Discovering Declarative Processes in Textual Descriptions using Large Language Models

Jonas M. Lindner and Hugo A. López

Technical University of Denmark,
Richard Petersens Plads, 321, 2800 Kgs. Lyngby, Denmark
{jmali,hulo}@dtu.dk

Abstract. We explore whether GenAI techniques can improve the extraction of declarative process models as DCR graphs. Compared to imperative process models, declarative process models capture rules and constraints, and mapping their semantics from textual specifications is far from trivial. Previous works have explored text mining and pattern-matching techniques for the extraction of declarative processes from sentences, while the extraction of full specifications is considered a harder problem. We investigate whether prompt engineering techniques and large language models can enhance the performance of this task, comparing them with existing approaches. First, we introduce an LLM-based architecture for the extraction of declarative process models as DCR graphs. Second, we compare the extraction against a manually generated gold standard of process models, and third, we demonstrate how the techniques compare to previous extraction pipelines. Our initial evaluations show that LLM-based pipelines provide promising results in the discovery of events and roles, but constraint mining still requires more accurate methods to be used as an automated technique.

Key words: Declarative Process Models, DCR graphs, Text-to-Process extraction, LLMs

1 Introduction

Process models are crucial artifacts in the understanding, analysis, communication, optimization, and automation of processes [1]. To discover a process, multiple data sources need to be integrated: models derived from event logs provide a picture of the running process *as is*. In contrast, processes need to abide by constraints and regulations. These come in the shape of process descriptions, service operating procedures (SOPs), or clinical guidelines, and they are the reference description of the process models *as they should be*. The mining of process models from normative texts provides reference models that can be compared with the process execution [2], facilitating organizational compliance tasks.

The manual elicitation of models is a complex problem since the identification of process constructs in text requires an understanding of the semantics of the modelling language, as well as advanced techniques to resolve ambiguities in text [1]. Starting with the seminal works of Friedrich [3], the discovery of process models via natural language processing (NLP) has been a topic of interest for over a decade. With the introduction to Generative AI (GenAI) as a technique in Business Process Management (BPM) [4], GenAI has become a promising tool to simplify some NLP tasks involved in the generation of models from texts.

This paper investigates whether GenAI could leverage the task of process generation of declarative process models from business process descriptions. While a great deal of attention has been given to the extraction of imperative process models (such as BPMN [5]), few works have explored the application of GenAI to declarative processes [6]. Compared to [6], both the sources, methodology, and target languages of our exploration differ. We extend our analysis to document-level elicitation (instead of sentence-level in [6]), define an evaluation methodology based on annotation guidelines for declarative models, and generate declarative process models as DCR graphs [7], which, compared to Declare, is expected to have lower cognitive load than Declare, as it has fewer modelling constraint patterns and it is currently used for larger digitalization projects in the public sector [8]. Specifically, this paper explores the following questions:

– RQ1: Can one use prompt engineering to generate DCR graphs from business process specifications?
– RQ2: To what degree of correctness do prompt engineering techniques capture the extraction of declarative process models compared to manual annotations?
– RQ3: How does the extraction of declarative process models compare against classical Text-to-Process techniques used in DCR graphs?

To answer RQ1, we propose multiple prompt-engineering pipelines that receive a process specification and generate a fully-compliant DCR graph. To answer RQ2, we created a gold-standard set of annotated process descriptions that was later used to compare the correctness of the DCR models generated by the different prompt techniques. Finally, RQ3 compared against the existing NLP approaches to extract DCR graphs from process descriptions, showing up to 50% improvement in the precision compared to the state of the art.

The remainder of this paper is structured as follows: Section 2 presents related work. Section 3 provides background on DCR graphs and NLP techniques used in process extraction with Large Language Models (LLM). Our process discovery pipelines are introduced in Section 4. The internal validation is provided in Section 5. The external validation is discussed in Section 6. Section 7 concludes the paper and presents future work.

2 Related Work

Text-to-Process extraction pipelines are an established task in Business Process Management, and we refer the reader to [9] for a recent survey on NLP techniques applied in this area. Earlier works [3, 10, 11] decomposed the problem into multiple standard NLP tasks such as Named Entity Recognition, Part of Speech and Relationship Extraction, they have struggled to generalize to different linguistic patterns, writing styles, and ambiguities present in process descriptions [12, 13].

