LLMs and Process Mining: Challenges in RPA

Task Grouping, Labelling and Connector Recommendation

Mohammadreza Fani Sani[1], Michal Sroka[1], and Andrea Burattin[2]

[1] Microsoft Development Center Copenhagen, Copenhagen, Denmark
[2] Technical University of Denmark, Kgs. Lyngby, Denmark
{mfanisani, misroka}@microsoft.com, andbur@dtu.dk

Abstract Process mining is often used to identify opportunities for process automation leading to improved efficiency and cost savings. Robotic process automation (RPA) is a fast-growing area that provides tremendous productivity growth to a growing number of companies across many industries. RPA tools allow users to record their work and then propose areas for automation, and produce scripts to automate work. Recording how a process is conducted, coupled with process mining techniques, offers the most detailed view of what process is followed, how well it is followed, and whether there are areas for automation or improvement in process or policies. However, the main challenge to deriving these insights is the need for grouping fine-grained recorded tasks into events, giving them names, and proposing how to automate those high-level tasks. In this paper, we propose a framework for using large language models (LLMs) to assist users in these steps, by leveraging their natural language understanding and generation capabilities. We first address the problem of event log generation, which is an input of automation techniques, by using LLMs to group and label tasks based on their semantic similarity and context. We then tackle the problem of connector recommendation by using LLMs to recommend best plugins to automate tasks. We evaluate our approach on a real publicly available dataset, and show that it can improve the quality and efficiency of event log generation and connector recommendation, compared to the baseline methods.

Keywords: Process Mining · Large Language Models, Robotic Process Automation

1 Introduction

Process mining is a data science field that aims to discover, monitor, and improve business processes based on event data produced by information systems supporting process executions. Process mining consists of several tasks, such as *process discovery*, *conformance checking*, and *process enhancement*. Process discovery generates a process model from event logs, conformance checking compares the logs and model, and process enhancement provides insights for improvement [1].

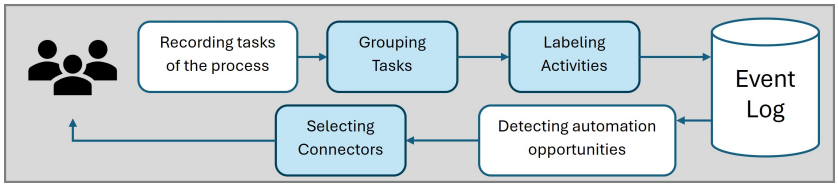

Figure 1: Schematic view of process automation. We first need to gather an event log from the users' process recordings. Thereafter, detecting automation opportunities for automation and select the corresponding connectors for recommendations. In blue, the activities tackled by this paper.

One of the applications of process mining is process automation, which aims at increasing the performance and efficiency of processes by automating manual and repetitive tasks. Process automation can increase business processes' efficiency, quality, and reliability by using software robots or *connectors* that can execute tasks automatically or semi-automatically. For example, in an invoice processing scenario, a connector can extract the relevant information from a scanned document and store it in a database, or send an email notification to the customer. To do that it is usually required to provide event logs that capture the execution of the process instances and to select the corresponding connectors that enable the integration of different services and applications in the automation workflow.

Preparing event logs and selecting connectors are not trivial tasks, and often involve manual effort and domain knowledge from users. For example, users need to group their recorded tasks into activities, which are the basic units of analysis in process mining, and label them with meaningful names that reflect their semantics and goals. These steps are crucial for producing high-quality data (necessary to obtain reliable results) yet can be challenging as, for example, grouping tasks and labeling the activities can be inconsistent and error-prone, especially for complex and heterogeneous processes that involve multiple users, systems, and data sources. Moreover, users must select the most suitable connectors for automating the activities, among the hundreds of available options, depending on the functionality, compatibility, and security of the services and applications involved. These steps can be time-consuming and may limit the potential benefits of process automation.

Fig. 1 provides a schematic view of process automation. To automate a process, we usually need to collect an event log which is usually done by several users collaboratively. We record all the tasks that different users do. Afterward, the users should group the tasks into activities, and label the activities by different words. After that we obtain an event log. If we find some places for automation we can recommend the connectors for each activity.

