Retrieval Augmented Generation with LLMs for Explaining Business Process Models

Mirjam Minor[0000-0002-6592-631X] and Eduard Kaucher

Goethe University Frankfurt, Department of Business Informatics, Germany
minor@cs.uni-frankfurt.de

Abstract. Large language models (LLMs) and retrieval augmented generation (RAG) are undergoing rapid development. Considering a case base as a memory in a RAG system provides novel opportunities for text generation. In this paper, we investigate the role Case-Based Reasoning (CBR) could play for supporting RAG systems in generating accessible explanations of business process models. We experiment with two different case bases in a RAG system. Case base a) is dedicated to support prompt chaining by reusing index knowledge on the cases with the aim to deal with large process models that do not fit into the context window size of a recent LLM. Second, case base b) contains model-text pairs to serve as in-context examples to enhance prompt templates. Approach b) aims to improve the quality of generated text explanations for process models of normal size. Our contribution opens a novel application area for process-oriented CBR. Further, our case-based RAG system provides a contemporary alternative to traditional Natural Language Processing pipelines. The experimental results contribute to gain some insights on an inherent capability threshold of GPT-4 at which the performance decreases much earlier than having reached the given context window size, on the number of retrieved cases a recent RAG system should use as in-context examples, and on suitable prompt templates.

Keywords: Process-oriented CBR · Large Language models · Retrieval-Augmented Generation.

1 Introduction

Most digital transformation projects target at process improvements when investigating the change that digital technologies can bring about in a company [9]. Business processes are “a set of logically related tasks performed to achieve a defined business outcome for a particular customer or market” [3]. Weske claims that business processes are “essential to understanding how companies operate” [27]. The research field of process-oriented case-based reasoning (PO-CBR) [20] addresses those business needs. PO-CBR has achieved considerable success in assisting system analysts and further business process professionals [21, 11, 30, 19, 23].

However, many application domain experts and other users lack knowledge on process modelling languages. They are struggling to understand the graphical process models. Model explanations with plain text can help those users and save precious hours or days of workshops with system analysts [14]. Early model-to-text approaches use traditional Natural Language Processing (NLP) pipes as a process model verbalization technique [14]. Yet, the success of such methods depends on major knowledge engineering efforts. The coverage of the generated explanations depends on those efforts. Vidgof et al. [25] discuss the high potential of Large Language Models (LLMs) for tasks related to business process management, including model explanation. Hammond and Leake [8] as well as Watson [26] encourage the research community to experiment with a case-based memory for LLMS. The role Case-Based Reasoning (CBR) could play for generating explanations of structural models like business process models is an interesting research question.

The paper introduces a novel, retrieval augmented generation (RAG) system with an LLM using PO-CBR for the downstream task of explaining business process models. The basic idea of RAG systems is to combine a retriever module with a generator module [16], i.e. to "decouple the capacity that language models have for understanding text from how they store knowledge" [12]. We have investigated the utility of two alternative case retrieval approaches to support RAG systems:

a) for prompt chaining to overcome the size restrictions of the LLM's context window
b) for enhancing prompt templates with in-context examples for the downstream task

Approach a) reuses the indexing knowledge from the case base to tokenize the prompts into a chain. The retriever component in b) provides problem-solution-pair cases to fill the prompt templates. Approach b) is following the classical CBR approach where the case base comprises experiential knowledge to directly solve a downstream task. Both approaches provide in-context learning [2] when using the retrieval results to activate the appropriate, task-specific context of the LLM. A standard LLM has been pre-trained on large amounts of text in a task-agnostic manner. In contrast to parameter fine-tuning by further pre-training LLMs for a specific task, in-context learning allows the use of the LLM as it is [2].

For both approaches, we have conducted experiments with two process-oriented case bases derived from the same repository of 37 process models. They describe real business processes from different application areas, including human resources, insurance claims or supply chains. Case base a) comprises the business process models per se. In case base b) model-text pairs with a textual explanation as solution part are recorded. Different variants of prompt templates and prompting strategies (one-shot, few-shot) per approach are explored. The experimental results show that RAG following approach b) achieves better process explanations than zero-shot approaches. The results for approach a) are not yet convincing. However, some surprising lessons have been learned for both approaches.

The remainder of the paper is organized as follows. Some foundations of LLMs are reported as background for the work in Section 2. Related work is discussed in Section 3. The two approaches a) and b) are presented at a conceptual level in Section 4 and evaluated by means of a prototypical implementation with a repository of real process models in Section 5. A conclusion is drawn in Section 6.

2 Foundations of LLMs

LLMs address text generation by a probabilistic model $P(y|x;\theta)$ that predicts an output y for an input x [18]. The parameters θ of the model are learned in a pre-training process. Instead of learning the conditional probability distribution for (x, y)-pairs directly, the model is pre-trained for the probability $P(x;\theta)$ of text x itself from large amounts of text. In a standard language model such as GPT the text is predicted in an autoregressive fashion, predicting the linguistic tokens in the sequence one at a time. This is usually done from left to right [18].