Due to recent advances in language models, LLMs have been utilized to support the discovery process and facilitate the setup of extraction pipelines [14, 4, 5], including prompt engineering approaches [15, 6]. Their application has so far been targeted at sequential descriptions and imperative process models. To the best of our knowledge, the extraction of declarative constraints via LLMs has only been addressed by Grohs et al. for the Declare language [6]. For DCR graphs, we are only aware of the works in [11, 16], which will be addressed in Section 6.

3 Background

We will recall the concepts behind DCR graphs and LLMs used in this work.

3.1 DCR graphs

A DCR graph is a process modelling notation used in the formalisation and digitalisation of knowledge-intensive processes [8]. In this paper, we use DCR graphs as a graphical notation and a formal modelling language. We refer the reader to [7] for its formal semantics. We will illustrate the language via the example in Figure 2. The graph includes a set of labelled events (boxes), representing an event or activity that is happening or performed during the process execution. Events can be assigned to a set of roles, defining access control policies and collaboration constraints. The edges between events are described by the different typed constraints in DCR graphs. We use the types introduced in [17]: (1) Condition e f: Represents that f cannot be executed until e has been executed, or e is excluded; (2) Response e f: Denotes that when e executes, f becomes pending and must eventually occur or be excluded; (3) Dynamic inclusion e f: Says that after executing event e, event f is included among the possible actions to take; (4) Dynamic exclusion e f: Represents that after executing event e, event f is excluded from the possible actions to take; (5) Finally, milestone e f: Says that if e is pending, f cannot be enabled.

3.2 Large Language Models and Prompt Engineering

The creation of LLM leverages deep learning techniques that exploit the construction of neural networks on dozens of layers with billions of parameters. The large amount of data enables LLMs to process natural language and generate expressive responses by determining successive words based on the likelihood of their occurrence. To support the LLM in its tasks, *in-context learning* [18] and *retrieval augmented generation (RAG)* [19] techniques can be applied. In-context learning provides the LLM with additional context to understand the task as part of the given prompt. RAG extracts information from available documents. Both have been shown to enhance the correctness of the outputs [18, 19].

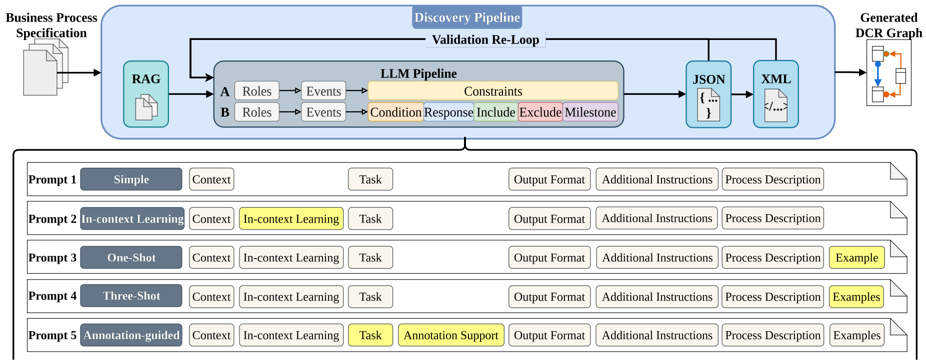


Fig. 1: Components of discovery pipeline and prompts used in the experiments

4 A Prompt Engineering Architecture for DCR graphs

We answer RQ1 by building an LLM architecture used to generate DCR graphs from business process specifications (see Figure 1). This section presents the components that comprise the graph **discovery pipeline**.

LLM requests build the core of the discovery pipeline. In each run, the components employ the same prompt type. The **LLM pipeline** employs a bottom-up approach, identifying the simplest concepts first and utilizing them to construct more complex ones. In our case, we first identify active roles and then their corresponding events. Later, we identify the constraints between the discovered events. We use two different LLM pipelines. One (*pipeline A*) identifies all constraints at once, and the other (*pipeline B*) requires the LLM to find all constraint types individually, demonstrating the effects of task breakdown.