To overcome these challenges, in this paper, we propose a novel approach incorporating large language models (LLMs) to assist users in task grouping, labeling, and connector recommendation. LLMs are neural network models trained on massive amounts of text data. They can perform various natural language

understanding and generation tasks, such as answering questions, summarizing texts, or generating sentences [2].We leverage the LLMs' capabilities to analyze the event data, generate meaningful labels, and recommend relevant connectors, based on natural language prompts and queries. We use GPT 3.5 Turbo [3] and GPT-4 [4] as an example of LLMs, but our approach can be generalized to other LLMs as well.

Our approach consists of three main steps:

1. Task grouping: We use LLMs to cluster the tasks into activities, based on their similarity in terms of data, system, and user. We describe the tasks' attributes, their time, and their order in a natural language prompt and ask the LLM to assign a cluster label to each task. We then aggregate the tasks with the same cluster label into an activity.
2. Labeling: We use LLMs to generate meaningful labels for the activities, based on their cluster labels and task attributes. We use LLM to generate a short and descriptive label for the activity.
3. Connector recommendation: We use LLMs to recommend the most suitable connectors for automating the activities, based on their labels, task attributes, and available connectors.

We evaluate our approach on a real publicly available dataset, which contains event logs from 50 different business processes [5]. We compare our approach with the baseline method, based on traditional clustering, string matching, and ranking algorithms leveraging process discovery and conformance checking [6]. We measure the quality and efficiency of our approach in terms of accuracy, completeness, consistency, and time. We show that our approach can improve the quality and efficiency of task grouping, labeling, and connector recommendation, compared to the baseline method.

The contributions of this paper are as follows:

- We propose a novel approach to use LLMs for task grouping, labeling, and connector recommendation in process mining, which can assist users in preparing event logs and selecting connectors for process automation.
- We evaluate our approach on a real, publicly available dataset, and show that it can improve the quality and efficiency of the process automation steps, compared to the baseline methods.

The rest of the paper is organized as follows. Section 2 reviews the related work on process mining, process automation, and LLMs. Section 3 describes the preliminaries and the notation used in the paper. Section 4 presents the proposed approach for task grouping, labeling, and connector recommendation. Section 5 reports the evaluation results, then, Section 6 discusses the limitations and the future work of our approach and finally, Section 7 concludes the paper.

2 Related Work

Several research has been done in the areas of recommender systems and process automation in process mining. A comprehensive survey of these systems is available in [7].

In [8], the authors present a process mining technique to enable effective RPA activities toward process improvement. Moreover, [9] discusses the capabilities of processes-based techniques with RPA and proposes an automatable indicator system as well as RPA activities to maximize the automation investment.

In [10], a survey on task mining has been reported and several applications and challenges in front of this research are represented. The authors in [11], present an unsupervised approach for task recognition from user interaction streams. Moreover, [12] has proposed a tool to record the interactions with user interfaces and to generate an event log that can be used to bridge the gap between process mining and RPA by detecting the tasks that can be automated. Furthermore, in [6], we propose to use process models and their conformance for recommending connectors. This approach requires some training event logs and discovers models for each connector. Thereafter, for each process instance, we recommend the connectors that their models are fitted more to it.

Several recent works analyzed the interaction of LLMs and process mining. For example, in [13] authors focused on conversation agents; [14] focus on specific prompt strategies similarly to what is reported in [15] (where the focus is also on abstraction). Other work focused on explainability of prediction results in process mining [16]. Authors of [17] identify six research directions where LLMs can be useful tools in BPM.

3 Preliminaries

In this section, we formalize the main concepts necessary to define a task, a recording, a grouping of tasks, labeling, and connector recommendation.

To capture how a process is executed by different users, they record the required tasks regarding that process. For each task, different information is captured such as the tools that are used, the input data, the action that is done, etc. In the following, we formally define a task.