During pre-training [22], each linguistic token u_i from a universe $U = \{u_1, u_2, ...u_n\}$ is assessed within the context of its k predecessor tokens: $P(u_i|u_{i-k}, ..., u_{i-1})$. The target function is to maximize the following likelihood:

$$L(U) = \sum_i \log P(u_i|u_{i-k}, ..., u_{i-1};\theta)$$

for the context window size of k tokens where P is modeled using a neural network with parameters θ. For details of the pre-training, we refer to the literature [22].

Prompt engineering tries to find the best prompt to solve a downstream task with the LLM [18]. A *prompt template* implements a prompting function f_{prompt} to modify the input text x into a prompt $x' = f_{prompt}(x)$ [18]. A naive example of a prompt template to solve the task of explaining a business process model is the following.

```
Consider the standard BPMN 2.0.2 specification. Please create a textual process
description for following BPMN model "Example" serialised in XML. [X]
```

where `[X]` is an input slot that can be filled with x. Further slots can be specified during prompt template engineering, such as slots for an intermediate answer `[Z]` or slots to shape the answer space `[Y]`.

Prompt augmentation (or demonstration learning) provides answered prompts that can be used to demonstrate how the LLM should provide the answer to the actual prompt instantiated with the input x [18]. Brown et al. distinguish “zero-shot”, “one-shot”, or “few-shot” learning depending on how many demonstrations (in-context examples) are provided at inference time [2]. *Prompt chaining* is “the process of breaking up complex tasks into smaller steps, where each step can be completed by an independent run of an LLM, and where the output of one or more steps is used as input for the next” [29]. Prompt chaining can be combined with prompt augmentation in a way that the prompt is enriched with a different demonstration example for each iteration in the chain to refine the intermediate answer, for instance. A systematic survey on typical prompting methods for a variety of pre-trained language models is provided in the literature [18].

3 Related work

Conversational methods have been applied in the literature [30, 13] for process modeling tasks. A conversational CBR approach has been used by Zeyen et al. [30] to support users in formulating small process models as queries to a PO-CBR system. LLMs have been applied for conversational process modeling by Klietsova et al. [13]. The conversation with an LLM provides a toolchain to support modelers in extracting tasks from textual descriptions, modeling and refining the logic and layout of a BPMN model. [1] Fill et al. [5] go a step further in fully-automated generating process models from text by means of an LLM. Another text-to-process approach using LLMs is investigated by Grohs et al. [7]. In addition, the authors aim to identify activities within a process model that are suitable to be automated by means of RPA. [2] Bellan et al. [1] use in-context learning to achieve conversational process extraction from text. In contrast to the text-to-model work reported from the literature, our explanation approach addresses the opposite direction of the pipeline namely model-to-text. Further, our RAG system does not require any intermediate representations of the BPMN models.

Explanation has been considered in PO-CBR for the purpose of understanding similarity functions by visualisation [23]. The objective and the explanation type differs from our work where a textual explanation type is used to explain process models. The explanation goal of our work is to increase transparency for the users how a process is described in a BPMN model, e.g. within a PO-CBR system or within a traditional business process management system. In particular, the explanatory texts describe the elements of the process model under consideration and elucidate the semantics of the element types in BPMN.

Somewhat less related is the work of Upadhyay et al. [24] where a data-to-text generation approach is developed based on a neural net. Another type of structural data (database tables on sports events) to be explained and another method is used to create text from data.

4 Method

The RAG system comprises a retriever module and a generator module as illustrated in Fig. 1. The input to the RAG includes a BPMN model serialized into XML. The output of the RAG is a text answer that explains the BPMN model to a user who is not familiar with process modeling and BPMN. An example is depicted in Fig. 2 with the graphical representation on the left hand side and the serialized, abbreviated form on the right hand side. The rectangles with the rounded corners depict activities, namely to invoke a risk assessor service first and, in case of a large loan or in need of a review for the risk, to invoke

[1] Business Process Modeling Notation: a standard graphical modeling language for business processes, see Decker et al. [4] for an introduction.

[2] Robotic Process Automation: the automation of tasks which have been performed by human agents so far by means of software robots technology [28].

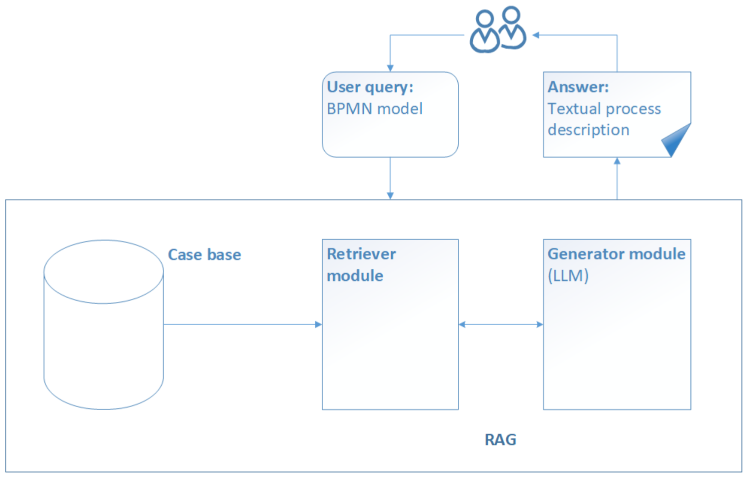


Fig. 1: RAG system for explaining business process models.