Five **prompts** were defined with gradually increasing details. Their design has been influenced by related work [15]. Figure 1 shows the components that are part of each prompt. The highlighted components were iteratively added or changed in the design process. *Prompt 1* is considered a simple one, that only includes: the role and knowledge of the LLM (*Context*), the discovery task (*Task*), the expected output format (*Output Format*), dos and do-nots that shall guarantee steady outputs (*Additional Instructions*), and process specification (*Process Description*). In contrast, *Prompt 2* includes in-context learning with DCR definitions. *Prompt 3* adds a correct example to the prompt, which is based on extracts of the process specifications (*Example*). For *Prompt 4*, another correct and one false example are provided. At last, *Prompt 5* additionally asks the LLM to consider a set of annotation guidelines that have previously been defined for circumstantial information in the legal sector. All prompts and documents can be found in our repository[1].

[1] https://github.com/lindner-jonas/discovery_DCR_from_Text

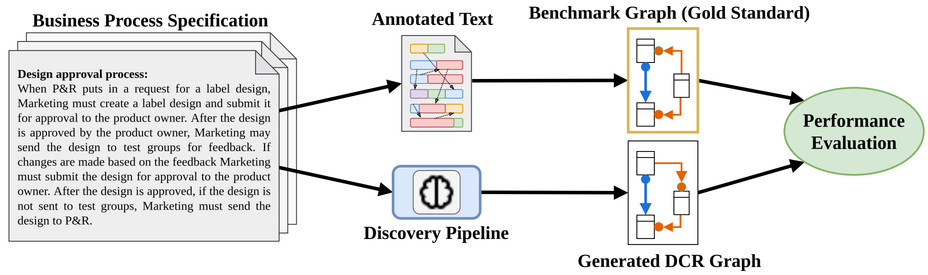


Fig. 2: Approach for evaluating discovered DCR graphs against annotated business process specifications (process description for ”10_Design_approval”; see repository)

The LLM pipeline is only used to discover DCR elements, but does not perform the DCR graph generation, which in our case requires the creation of an XML file. As we experienced difficulties generating consistent outputs in XML format, we instructed the LLM to embed its reply in a simple JSON structure that we can convert into XML. Thus, the architecture includes an **output validation loop** to guarantee that the LLM output matches the expected format before further processing. When an invalid generation is found, the whole LLM pipeline is run repeatedly until a valid graph is successfully generated (or aborted after a parameterised number of unsuccessful attempts).

With the help of **retrieval augmented generation** – applied to prompts 2, 3, 4, and 5 – one can retrieve additional information from documents and support the LLM by providing more context. The retriever uses two documents: The first includes basic knowledge about DCR and its entities. The second document includes the annotation guidelines for legal specifications, which have been used in constructing the gold standard for our evaluation (see Section 5.1). Throughout the use of larger documents, the context length of the prompts might substantially increase.

5 Internal Validity

We conducted an internal validation of the discovery pipeline to answer RQ2. This validation is based on a comparison between generated DCR specifications and our gold standard, following the approach presented in Figure 2. This section describes the experimental setup and results of the internal validation.

5.1 Methods and Materials

The experimental setup includes the definition of a gold standard, the selection of LLMs, the definition of metrics to perform the evaluation and data generation.

Entity	**Sentences**	**Words**	**Roles**	**Events**	**Constraints**
Total	121	2,272	41	215	188
Average per sentence*		18.40	0.39	1.88	1.54

$*\overline{x} = \frac{\sum_{i=0}^{N} \frac{x_i}{sentences_i}}{N}$, for x = Entity and $N = 10$ process specifications

Table 1: Numerical distribution of identified entities in the ten annotations

Model	**Parameter Size**		**Context-Length for Prompt N°**				
			1	**2**	**3**	**4**	**5**
Llama3.1	8 B	4.7 GB	2,500	5,000	5,000	5,000	2,0000
Llama3.2	3 B	2 GB	2,500	5,000	5,000	5,000	2,0000

Table 2: LLM used for AI-assisted discovery

Annotation of Process Descriptions A gold standard was required to assess the correctness of the LLM generation pipelines. Thus, we have annotated ten process descriptions. The used annotation guidelines were based on the ones for the legal sector and have been adapted for business process descriptions. The original texts are a subset of the dataset of process descriptions in [13] and represent a diverse set of different types of descriptions (smaller – larger; more procedural – less procedural; written by students – written by more advanced modellers). An overview of the distribution of identified entities in our gold standard[2] as well as the number of words and sentences is presented in Table 1.