Definition 1 (Task). *A task* $s{=}(a_1, a_2, \ldots, a_n, \tau)$ *is a tuple of attribute values, with domains* $a_i{\in}\,\mathrm{dom}(a_i)$ *and* $\tau{\in}\mathbb{T}$ *where* $\mathbb{T}$ *is the time domain with discrete time units. Let* $\mathcal{TK}$ *be the universe of tasks and* $s{\in}\mathcal{TK}$ *be a task. The projector operator* π *can be used to access specific components of the tuple:* $\pi_i(s){=}a_i$ *returns the attribute value of attribute* A_i *and* $\pi_\tau(s)$ *returns the execution time of* s*.*

Given a set $\mathcal{S}$ of elements, we apply the Kleene operator to it $\mathcal{S}^*$ to indicate all possible sequences comprising all elements of $\mathcal{S}$. An element $S{=}\langle s_1, \ldots, s_m\rangle{\in}\mathcal{S}^*$ is therefore a sequence of elements each of which belongs to $\mathcal{A}$. As shortcuts, we use the length operator $|S|{=}m$; indexes: $S_i{=}s_i$; and $S^{first}{=}s_1$, $S^{last}{=}s_m$. The concatenation of two sequences s_1 and s_2 is defined as $s_1{\cdot}s_2$.

A record is a sequence of tasks that are sorted based on their execution times.

Definition 2 (Record). *Let* $\mathcal{TK}$ *be the universe of tasks,* $\mathcal{I}$ *be the universe of recording ids, and* $\mathbb{T}$ *be the time domain with discrete time units. A record* $r = (id, S)$ *is a tuple where* $id \in \mathcal{I}$ *is the recording identifier for this sequence*

and $S \in \mathcal{TK}^*$ is a finite sequence of tasks which are ordered by execution time: $\forall_{i,j \in \{1,\ldots,|S|\}} i < j \implies \pi_\tau(s_i) \leq \pi_\tau(s_j)$.

In task grouping, the goal is to convert low-level behavior to higher-level. To achieve this, we segment and group the tasks of each record. In the following, we formally define task grouping.

Definition 3 (Task grouping). *Let $\mathcal{R}$ be the universe of records achievable from the universe of tasks $\mathcal{TK}$ and the universe of recording ids $\mathcal{I}$; then let $(id, S) \in \mathcal{R}$ be a record. We can define $TG : \mathcal{R} \to \mathcal{P}(\mathcal{TK}^*)$ [1] as a task grouping function. This function receives a record and returns a set of sequences of tasks such that, given a record $r = (id, S)$ and the resulting grouping $g = TG(r)$ we have that $g_1 \cdot g_2 \cdot \ldots \cdot g_{|G|} = S$ and*

$$\forall_{i,j \in \{1,\cdots,|g|\}} \left(\pi_\tau(g_i^{first}) < \pi_\tau(g_j^{first}) \implies \pi_\tau(g_i^{last}) < \pi_\tau(g_j^{first})\right) \wedge \left(\pi_\tau(g_i^{first}) > \pi_\tau(g_j^{last}) \implies \pi_\tau(g_i^{first}) < \pi_\tau(g_j^{last})\right)$$

In other words, the task grouping splits a sequence of tasks into several subsequences of its tasks. However, the concatenation of all subsequences should recreate the original sequence and, for any two subsequences S_i and S_j, all the tasks S_i should be executed before all the tasks of S_j, or all of them should be executed after the tasks in S_j.

After grouping the tasks, we must also provide a descriptive name for each group. We call this part *labeling* and define it in the following.

Definition 4 (Labeling). *Let $\mathcal{I}$ be the universe of recording ids, $\mathcal{TK}$ be the universe of tasks, and $\mathcal{A}$ be the universe of activity names (to an abstraction level that is meaningful to the end-user). We define $L : \mathcal{TK}^* \to \mathcal{A}$ as a labeling function that assigns an activity name to a sequence of tasks.*

Using the labeling function, we can generate an event log from the recordings. The procedure of converting task attributes to event log attributes with high-level activities is out of the scope of the paper. However, in the following, we formally define an event log, following the typical definitions found in the literature.