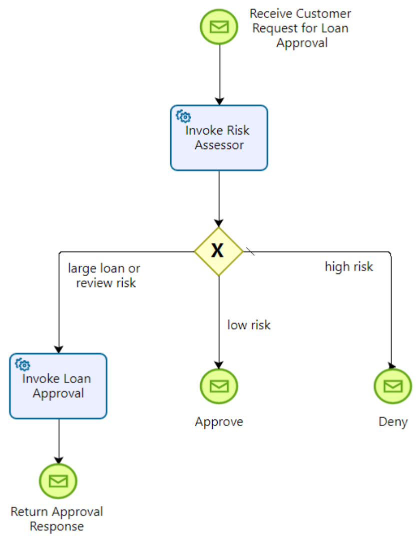
a loan approval service next. The `<bpmn:serviceTask ...>` on the right hand side corresponds with the first activity "Invoke Risk Assessment" (for a more detailed description including some explanatory remarks on each element type of the BPMN process, see Section 4.2).

The retriever module differs among the approaches. Approach a) is described in Subsection 4.1 and approach b) in Subsection 4.2. In contrast to Lewis et al. [16], we do not train the parameters of the entire RAG system end-to-end. Instead, in-context learning is applied for both case-based approaches.

4.1 RAG with case-based prompt chaining

In this approach, the retrieval of cases provides indexing knowledge for the decomposition of a process model for prompt chaining. The intuition behind this is that process models exceeding the size of the context window can be explained part by part. For each chunk, explanatory texts are generated and merged into an overall explanation of the entire process model.

The case base records serialized BPMN models (see right hand side of Fig. 2). The cases have been automatically sliced into chunks of the same length already. The size of the chunks and the chunk overlap is specified in terms of number of tokens. The chunk overlap aims to inform the LLM how the multiple chunks connect to each other. The set of chunks is sent to a general purpose embedding model that turns the chunks into embedding vectors. Such neural embeddings are a common means in LLM's to represent data. In a RAG system, a similarity metric (e.g., cosine similarity) is calculated on the embedding vectors. A node records an embedding vector and the reference back to the actual textual data from the chunk. These nodes form the index knowledge to represent a BPMN case.



```
<?xml version="1.0" encoding="UTF-8"?>
<bpmn:definitions
xmlns:bpmn="http://www.omg.org/spec/BPMN/
  20100524/MODEL"
... >
<bpmn:process id="Process_03htkt1" isExecutable=
"true">
  <bpmn:startEvent id="Event_1haplvf" name=
  "Receive Customer Request for Loan Amount">
    <bpmn:outgoing>Flow_0haw72c</bpmn:outgoing>
    <bpmn:messageEventDefinition
    id="MessageEventDefinition_1xqu562" />
  </bpmn:startEvent>
  <bpmn:sequenceFlow id="Flow_0haw72c" sourceRef=
  "Event_1haplvf" targetRef="Activity_0mew4hd" />
  <bpmn:serviceTask id="Activity_0mew4hd" name=
  "Invoke Risk Assessor">
      <bpmn:incoming>Flow_0haw72c</bpmn:incoming>
      <bpmn:outgoing>Flow_1ssbkxk</bpmn:outgoing>
  </bpmn:serviceTask>
  <bpmn:exclusiveGateway id="Gateway_0wr5ews"
  name="If" default="Flow_1wkq6qn">
    ...
  </bpmn:exclusiveGateway>
    ...
</bpmn:process>
```

Fig. 2: Sample model for a loan request process in BPMN following [6, p.183].

A user query with a BPMN model to be explained could be sliced into chunks in the same manner as described above. However, instead of sequentially prompting the LLM with chunk$_1$ to chunk$_n$, the parts of the process model to be explained first are prioritized by means of retrieval as follows. The k best matching nodes from the case base's index provide the index knowledge for the query, i.e. their embeddings form latent features to draw the attention of the LLM to a particular part of the BPMN model to be explained next. For this, the retrieval result is inserted as context into the prefix of the prompt chunk by chunk for each iteration through prompt chaining until an explanation for the entire process model has been generated.

4.2 RAG with explanatory cases as in-context examples

The retrieval module addresses the explanatory task itself. Process models that have already been explained are prompted to the LLM as in context-examples.