Choice of Models The requirements we set out for our experiment included the demands for easy reproducibility as well as assessment of the usability of LLMs in local open-source modelling environments, like DCR-js [20]. For that reason, the models that we used in the experiments are Llama3.2 and Llama3.1[21], and are presented in Table 2. We consider them to be small-sized LLMs that one can run without excessive computational resources, storage usage, or paid API access at hand. The context lengths were selected based on the input token lengths to minimize the number of unused tokens.

Similarity Thresholds We must account for variations in the output of the LLMs. Identified labels might be syntactically different (strings do not match), but their semantic meaning is the same. For instance, a role might be introduced as *"Computer repair service (CRS)"*, and later be referred to by the alias *"CRS"*. Even though the syntactic similarity is small, their semantic similarity might be higher. We must therefore determine how similar the generated output is to our gold standard and decide on its semantic correctness based on a given similarity threshold. These thresholds were used in the performance evaluation.

The determination of thresholds made use of a library for semantic textual similarity calculation (SBERT) [22]. SBERT creates a fixed-length vector representation for each entity and calculates their similarity based on the cosine distance. Finding the thresholds involved a setup that compared different variations of aliases and co-references, but also conceptually different entities, from

[2] `https://github.com/lindner-jonas/discovery_DCR_from_Text/tree/main/2_Gold_Standard`

the annotated process descriptions. The following examples demonstrate the difficulty in choosing a threshold: For the example above, the similarity between the role and its alias was 0.64. The similarity between the event *"adjust the quantity"* and its co-reference *"adjust the quantity within the basket"* was 0.35, and for two different events, *"creates an expense report"* and *"approve the expense report"*, the similarity was 0.79. Because no threshold can perfectly determine all correct matches, one wants to maximize the chance of identifying true positive matches. As a result of the comparisons, the threshold for roles was decided to be 0.7, and 0.4 for events. The setup can be found in the repository as well.

Performance Metrics The performance for each prompting strategy was assessed for four types of discoveries: roles, events, role-event assignment, and constraints. Their F1-score was calculated with $2 \times \frac{Precision*Recall}{Precision+Recall}$, where $Precision = \frac{true\ positive}{true\ positive\ +\ false\ positive}$ and $Recall = \frac{true\ positive}{true\ positive\ +\ false\ negative}$. For roles and events, every generated entity and the gold standard entities were compared with each other using SBERT. If multiple generated entities exceeded the similarity threshold, only the one with the highest similarity value was matched to a gold standard entity to avoid double hits in our results. The matching generated entities were considered true-positives; the others false-positives. If a gold standard entity was not found, it was counted as false-negative. Role-event assignments and constraints were assessed as a whole between the assigned entities – combinations including unmatched events were considered false-positive.

Data Generation and Validation The annotation process resulted in ten annotated process descriptions. But to validate the results on a sufficiently large data set and, by that, ensuring their consistency, we decided to run each of the combinations of LLM, pipeline and prompt ten times for each annotated process description, resulting in 100 runs for each combination and a total of 2,000 runs of the discovery pipeline (2 LLMs $\times$ 2 pipelines $\times$ 5 prompts $\times$ 10 process descriptions $\times$ 10 runs each). Due to the non-determinism of LLM replies, every generated graph can be considered as an independent result.

5.2 Results Internal Validity

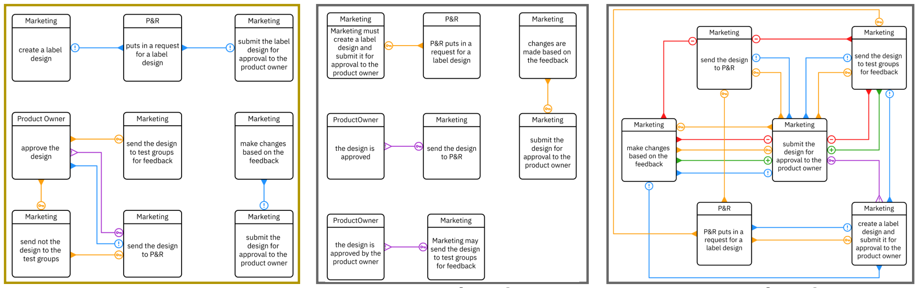
The results of the experiments for the internal validation are presented in Table 3. The experiments were run with access to five GPUs (RTXA5000, GPU VRAM 24 GB) and max. 512 GB of RAM. Each run of the discovery pipeline had a time limit of 300s, and the maximum number of failed generations was 30.

