning with 80 examples, all modeled process trees were sound. After fine-tuning with 120 examples, the proposal of process activity labels improved further - on average, 6,66 out of the 8 expected activities were modeled successfully. However, despite these advances, challenges remained in finding the correct process model. Especially the identification of parallel activities was difficult. The average F1 score of discovered process models increased from 0.36 after the first to 0.53 after the third iteration.

[4] https://llama.meta.com/.
[5] https://huggingface.co/spaces/open-llm-leaderboard/open_llm_leaderboard.
[6] https://pdi.fbk.eu/pet-dataset/.

4 Conclusion

In this paper, we proposed a pipeline for fine-tuning locally deployed LLMs for modeling process trees based on textual descriptions. We demonstrated the feasibility of the pipeline by generating training data and fine-tuning a local LLM. After fine-tuning with 120 examples, the local LLM exhibited adequate performance in proposing activity labels. It consistently generated sound process trees, while the identification of parallel activities still has potential for improvement.

Future research should focus on expanding the size and diversity of the training dataset. Incorporating a wider array of process trees and using different linguistic styles for process descriptions could enhance the LLM's ability to generalize. Additionally, decomposing the modeling task into smaller steps could lead to more robust results. For instance, initially extracting activity labels followed by determining the control flow for each activity might yield improved outcomes. Finally, exploring various LLMs and fine-tuning configurations could provide further insights into optimizing performance.

Acknowledgments. This project has received funding from the State of Schleswig-Holstein under the Datencampus project grant no. 220 21 016, the Federal Ministry for Digital and Transport under the CAPTN-Förde 5G project grant no. 45FGU139 H and the German Federal Ministry of Education and Research (BMBF) for the ABBA project grant no. 16DHBKI002, 16DHBKI003, 16DHBKI004, 16DHBKI005.

References

1. 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. Lecture Notes in Computer Science, vol. 13585, pp. 182–199. Springer, Cham (2022)
2. Bucher, M.J.J., Martini, M.: Fine-tuned 'small' LLMs (still) significantly outperform zero-shot generative AI models in text classification (2024)
3. Fill, H.G., Fettke, P., Köpke, J.: Conceptual modeling and large language models: impressions from first experiments With ChatGPT. Enterp. Modell. Inf. Syst. Archit. (EMISAJ) **18**(3), 1–15 (2023)
4. 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)
5. Hu, E.J., et al.: LoRA: low-rank adaptation of large language models (2021)
6. Jouck, T., Depaire, B.: PTandLogGenerator: a generator for artificial event data (2016)
7. Kourani, H., Berti, A., Schuster, D., van der Aalst, W.M.P.: Process Modeling with large language models. In: International Conference on Business Process Modeling, Development and Support, pp. 229–244. Springer (2024)
8. van Zelst, S.J., Leemans, S.J.J.: Translating workflow nets to process trees: an algorithmic approach. Algorithms **13**(11), 279 (2020)
9. Zhao, J., et al.: LoRA land: 310 fine-tuned LLMs that rival GPT-4, a technical report (2024)