The case base comprises model-text pairs. Fig. 2 depicts the problem part of a sample case on a loan request process. The following textual explanation (created by a human expert, with omissions) forms the solution part of the sample case:

```
The "Loan Request" process begins when a customer sends a request for a loan amount via a
   message, which triggers the message start event "Receive Customer Request for Loan
   Amount". A message start event waits for receiving a message from a participant to start
   the process.
```

```
Once the request is received, the process moves to the service task "Invoke Risk Assessor".
   A service task is a task that uses some sort of service, which could be a Web service or
   an automated application.

After the risk assessment, the process reaches an exclusive gateway named "If". A diverging
   exclusive gateway (decision) is used to create alternative paths within a process flow,
   whereas only one of the paths can be taken. In this case, the gateway represents a
   decision point in the process where the flow of control will diverge based on the risk
   assessment. This gateway thereby defines a default flow.

If the risk is high, the process will follow the default sequence flow named "high risk",
   and reach the throwing message end event "Deny". This type of end event indicates that
   a message is sent to a participant at the conclusion of the process.

If the risk is low, ...

If the loan is large or the risk needs review, ...

After the loan approval process, the process reaches the throwing message end event "Return
   Approval Response".

In summary, this process model represents a loan request process where the loan request is
   received, the risk is assessed, a decision is made based on the risk assessment, and the
   decision is communicated back to the customer.
```

The explanation is a textual process description that coherently describes all model elements (activities, gateways, events, etc.) and their relationship along the displayed control flow in the given BPMN model. Additionally, it provides explanations of the model elements' semantics, given the underlying defined business need to explain BPMN models to users lacking expertise on BPMN notations. In the above sample, the semantics of the BPMN element types "message start event", "service task" - a special type of activity which is executed by a service, "exclusive gateway", and "throwing message end event" is explained.

The retrieval module of the RAG provides the k best matching cases to the user query. The response is generated in an iterative way following a recommendation from LlamaIndex. [3] First, the best matching case is inserted as context into the prompt template, along with the query, to generate an initial answer. This answer, the query and the second best matching case is then put into a "refine prompt" to generate a refined answer. The refine prompt is applied $k-1$ times until all cases from the retrieval result have been used as context information.

5 Experiments

The two case-based RAG approaches have been evaluated with a sample repository of BPMN process models. A human expert determined values for some evaluation criteria from the experimental results. At the end of this section, some observations and lessons learned from the experimental runs are discussed.

[3] https://docs.llamaindex.ai/en/stable/api_reference/query/response_synthesizer.html, last access: 03-18-2024

5.1 Experimental data

The data underlying the experiments comes from two repositories of process models in BPMN taken from two Master theses [6, 10]. The process models describe real business processes from different organisations and application fields, including human resource processes, insurance processes, and production processes. 37 BPMN models have been selected to achieve examples of various model complexity in terms of number of tokens, number of model elements and types of elements. All BPMN flow objects, i.e. events, activities and gateways are counted as model elements. BPMN types such as an XOR split gateway and an AND split gateway are counted as types of elements.

A pre-processor slightly abbreviates the XML files exported from BPMN modeler tools to save tokens as follows. The graphical structuring parts as well as the diagram interchange information are cut off from the XML files. Those parts are specified by means of tool-specific XML tags like `<extensionElements> ... </extensionElements>` in BizAgi or `<bpmndi : BPMNDiagram> ... </bpmndi : BPMNDiagram>` in Camunda. That is, the pre-processing of the user query requires a tool-specific filter per modeler tool.

The entire experimental data set can be accessed through a GitHub repository. [4]

5.2 Evaluation criteria and performance metrics

We have defined evaluation criteria to assess the experimental results in a semi-automated manner. Some performance metrics are specified based on the well-known precision and recall formulas introduced by Lesk and Salton in the field of information retrieval in the sixties [15]. The precision metrics identify "hallucinations" by the LLM, which may describe model contents that actually do not occur in the given BPMN model.

Two pairs of metrics aim to measure the accuracy the explanatory texts achieve in terms of covering the model elements `el` and the types of elements `ty` that occur in a process model. A model element mentioned in the explanation, such as the activity "Invoke Risk Assessment", is considered relevant if it occurs also in the BPMN model. Irrelevant elements are those that have been hallucinated by the LLM. The accuracy metrics are specified as follows.

$$precision_{el} = \frac{|relevant\ el\ in\ explanation|}{|el\ in\ the\ explanation|} \qquad recall_{el} = \frac{|relevant\ el\ in\ explanation|}{|el\ in\ the\ model|} \tag{1}$$

$$precision_{ty} = \frac{|relevant\ ty\ in\ explanation|}{|ty\ in\ the\ explanation|} \qquad recall_{ty} = \frac{|relevant\ ty\ in\ explanation|}{|ty\ in\ the\ model|} \tag{2}$$

Further, the extent to which users can easily follow the generated explanation strongly depends on the order of explanatory passages within the text. Thus, the next pair of metrics measures whether the model elements are described at the expected position (`pos`) within the text following the control flow in the given

[4] https://github.com/hoerb1337/LLMmodel2text/tree/main/Data_set, last access: 03-25-2024

BPMN model. The performance metrics for the correct positions are specified as follows.

$$
\begin{aligned}
precision_{pos} &= \frac{|relevant\ el\ in\ explanation\ at\ expected\ position|}{|el\ in\ the\ explanation|} \\
recall_{pos} &= \frac{|relevant\ el\ in\ explanation\ at\ expected\ position|}{|el\ in\ the\ model|}
\end{aligned}
\tag{3}
$$

The last pair of formulas measures the additional explanations of the model type's semantics, assuming to facilitate users' understanding of the BPMN model. For instance, the following sentence belongs to these meta model sentences.

```
A diverging exclusive gateway (decision) is used to create alternative paths within a
process flow, whereas only one of the paths can be taken.
```

We assess the number of explanatory sentences on the meta model of BPMN as follows.

$$
\begin{aligned}
precision_{meta} &= \frac{|relevant\ meta\ model\ sentences\ in\ explanation|}{|meta\ model\ sentences\ in\ the\ explanation|} \\
share_{meta} &= \frac{|relevant\ meta\ model\ sentences\ in\ explanation|}{|ty\ in\ the\ model|}
\end{aligned}
\tag{4}
$$

Please note that $share_{meta}$ is not a metric since it may yield a ratio value above 1.0, potentially indicating longer explanations provided per type of flow object.