Llama3.2 had more successful runs with on average 95 out of 100 generated models, while Llama3.1 was more error-prone with only 66 models on average. For both pipelines A and B, precision and recall for roles and role-event-assignment are mostly above 0.5; the results for events are relatively consistent above 0.9 throughout all strategies. The F1-scores of Llama3.1 are almost always higher than the ones of Llama3.2. Even though the prompts increase in detail, no relevant trend for increased performance can be observed when adding information. As for the discovery of constraints, prompt 1 and 2 resulted in no

Prompt	LLM	Entity	Pipeline A					Pipeline B				
			N°	Prec.	Rec.	F1	Time	N°	Prec.	Rec.	F1	Time
1 Simple	Llama3.1	Roles	83	0.68	0.57	0.62	0.49	22	0.73	0.58	0.65	0.42
		Events		0.95	0.60	0.74	2.81		0.97	0.60	0.74	2.35
		Role+Event		0.60	0.48	0.54			0.68	0.50	0.58	
		Constraints		0.00	0.00	0.00	5.41		0.00	0.00	0.00	15.00
	Llama3.2	Roles	89	0.58	0.55	0.56	0.30	71	0.76	0.58	0.66	0.26
		Events		0.94	0.48	0.64	1.05		0.93	0.43	0.59	1.16
		Role+Event		0.46	0.31	0.37			0.53	0.30	0.39	
		Constraints		0.00	0.00	0.00	1.76		0.00	0.00	0.00	7.1
2 In-context Learning	Llama3.1	Roles	100	0.69	0.55	0.62	1.03	74	0.80	0.62	0.70	0.98
		Events		0.97	0.57	0.72	4.06		0.95	0.56	0.70	3.93
		Role+Event		0.60	0.45	0.51			0.66	0.47	0.55	
		Constraints		0.00	0.00	0.00	5.24		0.00	0.00	0.00	16.24
	Llama3.2	Roles	99	0.58	0.54	0.56	0.54	96	0.54	0.50	0.56	0.53
		Events		0.94	0.49	0.65	1.80		0.95	0.51	0.66	1.84
		Role+Event		0.43	0.30	0.36			0.42	0.31	0.36	
		Constraints		0.00	0.00	0.00	2.18		0.00	0.00	0.00	6.18
3 One-Shot	Llama3.1	Roles	100	0.67	0.57	0.62	1.41	81	**0.83**	0.66	0.73	1.07
		Events		**0.99**	**0.63**	**0.77**	4.53		0.98	0.59	0.74	4.68
		Role+Event		0.66	0.53	0.58			**0.76**	**0.76**	**0.62**	
		Constraints		0.03	0.02	0.02	6.13		0.01	0.02	0.02	18.47
	Llama3.2	Roles	100	0.57	0.54	0.56	0.67	100	0.57	0.54	0.55	0.57
		Events		0.90	0.56	0.69	1.96		0.91	0.56	0.69	2.02
		Role+Event		0.53	0.43	0.48			0.51	0.41	0.46	
		Constraints		0.01	0.01	0.01	2.32		0.01	0.01	0.01	6.47
4 Three-Shot	Llama3.1	Roles	100	0.67	0.57	0.62	1.09	85	0.75	0.61	0.67	1.10
		Events		**0.99**	0.60	0.75	4.51		**0.99**	0.61	0.76	5.08
		Role+Event		0.68	0.50	0.58			0.74	0.54	**0.62**	
		Constraints		0.03	0.02	0.02	6.89		0.02	**0.03**	0.02	21.52
	Llama3.2	Roles	100	0.63	0.56	0.60	0.67	98	0.56	0.50	0.53	0.59
		Events		0.93	0.54	0.68	2.45		0.92	0.54	0.68	2.02
		Role+Event		0.58	0.42	0.49			0.52	0.40	0.45	
		Constraints		0.02	0.01	0.01	4.32		0.01	0.01	0.01	6.67
5 Annotation-guided	Llama3.1	Roles	11	0.79	**0.76**	**0.77**	14.00	0	NA	NA	NA	13.77
		Events		0.93	0.58	0.71	19.18		NA	NA	NA	18.95
		Role+Event		0.74	0.52	0.61			NA	NA	NA	
		Constraints		**0.05**	0.02	**0.03**	11.86		NA	NA	NA	34.08
	Llama3.2	Roles	99	0.69	0.53	0.60	8.80	96	0.77	0.58	0.66	8.45
		Events		0.95	0.54	0.69	11.97		0.96	0.50	0.65	11.76
		Role+Event		0.60	0.42	0.50			0.68	0.41	0.51	
		Constraints		0.03	0.01	0.01	6.18		0.02	0.01	0.01	11.56