Definition 5 (Event log). *Let $\mathcal{C}$ be the universe of case identifiers, $\mathcal{A}$ the universe of activity names, and $\mathcal{X} = \mathcal{N} \times \mathcal{V}$ the set of extra attributes (each comprising a name in $\mathcal{N}$ and a value in $\mathcal{V}$). The event universe $\mathcal{E} = \mathcal{C} \times \mathcal{A} \times \mathcal{P}(X)$ is the set of all possible events. From the event universe, it is possible to extract* traces *as sequences of events that share the same case identifier. An event log is then a multi-set of traces.*

It is possible to see a direct mapping from the concepts in record, tasks (grouping), and labeling into the definition of event log. Specifically, the recording ids (of records) are the case identifiers (of event logs) which are producing

[1] With $\mathcal{P}(A)$ we indicate the powerset of A: the set of all possible subsets of A.

traces (of event logs), and tasks in records are grouped and labelled, and these labels are activity names (in event logs).

After generating an event log and finding the automation opportunities, we may need to select some connectors for the automation. As there could be many available connectors, having a connector recommender that suggests the ideal recommender is valuable. In the following, we formally define a connector recommender.

Definition 6 (Connector recommender). *Let $\mathcal{A}$ denote the universe of activity names and let Γ be the universe of connectors. We define $CR : \mathcal{A}^* \to \mathcal{P}(\Gamma)$ as a* connector recommender *that receives a sequence of activities and returns a set of connectors that could replace the sequence of activities. Finally, we can use CR_k to denote a recommender that returns exactly k connectors.*

4 Using LLM for Task Grouping, labeling, and connector recommendation

In this section, we aim to explain how, by using LLMs, we are able to group the tasks, label appropriate activities to each group, and recommend some connectors for each trace. We aimed to design a workflow that requires business knowledge. Consequently, the designed workflow can be used for different business recordings.

The schematic view of the proposed workflow is shown in Fig. 2. We separate the proposed method into *Preparation* and *Application* phases. In the preparation phase, the goal is to generate a template for the future prompts. The key to high quality results from LLM is a process of improving this template called prompt engineering [18]. In the prompt, we should describe the assignment (i.e., the role of LLM), what are inputs, and how we expect to have outputs. Depending on the assignment, we can have zero to few examples to let LLM understand the task more.

One aspect that should be considered in the preparation phase is that different LLMs may need different prompt engineering styles. In this paper, we consider consider GPT-4-32K-0314 and GPT-3.5-Turbo-16K models.

In the application phase, we use the designed template and fill it with the values of inputs for the new cases. The filled template is given to LLM and it returns the expected output.

The proposed workflow, can be used for different assignments and in this paper, we used it for task grouping, labeling, and connector recommendation.

5 Evaluation

We applied these methods on a public dataset reflecting real life scenarios, containing 50 processes labelled by humans [5]. For each process, we have at least 3 cases manually labelled. The labels include how tasks should be grouped and named. The dataset includes ids of step, process, and recording and label event

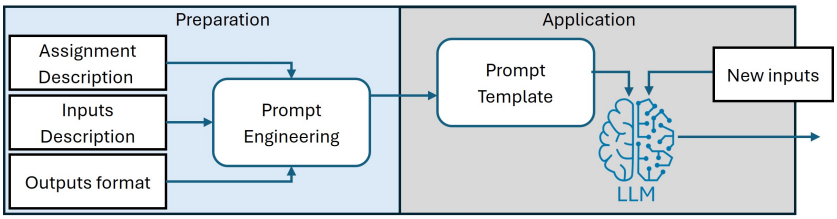

Figure 2: The schematic view of process workflow to use LLMs for RPA.

names. It also contains some columns like *StepName* which is the name of the step that can be from one of 19 different options, e.g., *Press button in window*. StepDescription, which is a more detailed description of the step, e.g., *Button 'New Tab' in Window 'Process-Microsoft Edge'*, or ApplicationProcessName, which is the process name, taken from the opened application, e.g., *msedge*. For details, we refer the reader to [19]. Moreover, a filtered version of this dataset is preprocessed and labeled for evaluating the connector recommendation[2]. The prompt templates that we used for the experiments can be found in `https://github.com/mfanisani/LLM4RPA`.