5.3 Implementation with LlamaIndex

The implementation of the two RAG approaches described in Sections 4.1 and 4.2 is based on an open source orchestration framework for RAG systems called *LlamaIndex* [5] [17]. We have used LlamaIndex to bind the general purpose embedding model *text-embedding-ada-002* [6] by OpenAI to the retriever module to implement the similarity measure within LamaIndex' pre-defined Retriever Query Engine. We bind *chat-GPT4* [7] to the generator module of our RAG system. The tokenizer library tiktoken[8] is integrated to split the texts from the user query and the particular cases into tokens. The interaction between the retriever and the generator is implemented by a LlamaIndex module called *Response Synthesizer*. The latter is able to perform prompt chaining at its core [17]. It allows to feed in any LLM input prompt, get back retrieved context and send knowledge-augmented prompts to the generator module which are created by customized prompt templates. The templates and the configurations of all modules, including the number of cases to be retrieved, the chunk size, the temperature of the generator module (to specify the trade-off in GPT models between creativity and reproducability) are specified in Python scripts.

OpenAI recommends prompting tactics when engineering prompt templates for their GPT models. We have implemented the following model-specific tactics for GPT-4. Delimiters like triple quotation marks or XML tags indicate distinct sections of the prompt to be treated differently. Second, the steps required to complete a task are specified in a step-by-step instruction. Third, the LLM is

[5] for technical details, see also https://docs.llamaindex.ai/, last access: 03-25-2024

[6] https://platform.openai.com/docs/guides/embeddings, last access: 03-25-2024

[7] https://openai.com/gpt-4, last access: 03-25-2024

[8] https://cookbook.openai.com/examples/how to count tokens with tiktoken, last access: 03-20-2024

asked to adopt a persona to set its behavior throughout the conversation. The OpenAI API reference for GPT-4[9] provides a list of pre-defined messages for chat completion. A *system message* describes the perspective that the LLM should consider when generating text, for instance the perspective of a persona like a process modeling expert. Such a system message is followed by a *user message* providing the actual request to respond to. Technically, orchestration frameworks like LlamaIndex can call the OpenAI API via Python code with an array of message objects where each object has a role and content.

We have specified two different templates for the initial (`Text QA Prompt`) and the refine prompts (`Refine Prompt`) per RAG approach. As an example, the Python code for the refine prompt of approach b) (see Section 4.2) is depicted in the following.

```
# Refine Prompt
chat_refine_msgs = [
    ChatMessage(
        role=MessageRole.SYSTEM,
        content="""You are an expert Q&A system with expert knowledge on the business process
        modeling language BPMN. You will get presented an exemplary textual process
        description for BPMN model "Example", which you should use as an example for the
        generation of textual process descriptions for given BPMN model "Explain" serialised
        in XML. Additionally, you will get presented your initially generated textual process
        description for the BPMN model "Explain". You therefore strictly operate in two
        modes when refining existing answers:\n1.

        **Add** new information to the original answer, using the exemplary textual process
        description.\n2.

        **Repeat** the original answer, if the exemplary textual process description is not
        useful.\n

        Never reference the original answer or the exemplary textual process description
        directly in your answer. When in doubt, just add to the original answer."""
    ),
    ChatMessage(
        role=MessageRole.USER,
        content=(
            """Please create a textual process description for the given BPMN model
            serialised in XML. Add to each type of BPMN element used in the BPMN model a short
            explanation of this element type’s semantics.

            Exemplary textual process description for BPMN model "Example": {context_msg}
            BPMN model "Explain" serialised in XML: {query_str}
            Initial textual process description for BPMN model "Explain": {existing_answer}
            Adapted textual process description for BPMN model "Explain": """
        ),
    ),
]
```

Template variables are filled with the case from the retrieval result to be sent along as context in the current prompting iteration (`context_msg`), the user query ({`query_str`}) and the intermediate answer from the previous call (`existing_answer`) to be refined. In the system role, the LLM is instructed to behave in a step-by-step manner through the generation of a refined answer.

[9] https://platform.openai.com/docs/api-reference, last access 03-18-2024

5.4 Experimental setup

We have conducted experiments for both approaches using the LlamaIndex implementation described above in Section 5.3. We have chosen a size of 4096 tokens per chunk with a chunk overlap of 800 tokens. The tokens have been created from the XML files following the cl100k_base encoding standard. [10] The chunks are turned into embedding vectors with a maximum size of 8,191 tokens, aligning with GPT-4's context window.