Table 3: Results of internal validation. Performance presented as averages over successfully generated graphs for the corresponding combination of model, pipeline, and prompt. Time corresponds to average response time of LLM in seconds. NA is used when all runs reached the cut-off time or maximum number of failed generations.

matching constraints (0.00). While the task breakdown in pipeline B did not increase the performance for discovering constraints, the LLM response time for the discovery of constraints increased by on average 2.8 times.



Gold Graph for process description "Design approval"

Generated Graph (Llama3.1, Pipeline A, Prompt 3)

Generated Graph (Llama3.1, Pipeline B, Prompt 3)

Fig. 3: Example graphs for process description ”10_Design_approval” (see repository)

5.3 Qualitative Analysis

The annotation guidelines helped to build an unbiased gold standard, by keeping a modeller's decision-making to a minimum. In Figure 3, we see the difference between the gold graph and the generated graphs. Even though the gold graph includes more events, the focus on explicit statements during the annotation process results in fewer constraints. In contrast, the generated graph for pipeline B includes the most constraints for two reasons: The task breakdown in pipeline B enables the LLM to discover more constraints than pipeline A by focusing on one constraint type at a time. And the LLM mimics the human ability to reveal relations between sentences, that are based on intuition and assumptions. For that reason, some false positive values might be considered correct by a process modeller, however, they are not based on guided annotations and, thus, not included in the gold standard.

The results demonstrate that LLMs are capable of extracting process elements, especially for roles and events. Their meaning is independent of DCR, which explains why prompt 2 did not much improve their scores. The high scores for events in Table 3 do seem surprising, however, result from the low similarity threshold used in the comparison with the gold standard. As a consequence, misallocation cannot be ruled out.

The prompt design phase revealed that LLMs hallucinate more often with increasing text-length, which explains almost constant performance after adding more examples in prompt 4. But precise task descriptions and prohibitions help to counteract common hallucinations, and the results show low error rates[3]. For that reason, we see potential application of the pipeline as a supportive system in modelling environments, but not in unsupervised and automated systems.

6 External Validity

In this section, we will answer RQ3 by comparing our prompt engineering techniques against other process extraction tools for DCR graphs. The literature

[3] https://github.com/lindner-jonas/discovery_DCR_from_Text/tree/main/6_Time_Analysis/data

presents two approaches. The first [11] introduces semi-automatic support for the creation of DCR graphs. Pre-trained lexical data sets are used in combination with standard parsing and classical NLP to identify language patterns and extract roles, events, and constraints from text. The second [16] uses transformer models and Named Entity Recognition to mine process elements from text fragments. Here, BERT is used to discover roles, events, and relations, and a co-reference between the text and model is built to decrease ambiguity. We compare our approach to [11], since it is the only available in the commercial offering for DCR solutions, implemented on top of the DCR highlighter [23].

6.1 Experimental Setup

The highlighter finds roles and events and assigns roles to the events. Each process description used in the internal validation (c.f. Section 5) was fed into the tool once. In order to compare the results to our discovery pipeline, which automatically generates DCR graphs without human intervention, the suggestions for roles and events have been accepted as they were. As for constraints, the tool only highlights words that suggest the presence of a constraint, but the constraint type and corresponding events have to be assigned manually. Hence, constraints have not been included in the graphs due to the absence of automation by the highlighter, to allow for true comparison with our approach.