We utilized LLMs for task grouping, labeling, and connector recommendation assignments. Thus, we provided a prompt template for each assignment, and we used two configurations of OpenAI as the LLMs, i.e., GPT-3.5-Turbo (*GPT3.5* in short) [3] and *GPT-4* [4]. In the following, for each assignment, we first explain the experimental metrics. Then, for each task, we provide the evaluation results. Because the LLM output is non-deterministic, we repeat the experiments for each assignment 6 times.

5.1 Task Grouping Experiments and Results

To evaluate the quality of the proposed task grouping, we compared the groups suggested by the LLM method with human labels from the dataset. A metric widely used for measuring how the quality of clusters compares with expected clustering is Normalized Mutual Information (NMI) [20]. It is based on the mutual information measure, which quantifies how much information is shared by two random variables. NMI normalizes the mutual information by some function of the entropy of the two clustering, so that the score ranges from 0 to 1. NMI is often used to evaluate the performance of clustering algorithms when the true labels are known.

We can compute the NMI using the following formula

$$\mathrm{NMI}(U,V)=\frac{2\times I(U,V)}{H(U)+H(V)} \tag{1}$$

where U and U are two clustering outputs for the same data set, $I(U,V)$ is the mutual information between U and V, and $H(U)$ and $H(V)$ are the entropy of

[2] The dataset and its filtered version are available via `https://github.com/microsoft/50BusinessAssignmentsLog` and `https://github.com/nikraftarf/Recommender-System-Based-on-processMining/tree/main/Data`, respectively.

Table 1: NMI scores for different grouping methods. Where the Error column represents the margin of error at 1.984 confidence coefficient.

Grouping Method	**Run 1**	**Run 2**	**Run 3**	**Run 4**	**Run 5**	**Run 6**	**Mean**	**Error***
ApplicationProcessName	0.36	0.36	0.36	0.36	0.37	0.36	0.36	0.0016
ApplicationParentWindowName	0.55	0.54	0.54	0.54	0.54	0.54	0.54	0.0023
GPT3.5 Turbo	0.58	0.59	0.56	0.61	0.59	0.59	0.59	0.0123
GPT4	0.75	0.76	0.75	0.76	0.74	0.74	0.75	0.0060

U and V, respectively. A higher NMI indicates a better agreement between the clustering 's outputs, while a lower NMI indicates a worse agreement.

As a baseline we consider grouping tasks when the user changes their focus. The change of focus can manifest itself in changing of the window in focus, or changing the window name. The former is represented in the dataset as changes of ApplicationProcessName column, and the latter as ApplicationParentWindowName.

Task grouping is the first task in the pipeline as indicated on the diagram in Fig. 1. The quality of task grouping strongly depends on the considered input data as described in Definition 3. For grouping and labelling, we use the following columns: *StepId*, *StepName*, *StepDescription*, *ApplicationProcessName*, and *ApplicationParentWindowName*. then, for each step, we concatenate the mentioned values into a concatenated string as the representative of the step.

Thereafter, we formulate a prompt template including instructions, knowledge about task grouping, and a few examples to understand the response and the expected outputs. As a result, for each row of input data representing a task, we obtain a predicted label. If multiple consecutive tasks are assigned the same label, we consider them as a group, i.e., a high-level event.

Table 1 shows the results of comparing human-assigned and LLM-generated groups. We run the experiments 6 times and report the average values and Error*. The results indicate that ApplicationProcessName as a baseline performs the poorest with an average score of 0.36, and ApplicationParentWindowName gives a much better estimate with an average NMI score of 0.54. The results from GPT 3.5 Turbo, are better than both baselines and offer a stable increase in quality across all experiments. We achieved the best results by using $GPT4$ that outperforms all others by a large amount, obtaining an average NMI score of 0.75. It indicates that using LLMs improves the quality of *task grouping* assignment.

5.2 Group Labelling Experiments and Results

To evaluate the quality of the label generated for each event, i.e., a group of tasks, we are using three metrics to measure different types of similarities. To check for *syntactical similarity*, we used BLEU score [21] and normalized Levenstein distance [22]. Moreover, to assess *semantic similarity*, we used the *cosine similarity* between the embedding generated using text-embedding-ada-002 model for both the generated and expected labels.