The two case bases have been created from the experimental data as follows. For approach a), the five BPMN models with the highest number of tokens have been selected as queries. Four of them are larger than the context window of 8,191 tokens itself. For the fifth case it is very likely that the template (see previous subsection) filled with the corresponding XML file and any in-context example exceeds the context window size. For approach b) the 33 BPMN models that fit into the context window size have been extended by explanatory texts created by a human expert. Six of them have been arbitrarily selected to form the case base. The remaining 27 are used as query cases for approach b). As a baseline to approach b), each query has been prompted to a zero-shot approach with a template using a system message and a straightforward output instruction. Approach b) has been run in three configurations for $k = 1, 2, 3$. Each run (except for the zero-shot baseline approach) is replicated once to observe the impact of the remaining creativity of the LLM despite the temperature value of 0 is on the experimental results.

5.5 Experimental results

The experimental results for approach a), which tries to overcome the restricted context window size by means of prompt chaining, show a mixed picture. One round for model 8-5 generated output that exhibited a very low performance, reflecting the reason for not evaluating this textual process description. Table 1 depicts the F1 score values of the metric pairs. Especially the two rows for the values of $share_{meta}$ highlight that our approach was not successful in explaining the semantics of BPMN.

The experimental results for approach b), which uses cases from the case base as in-context examples, are depicted as average values per run in Table 2. The pairs of metrics are combined into F1 scores again. Instead of the 27 models as planned, only 22 models have been considered. The computational time increased tremendously with model complexity, actually requiring a few minutes for only one larger BPMN model with around 5013 tokens and 72 model elements. The quality of the explanations of the larger models decreased rapidly (compare the discussion in Section 5.6). Since it would have caused costs of more than 3 US Dollar for generating a single textual process description, we decided to consider only models with less than 5k tokens.

[10] as implemented in OpenAI's tokenizer library tiktoken, see https://cookbook.openai.com/examples/how_to_count_tokens_with_tiktoken, last access: 20-03-2024

Table 1: Experimental results for approach a) on the five largest process models.

	Model 10-2	Model 2-2	Model 8-4	Model 8-5	Model 2-1	Average
No. of tokens	7,018	8,238	8,607	8,634	10,886	8,677
$F1_{el}$	0.6120	0.6283	0.4138	-	0.6227	0.5692
(repeated)	0.6736	0.6	0.4161	0.4824	0.4576	0.5259
$F1_{ty}$	0.6667	0.3571	0.88	-	0.3333	0.5593
(repeated)	0.7143	0.3077	0.7273	0.6316	0.2857	0.5333
$F1_{pos}$	0.6180	0.6147	0.0299	-	0.5208	0.4458
(repeated)	0.5574	0.4356	0.058	0.519	0.2895	0.3719
$share_{meta}$	0.1224	0.04	0.2292	-	0.0169	0.1021
(repeated)	0	0.02	0.1458	0	0	0.0332
$precision_{meta}$	0.8571	1	0.9167	-	1	0.9435
(repeated)	0	1	1	0	0	0.4

All RAG rounds achieved better values than the baseline LLM. The differences between two runs in the same configuration are rather small. Surprisingly, the improvements with an increasing number k of in-context examples are marginal on average.

Table 2: Experimental results for approach b) (avg. for 22 models per run).

	LLM (zero-shot)	RAG with k=1	RAG with k=2	RAG with k=3
$F1_{el}$	0.8130	0.8855	0.8790	0.8713
(repeated)	-	0.8646	0.8521	0.8736
$F1_{ty}$	0.7193	0.8676	0.8652	0.8698
(repeated)	-	0.8509	0.8574	0.8625
$F1_{pos}$	0.8071	0.8251	0.8234	0.8301
(repeated)	-	0.8353	0.7995	0.8402
$share_{meta}$	0.2289	0.5575	0.4780	0.5757
(repeated)	-	0.5695	0.4881	0.5556
$precision_{meta}$	0.6307	0.9811	0.9813	0.9924
(repeated)	-	0.9430	0.9738	0.9451

5.6 Lessons learned

In addition to the measured values, we made some surprising observations in the course of the experiments.

Setting the temperature parameter to 0 did not prevent the RAG from generating variants of output in the different experimental runs. Note that at the end of our work, OpenAI released an updated version of GPT-4 that now includes additional parameters to control non-deterministic behavior.[11] Since different

[11] see https://openai.com/blog/new-models-and-developer-products-announced-at-devday, last access: 03-26-2024

explanations for the same model might even be a benefit for the intended users it is not yet clear whether the additional parameters were useful.

Tests with inserting meta knowledge on BPMN into the prompt instructions did not change the quality of results. This is in contrast to the results reported by Fill et al. [5] who experimented with an older version of GPT. It seems that the meta knowledge on BPMN is an inherent part of GPT-4 already since the training set might have included process model descriptions in BPMN.