6.2 Results External Validity

Table 4 shows the results of the external validation setup. The overall performance of the discovered entities is similar to the contribution in our internal validation, where precision for roles and role-event-assignment are, on average, lower than the ones for events (DCR Highlighter: by 0.37; LLM: by 0.30). This is expected due to the low similarity threshold for events that was chosen. The F1-scores from the Highlighter are, on average, 0.24 and 0.22 lower than the best values for usability, and precision, respectively, from the discovery with LLMs; however, the recall on roles with the Highlighter is higher, with 0.75. As mentioned before, the DCR Highlighter does not automatically discover constraints between events. Thus, we consider its task performance as 0.0. While our approach still bears a low performance in constraint identification, the general capability of automatic discovery underpins the strengths of our LLM pipeline in comparison to previous NLP approaches.

7 Conclusion and Future Work

This work reports the first application of prompt engineering to the discovery of DCR graphs from natural language business process specifications. A prompt engineering architecture including a LLM-based discovery pipeline is presented, and different prompting strategies are tested and evaluated. Internal validation

Entity	DCR Highlighter			Llama3.1 - A - 3			Llama3.1 - B - 3		
	Precision	Recall	F1	Precision	Recall	F1	Precision	Recall	F1
Roles	0.30	**0.75**	0.44	0.67	0.57	0.62	**0.83**	0.66	**0.73**
Events	0.73	0.37	0.49	**0.99**	**0.63**	**0.77**	0.98	0.59	0.74
Role-Event-Assignments	0.42	0.25	0.31	0.66	0.53	0.58	**0.76**	**0.76**	**0.62**
Constraints	0.00	0.00	0.00	**0.03**	**0.02**	**0.02**	0.01	**0.02**	**0.02**
Average	0.36	0.34	0.31	0.48	0.33	0.38	**0.64**	**0.51**	**0.53**

Table 4: Results of external validation with DCR Highlighter Suggest compared with the best models for usability (Llama 3.1, Pipeline A, Prompt 3, 100/100 generated models) and precision (Llama 3.1, Pipeline B, Prompt 3, 81/100 generated models).

shows that while LLM cannot replace human involvement in the discovery of DCR graphs from text, their performance suggests a well-supporting role for process modellers. Furthermore, external validation has shown an increased performance on precision and recall of our approach compared to existing realisations of NLP-based extraction for DCR specifications.

Future work includes the extension of the empirical research by extending the dataset with more annotated process descriptions and evaluating the performance of other, likely larger LLM. Other prompting strategies and performance metrics need to be evaluated as well as the application to other regulatory texts, e.g., clinical guidelines, legal texts and contracts. Additionally, its supportive nature in DCR graph modelling tools will be explored. A revision of the annotation guideline for process descriptions has begun and will be published.

Acknowledgement. This work has received funding from the research grant "Center for Digital CompliancE (DICE)" (VIL57420) from VILLUM FONDEN, and by the Innovation Foundation project "Explainable Hybrid-AI for Computational Law and Accurate Legal Chatbots" 4355-00018B XHAILe.

References

1. M. Dumas, L. M. Rosa, J. Mendling, and A. H. Reijers, *Fundamentals of business process management*. Springer, 2018.
2. H. A. López, S. Debois, T. Slaats, and T. T. Hildebrandt, "Business process compliance using reference models of law," *Fundamental Approaches to Software Engineering LNCS 12076*, p. 378.
3. F. Friedrich, J. Mendling, and F. Puhlmann, "Process model generation from natural language text," in *CAiSE*, pp. 482–496, Springer, 2011.
4. H. Kourani, A. Berti, D. Schuster, and W. M. van der Aalst, "Process modeling with large language models," in *BPMDS*, pp. 229–244, Springer, 2024.
5. A. Nour Eldin, N. Assy, O. Anesini, B. Dalmas, and W. Gaaloul, "Nala2bpmn: Automating bpmn model generation with large language models," in *CoopIS*, pp. 398–404, Springer, 2024.
6. M. Grohs, L. Abb, N. Elsayed, and J.-R. Rehse, "Large language models can accomplish business process management tasks," in *BPM Workshops*, pp. 453–465, Springer, 2023.