Table 2: Examples of labels given by human experts and generated by *GPT4*.

Id	Label	Cosine Similarity	BlEU	Normalized Levenstein
1	*Human*: Open the attendance sheet *GPT4*: Open Attendance Sheet	0.96	0.71	0.16
2	*Human*: Go to approval app *GPT4*: Search for Approval App	0.95	0.5	0.44
3	*Human*: Open email app. *GPT4*: Access Gmail	0.86	0	0.71
4	*Human*: Create a report *GPT4*: Create Power BI report	0.86	0.5	0.53
5	*Human*: Create a tasks through roadmap app. *GPT4*: Add task to project	0.82	0	0.68
6	*Human*: Find the email notification regarding daily attendance and verify it *GPT4*: Get email notification	0.84	0.12	0.71

Table 3: The Similarity of labels that are assigned by humans and the ones predicted by GPT.

Label Similarity Metric		Run 1	Run 2	Run 3	Run 4	Run 5	Run 6	Avg	Stdev	Error
BLEU	*GPT3.5*	0.39	0.39	0.39	0.39	0.39	0.39	0.39	0.0	0.0
	GPT4	0.41	0.41	0.41	0.43	0.43	0.41	0.42	0.0103	0.0168
Cosine Similarity	*GPT3.5*	0.81	0.81	0.81	0.81	0.81	0.81	0.81	0.0012	0.0010
	GPT4	0.83	0.83	0.83	0.83	0.83	0.83	0.83	0.0005	0.0004
Normalised Levenstein	*GPT3.5*	0.87	0.89	0.88	0.85	0.87	0.85	0.87	0.0141	0.0114
	GPT4	0.86	0.88	0.87	0.86	0.87	0.86	0.87	0.0062	0.0050

The comparison of the human assigned labels and LLM-generated labels using different measures is presented in Table 3. The high cosine similarity scores indicate that, using LLM models, we provide activity labels that are semantically similar to the ones that were assigned by humans. It should be noted that by using the proposed approach, we can save a considerable amount of post-processing time as different users may assign different labels to a group. Note that using LLM lets us have labeling in a central way. It is also worth to mention that the syntactical similarity is relatively low. By investigating the labels, we found that it is mainly because of humans and GPT models may use different words to name the same activity. Selected examples of human and LLM-generated labels are presented in Table 2. For example, in the sixth example, the human and LLM-based labels are semantically very close; however, their syntactical similarity is quite low due to the chosen wordings. Moreover, the results indicate that GTP3.5 and GPT4 generated comparable quality of labels for the events.

5.3 Connector recommendation Experiments and Results

In the last assignment, to recommend connectors, we use cr_k with different k-values from 1 to 5 (cf. Definition 6). To evaluate the accuracy of the connector recommendation, we have used the following formula:

$$RetrieveRate = \frac{|\{\text{Labeled Connector}\} \cap \{\text{Recommended Connectors}\}|}{|\{\text{Recommended Connectors}\}|} \quad (2)$$

The higher value means a more accurate recommendation.

Table 4: Comparing the retrive rate of using process discovery [6] and LLM methods when different number of connectors are recommended.

		Retrive Rate											
		K=1			K=2			K=3			k=5		
Connector	Records#	PD	*GPT*3.5	GPT4	PD	*GPT*3.5	GPT4	PD	*GPT*3.5	GPT4	PD	*GPT*3.5	GPT4
Approvals	7	57	100	86	71	100	100	71	100	100	100	100	100
googlecalendar	6	50	50	17	50	67	67	67	67	67	100	83	83
Microsoftforms	5	60	60	100	60	60	100	80	60	100	100	60	100
Office365users	4	75	100	100	75	100	100	100	100	100	100	100	100
onenote	4	25	50	100	25	50	100	50	50	100	50	50	100
Outlook	5	60	0	80	60	60	80	80	60	100	80	60	100
Planner	5	60	60	100	80	60	100	80	60	100	100	60	100
rss	4	0	25	50	0	25	50	0	25	50	25	25	50
Sendmail	5	20	60	60	80	100	60	80	100	100	100	100	100
Sharepoint	4	40	75	100	40	100	100	60	100	100	60	100	100