The performance decrease with increasing model complexity was steady for any prompting strategy. In particular, we got a sense of increasing difficulty to map the described model elements and their relationships to the original BPMN model. For larger models, typically only parts of the process are correctly described while others - at random locations in the process model - did achieve explanations that are not useful. For instance, the LLM condensed several model elements into a high-level description instead of describing each of them. This makes it very difficult for novice users to understand the process model on the basis of the explanation. There seems to be a hidden capability threshold of somewhere around 3800 tokens which is less than half of GPT-4's context window of 8,191 tokens.

This inherent threshold might also provide a reason for the following, a bit disappointing observation. Increasing the number of in-context examples k achieves only marginal improvements compared to $k = 1$. In the light of energy consumption, the retrieval of a single best matching case seems the most recommendable strategy at the moment to configure the RAG.

6 Conclusion

In this paper, we have presented a RAG system based on the framework LlamaIndex and LLM GPT-4 with the aim to generate explanatory texts describing business process models for users who are not familiar with process modeling. Two alternative approaches of case-based support have been developed. While approach a) addresses reusing the partition of process cases into chunks for prompt chaining approach b) retrieves model-text-pair cases as in-context examples. The experiments for approach a) fail presumably at the limit of current LLM's capability to deal with larger context information. The experiments for approach b) succeed in generating explanations that cover the particular model elements, their control flow as well as the semantics of the BPMN modeling language. Surprisingly, the quality of the results are of nearly the same quality for configurations of one up to three in-context examples retrieved from the case base.

Meanwhile, GPT-4 provides pre-view versions with a context window size of 128k tokens.[12] It is likely that the capability threshold reported above is shifted beyond the 11k tokens of our largest model from the experimental repository.

[12] see https://openai.com/blog/new-models-and-developer-products-announced-at-devday, last access: 03-26-2024

Repeating the experiments for both approaches, prompt case-based chaining (approach a)) and the use of explanatory cases as in-context examples (approach b)) with the new models seems promising. We think that the current experimental results on a case-based memory for model-text-pairs (approach b)) demonstrate the feasibility of using CBR retrieval in RAG systems for generating text explanations of structural models.

References

1. Bellan, P., Dragoni, M., Ghidini, C.: Leveraging pre-trained language models for conversational information seeking from text (Mar 2022), http://arxiv.org/abs/2204.03542, arXiv:2204.03542 [cs]
2. Brown, T.B., Mann, B., Ryder, N., Subbiah, M., Kaplan, J., Dhariwal, P., Neelakantan, A., Shyam, P., Sastry, G., Askell, A., Agarwal, S., Herbert-Voss, A., Krueger, G., Henighan, T., Child, R., Ramesh, A., Ziegler, D.M., Wu, J., Winter, C., Hesse, C., Chen, M., Sigler, E., Litwin, M., Gray, S., Chess, B., Clark, J., Berner, C., McCandlish, S., Radford, A., Sutskever, I., Amodei, D.: Language Models are Few-Shot Learners. arXiv:2005.14165 [cs] (Jun 2020), arXiv: 2005.14165
3. Davenport, T.H.: Process innovation: reengineering work through information technology. Harvard Business Press (1993)
4. Decker, G., Dijkman, R., Dumas, M., García-Bañuelos, L.: The Business Process Modeling Notation. In: Hofstede, A.H.M., Aalst, W.M.P., Adams, M., Russell, N. (eds.) Modern Business Process Automation, pp. 347–368. Springer Berlin Heidelberg (Jan 2010)
5. Fill, H.G., Fettke, P., Köpke, J.: Conceptual Modeling and Large Language Models: Impressions From First Experiments With ChatGPT. Enterprise Modelling and Information Systems Architectures (EMISAJ) **18**, 3–1 (2023)
6. Friedrich, F.: Automated Generation of Business process Models from Natural Language Input. Diplomarbeit, Humboldt-Universität zu Berlin (2010)
7. Grohs, M., Abb, L., Elsayed, N., Rehse, J.R.: Large Language Models Can Accomplish Business Process Management Tasks. In: De Weerdt, J., Pufahl, L. (eds.) Business Process Management Workshops. pp. 453–465. LNBIP 492, Springer Nature Switzerland, Cham (2024)
8. Hammond, K., Leake, D.: Large language models need symbolic AI. In: Proceedings of the 17th International Workshop on Neural-Symbolic Learning and Reasoning, La Certosa di Pontignano, Siena, Italy. vol. 3432, pp. 204–209 (2023), https://ceur-ws.org/Vol-3432/paper17.pdf
9. Hess, T., Matt, C., Benlian, A., Wiesböck, F.: Options for Formulating a Digital Transformation Strategy. MIS Q. Executive **15**(2), 123–139 (2016)
10. Hoffmann, D.: Ontology-based transfer learning for the digitization of administrative processes. Master thesis, Goethe University, Frankfurt (2022)
11. Kendall-Morwick, J., Leake, D.: A Study of Two-Phase Retrieval for Process-Oriented Case-Based Reasoning. In: Montani, S., Jain, L.C. (eds.) Successful Case-based Reasoning Applications-2, vol. 494, pp. 7–27. Springer Berlin Heidelberg, Berlin, Heidelberg (2014)
12. Khattab, O., Potts, C., Zaharia, M.: Building Scalable, Explainable, and Adaptive NLP Models with Retrieval (Oct 2021), https://ai.stanford.edu/blog/retrieval-based-NLP/, last access: 04-24-2024