7. T. T. Hildebrandt and R. R. Mukkamala, "Declarative event-based workflow as distributed dynamic condition response graphs," *arXiv preprint arXiv:1110.4161*, 2011.
8. T. T. Hildebrandt, A. A. Andaloussi, L. R. Christensen, S. Debois, N. P. Healy, H. A. López, M. Marquard, N. L. Møller, A. C. Petersen, T. Slaats, *et al.*, "Eco-know: Engineering effective, co-created and compliant adaptive case management systems for knowledge workers," in *ICSSP*, pp. 155–164, 2020.
9. M. Barrientos, K. Winter, and S. Rinderle-Ma, "Automatic extraction and formalization of temporal requirements from text: A survey," in *EDOC*, vol. 15409 of *LNCS*, pp. 259–278, Springer, 2024.
10. H. van der Aa, C. Di Ciccio, H. Leopold, and H. A. Reijers, "Extracting declarative process models from natural language," in *CAiSE*, pp. 365–382, Springer, 2019.
11. H. A. López, M. Marquard, L. Muttenthaler, and R. Strømsted, "Assisted declarative process creation from natural language descriptions," in *EDOCW*, pp. 96–99, IEEE, 2019.
12. D. Grathwol, H. van der Aa, and H. A. López, "Automating pathway extraction from clinical guidelines: A conceptual model, datasets and initial experiments," in *CoopIS*, vol. 15506 of *LNCS*, pp. 296–312, Springer, 2024.
13. H. A. López, B. Feng, J. Lindner, M. Franceschetti, and A. Abbad-Andaloussi, "Ambiguity detection in business process descriptions: An evidence and an automated approach," in *BPM*, Springer, 2025.
14. P. Bellan, M. Dragoni, and C. Ghidini, "Extracting business process entities and relations from text using pre-trained language models and in-context learning," in *EDOC*, pp. 182–199, Springer, 2022.
15. J. Neuberger, L. Ackermann, H. van der Aa, and S. Jablonski, "A universal prompting strategy for extracting process model information from natural language text using large language models," in *ER*, pp. 38–55, Springer, 2024.
16. H. A. López, R. Strømsted, J.-M. Niyodusenga, and M. Marquard, "Declarative process discovery: Linking process and textual views," in *CAiSE*, pp. 109–117, Springer, 2021.
17. D. M. T. Trinh, A. Abbad-Andaloussi, and H. A. López, "On the semantic transparency of declarative process models: the case of constraints," in *International Conference on Cooperative Information Systems*, pp. 217–236, Springer, 2023.
18. T. B. Brown, B. Mann, N. Ryder, M. Subbiah, J. Kaplan, P. Dhariwal, A. Neelakantan, P. Shyam, G. Sastry, A. Askell, *et al.*, "Language models are few-shot learners. 2020. doi: 10.48550," *arxiv*, vol. 1, 2005.
19. P. Lewis, E. Perez, A. Piktus, F. Petroni, V. Karpukhin, N. Goyal, H. Küttler, M. Lewis, W.-t. Yih, T. Rocktäschel, S. Riedel, and D. Kiela, "Retrieval-augmented generation for knowledge-intensive nlp tasks," in *Advances in Neural Information Processing Systems*, vol. 33, pp. 9459–9474, Curran Associates, Inc., 2020.
20. L. K. Tamo, A. Abbad-Andaloussi, D. M. T. Trinh, and H. A. López, "An open-source modeling editor for declarative process models," in *CoopIS*, vol. 3552, pp. 1–5, CEUR-WS, 2023.
21. A. Grattafiori, A. Dubey, A. Jauhri, A. Pandey, A. Kadian, A. Al-Dahle, A. Letman, A. Mathur, A. Schelten, A. Vaughan, *et al.*, "The llama 3 herd of models," *arXiv preprint arXiv:2407.21783*, 2024.
22. N. Reimers and I. Gurevych, "Sentence-bert: Sentence embeddings using siamese bert-networks," in *EMNLP*, ACL, 11 2019.
23. H. A. López, T. Hildebrandt, S. Debois, and M. Marquard, "The process highlighter: From texts to declarative processes and back," in *CEUR workshop proceedings*, pp. 66–70, CEUR Workshop Proceedings, 2018.