To generate the prompt template for the connector recommendation assignment, we only used *StepName*, *StepDescription*, *ApplicationProcessName*, *ApplicationParentWindowName* columns of all steps that belong to a record as the inputs. We gave all the mentioned data as a concatenated string for each record. The results of using LLM for connector recommendation are presented in Table 4. Here, we compared the result of the proposed approach with the method that is presented in [6](i.e., *PD*). In most cases, we have a higher *Retrieve Rate* using LLM models. Specifically, for $k<3$, LLM could be considered as the best method. However, by increasing k, we do not have that much improvement.

Note that to use the method that is proposed in [6], we need some training recordings, and adjusting process models using different parameters of process discovery algorithms that can be time consuming. However, using LLMs, we do not need to have any labeled data, and the only requirement is having the list of available connectors for the recommendation. Moreover, using the method that is presented in [6], we only benefit from control-flow information, however, using LLM, we benefit from more data attributes.

6 Discussion

The results of experiments indicate that we can use LLMs for automating many process mining and automation activities. The task grouping and labeling that are covered in this paper can be challenging and error-prone. Different users, may group tasks with different granulates and use different labels. Note that even small difference in the labels (e.g., "send email" and "sending email") leads to having more than one activity in the event log.

Although the results indicate that we have high accuracy in the given tasks, it is suggested not to remove the users from the process. We recommend to use LLMs as a method to recommend grouping, labels, and connectors, and the user selects to confirm them. The initial exploration of the baseline Process Discovery (PD) method has yielded results not exceptionally off compared to those of the LLM, particularly evident with higher K values (cf. Table 4) by incorporating additional top connectors. Delving into the realm of explainability, it's essential

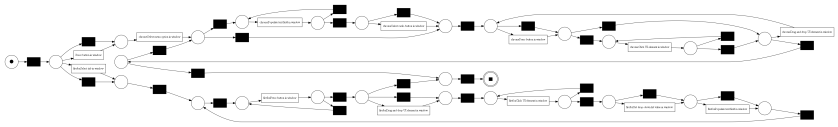

Figure 3: The process model (in the Petri net description) discovered on projected event log of *Office365users* connector.

to consider the differences in how the two approaches convey their results. The Process Models generated through the PD Method offer a level of explainability and the capability of comparing the models underlying different connectors, which contrasts with the self-explanation of the GPT model. Though it is possible to ask GPT models for an explanation of the results, their actual effectiveness remains to be assessed. For example, Fig. 3 presents the process model that will be used for the *Office365users* connector. For the process instances that belong to this connector, we usually have similar results using both approaches.

Additionally, the Process Discovery method is capable of computing the conformance of execution with respect to the reference model, thus providing a measure of the "confidence" of the trace to the reference. While the PD method has already demonstrated good performance from this point of view, LLMs are fundamentally incapable of expressing any confidence level for the results they provide, thus challenging their trustworthiness with their human operators.

An interesting point of comparison arises in terms of data ingestion capabilities. The GPT model possesses the capacity to assimilate a broader spectrum of information, as opposed to the process discovery, which can ingest only event logs (i.e., data referring to the control-flow). This aspect opens the discussion on the varying degrees to which each approach can accommodate and incorporate the available data. From this point of view, LLMs outperform process mining-based systems and certainly can serve as inspiration to improve process discovery algorithms.

7 Conclusion and Future work

In this paper, we proposed to use LLMs for assisting users in process mining and automation tasks, such as task grouping, event labeling, and connector recommendation. We showed that LLMs can leverage their natural language understanding and generation capabilities to produce high-quality and semantically coherent outputs, based on the recorded data of user activities. We evaluated our approach on a real publicly available dataset, and compared it with the baseline method. We found that by using LLMs we are able to improve the quality and efficiency of task grouping and event labeling, as well as provide accurate and relevant connector recommendations.