13. Klievtsova, N., Benzin, J.V., Kampik, T., Mangler, J., Rinderle-Ma, S.: Conversational Process Modelling: State of the Art, Applications, and Implications in Practice. In: Di Francescomarino, C., Burattin, A., Janiesch, C., Sadiq, S. (eds.) Business Process Management Forum. pp. 319–336. LNBIP 490, Springer Nature Switzerland, Cham (2023)
14. Leopold, H., Mendling, J., Polyvyanyy, A.: Supporting process model validation through natural language generation. IEEE Transactions on Software Engineering **40**(8), 818–840 (2014), publisher: IEEE
15. Lesk, M.E., Salton, G.: Relevance assessments and retrieval system evaluation. Information storage and retrieval **4**(4), 343–359 (1968), publisher: Elsevier
16. Lewis, P., Perez, E., Piktus, A., Petroni, F., Karpukhin, V., Goyal, N., Küttler, H., Lewis, M., Yih, W.t., Rocktäschel, T.: Retrieval-augmented generation for knowledge-intensive NLP tasks. Advances in Neural Information Processing Systems **33**, 9459–9474 (2020)
17. Liu, J.: LlamaIndex (11 2022). https://doi.org/10.5281/zenodo.1234, https://github.com/jerryjliu/llama_index, last access: 04-23-2024
18. Liu, P., Yuan, W., Fu, J., Jiang, Z., Hayashi, H., Neubig, G.: Pre-train, Prompt, and Predict: A Systematic Survey of Prompting Methods in Natural Language Processing. ACM Computing Surveys **55**(9), 1–35 (Sep 2023)
19. Minor, M., Herold, M., Rubbe, J., Dufner, S., Brussas, G.: Transfer Learning Operators for Process-oriented Cases. In: Proc. 2021 IEEE Fourth International Conference on Artificial Intelligence and Knowledge Engineering AIKE 2021. pp. 9 – 16. IEEE Computer Society Press (2021)
20. Minor, M., Montani, S., Recio-García, J.A.: Editorial: Process-oriented Case-based Reasoning. Inf. Syst. **40**, 103–105 (Mar 2014)
21. Minor, M., Tartakovski, A., Bergmann, R.: Representation and Structure-based Similarity Assessment for Agile Workflows. In: Weber, R.O., Richter, M.M. (eds.) Case-Based Reasoning Research and Development, 7th International Conference on Case-Based Reasoning, ICCBR 2007, Belfast, Northern Ireland, UK, August 2007, Proceedings. pp. 224 – 238. LNCS 4626, Springer (2007)
22. Radford, A., Narasimhan, K., Salimans, T., Sutskever, I.: Improving language understanding by generative pre-training. Tech. rep. (2018), https://openai.com/research/language-unsupervised, last access: 04-23-2024
23. Schultheis, A., Hoffmann, M., Malburg, L., Bergmann, R.: Explanation of Similarities in Process-Oriented Case-Based Reasoning by Visualization. In: Massie, S., Chakraborti, S. (eds.) Case-Based Reasoning Research and Development. pp. 53–68. LNCS 14141, Springer Nature Switzerland, Cham (2023)
24. Upadhyay, A., Massie, S.: CBR Assisted Context-Aware Surface Realisation for Data-to-Text Generation. In: Massie, S., Chakraborti, S. (eds.) Case-Based Reasoning Research and Development. pp. 34–49. LNCS 14141, Springer Nature Switzerland, Cham (2023)
25. Vidgof, M., Bachhofner, S., Mendling, J.: Large Language Models for Business Process Management: Opportunities and Challenges. In: Di Francescomarino, C., Burattin, A., Janiesch, C., Sadiq, S. (eds.) Business Process Management Forum. pp. 107–123. Lecture Notes in Business Information Processing, Springer Nature Switzerland, Cham (2023)
26. Watson, I.: A case-based persistent memory for a large language model. CoRR **abs/2310.08842** (2023). https://doi.org/10.48550/ARXIV.2310.08842, https://doi.org/10.48550/arXiv.2310.08842
27. Weske, M.: Business Process Management Concepts, Languages, Architectures. Springer Berlin Heidelberg, Berlin, Heidelberg, 2nd edn. (2012)

28. Willcocks, L.P., Lacity, M., Craig, A.: The it function and robotic process automation. Tech. rep. (2015)
29. Wu, T., Terry, M., Cai, C.J.: AI Chains: Transparent and Controllable Human-AI Interaction by Chaining Large Language Model Prompts. In: CHI Conference on Human Factors in Computing Systems. pp. 1–22. ACM, New Orleans LA USA (Apr 2022). https://doi.org/10.1145/3491102.3517582, https://dl.acm.org/doi/10.1145/3491102.3517582
30. Zeyen, C., Müller, G., Bergmann, R.: Conversational Process-Oriented Case-Based Reasoning. In: Aha, D.W., Lieber, J. (eds.) Case-Based Reasoning Research and Development. pp. 403–419. LNCS 10339, Springer International Publishing, Cham (2017)