Looking towards future prospects, the idea of combining the strengths of both process mining and LLMs emerges as a compelling avenue. This fusion could potentially yield results that surpass the capabilities of each method in isolation. A noteworthy distinction lies in the nature of the dataset: the wealth of textual information available can be harnessed by the LLM but not as readily

by process mining. This discrepancy underscores the unique advantages and considerations of each approach.

References

1. van der Aalst, W.M.P.: Process Mining - Data Science in Action, Second Edition. Springer Berlin Heidelberg (2016)
2. Kasneci, E., et al.: ChatGPT for good? On opportunities and challenges of large language models for education. Learning and Individual Differences **103** (2023)
3. OpenAI: Gpt-3.5 turbo: A large language and code model with function calling data. `https://platform.openai.com/docs/models/gpt-3-5` (2023)
4. OpenAI: GPT-4 technical report. CoRR **abs/2303.08774** (2023)
5. Sroka, M., Fani Sani, M.: Recording of 50 business assignments (2023)
6. Fani Sani, M., Nikraftar, F., Sroka, M., Burattin, A.: Behavioral recommender system for process automation steps. In: Proc. of DATA, Scitepress (2023)
7. Eili, M.Y., Rezaeenour, J., Fani Sani, M.: A systematic literature review on process-aware recommender systems. CoRR **abs/2103.16654** (2021)
8. Geyer-Klingeberg, J., Nakladal, J., Baldauf, F., Veit, F.: Process mining and robotic process automation: A perfect match. In: Proc. of co-located events with BPM. Volume 2196 of CEUR Workshop Proceedings. (2018) 124–131
9. Wanner, J., Hofmann, A., Fischer, M., Imgrund, F., Janiesch, C., Geyer-Klingeberg, J.: Process selection in RPA projects - towards a quantifiable method of decision making. In: Proc. of ICIS, Association for Information Systems (2019)
10. Mayr, A., Herm, L., Wanner, J., Janiesch, C.: Applications and challenges of task mining: A literature review. In: Proc. of ECIS. (2022)
11. Rebmann, A., van der Aa, H.: Unsupervised task recognition from user interaction streams. In: CAiSE 2023, Spain, Proceedings. Volume 13901 of Lecture Notes in Computer Science., Springer (2023) 141–157
12. Choi, D., R'bigui, H., Cho, C.: Enabling the gab between RPA and process mining: User interface interactions recorder. IEEE Access **10** (2022) 39604–39612
13. Jessen, U., Sroka, M., Fahland, D.: Chit-chat or deep talk: Prompt engineering for process mining. CoRR **abs/2307.09909** (2023)
14. Berti, A., Qafari, M.S.: Leveraging large language models (llms) for process mining (technical report). CoRR **abs/2307.12701** (2023)
15. Berti, A., Schuster, D., van der Aalst, W.M.P.: Abstractions, scenarios, and prompt definitions for process mining with llms: A case study. CoRR **abs/2307.02194** (2023)
16. Stevens, A., Smedt, J.D.: Explainable artificial intelligence in process mining: Assessing the explainability-performance trade-off in outcome-oriented predictive process monitoring. CoRR **abs/2203.16073** (2022) Withdrawn.
17. Vidgof, M., Bachhofner, S., Mendling, J.: Large language models for business process management: Opportunities and challenges. CoRR **abs/2304.04309** (2023)
18. White, J., Fu, Q., Hays, S., Sandborn, M., Olea, C., Gilbert, H., Elnashar, A., Spencer-Smith, J., Schmidt, D.C.: A prompt pattern catalog to enhance prompt engineering with chatgpt. arXiv preprint arXiv:2302.11382 (2023)
19. Sroka, M., Fani Sani, M.: 50 business assignments log. (2022)
20. Cover, T.M., Thomas, J.A.: Elements of Information Theory. Wiley (2006)
21. Papineni, K., Roukos, S., Ward, T., Zhu, W.J.: Bleu: a method for automatic evaluation of machine translation. In: Proc. of ACL. (2002) 311–318
22. Yujian, L., Bo, L.: A normalized levenshtein distance metric. IEEE transactions on pattern analysis and machine intelligence **29**(6) (2007) 1091–1095