POLYTECHNIQUE MONTRÉAL
affiliée à l'Université de Montréal

Process Modeling with Large Language Models: Towards a Fine-Grained Evaluation Framework

ALEXIS BRISSARD
Département de génie informatique et génie logiciel

Mémoire présenté en vue de l'obtention du diplôme de *Maîtrise ès sciences appliquées*
Génie informatique

Avril 2026

POLYTECHNIQUE MONTRÉAL
affiliée à l'Université de Montréal

Ce mémoire intitulé :

Process Modeling with Large Language Models: Towards a Fine-Grained Evaluation Framework

présenté par **Alexis BRISSARD**
en vue de l'obtention du diplôme de *Maîtrise ès sciences appliquées*
a été dûment accepté par le jury d'examen constitué de :

Nora BOULAHIA CUPPENS, présidente
Amal ZOUAQ, membre et directrice de recherche
Frédéric CUPPENS, membre et codirecteur de recherche
Omar ABDUL WAHAB, membre

ACKNOWLEDGEMENTS

Thank you to my supervisor Amal Zouaq and my co-supervisor Frédéric Cuppens for their continued involvement in my research and their always relevant feedback.

I also thank the Digital Research Alliance of Canada for providing the computing resources needed to achieve my experiments.

I am especially grateful to my lab mates Edouard, Gaya, Karou, Aditya, Paran, Zacharie, Fabrice, Clara, Arnaud, Gaetan and Neshat, it was a real pleasure to work and spend good times together.

Finally, I thank my family and friends for supporting me through this journey.

RÉSUMÉ

Ce mémoire de maîtrise se concentre sur la Modélisation de Processus qui consiste à construire des modèles de processus à partir de descriptions de processus en langage naturel. Deux approches principales ont été développées pour automatiser cette tâche complexe et chronophage en exploitant les Grands Modèles de Langage : l'Extraction d'Informations de Processus (PIE) et la Génération de Modèles de Processus (PMG). Cependant, ces deux approches ont des limitations et le domaine manque d'un cadre d'évaluation standardisé et complet pour évaluer les capacités réelles et comparer ces méthodes. Ce mémoire présente deux contributions principales : une comparaison détaillée des Représentations de Modèles de Processus, et une nouvelle méthode intégrant PIE et PMG pour fournir une évaluation fine des modèles de processus générés.

ABSTRACT

This master thesis focuses on Process Modeling which consists in constructing process models from process descriptions in natural language. Two main approaches have been developed to automate this complex and time-intensive task leveraging Large Language Models: Process Information Extraction (PIE) and Process Model Generation (PMG). However, they both have limitations and the field lacks a standardized and comprehensive evaluation framework to assess the real capabilities and compare these methods. This report present 2 main contributions: a detailed comparison of Process Model Representations for Process Modeling, and a new method integrating PIE and PMG to provide a fine-grained evaluation of the generated process models.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF SYMBOLS AND ACRONYMS

API	*Application Programming Interface*
BPM	*Business Process Management*
BPMN	*Business Process Model and Notation*
CoT	*Chain of Thought*
CR	*Coreference Resolution*
ICL	*In-Context Learning*
LLM	*Large Language Model*
MD	*Mention Detection*
NER	*Named Entity Recognition*
NLP	*Natural Language Processing*
PI	*Process Information*
PIE	*Process Information Extraction*
PM	*Process Mining*
PMG	*Process Model Generation*
PMR	*Process Model Representation*
RE	*Relation Extraction*
SOTA	*State-Of-The-Art*

LIST OF APPENDICES

CHAPTER 1 INTRODUCTION

Process Modeling is a fundamental activity in business process management that transforms knowledge from various sources into structured, formal process models. Among these sources, natural language textual descriptions represent the most common and accessible form of process documentation, capturing organizational procedures, workflows, and operational guidelines in human-readable format. The goal of Process Modeling is to convert these informal descriptions into precise, executable representations, typically expressed using standardized notations such as Business Process Model and Notation (BPMN) and visualized as diagrams. Figure 1.1 illustrates a representative example of a textual process description alongside its corresponding model, demonstrating the transformation from unstructured text to structured representation.

This research is situated within a broader project on insider threat detection. Identifying malicious or negligent insider behavior requires organizations to first establish a clear understanding of their normal operational processes. Formal process models enable the application of conformance checking techniques [3], which compare observed behavior against expected workflows to detect deviations such as unauthorized data access or procedural violations. However, most organizational process knowledge exists only in textual form. Automated Process Modeling thus serves as a foundational capability for insider threat detection: by transforming textual process descriptions into formal models at scale, it enables systematic monitoring and anomaly detection across an organization's operations.

"A customer brings in a defective computer and the CRS checks the defect and hands out a repair cost calculation back. If the customer decides that the costs are acceptable, the process continues, otherwise she takes her computer home unrepaired."

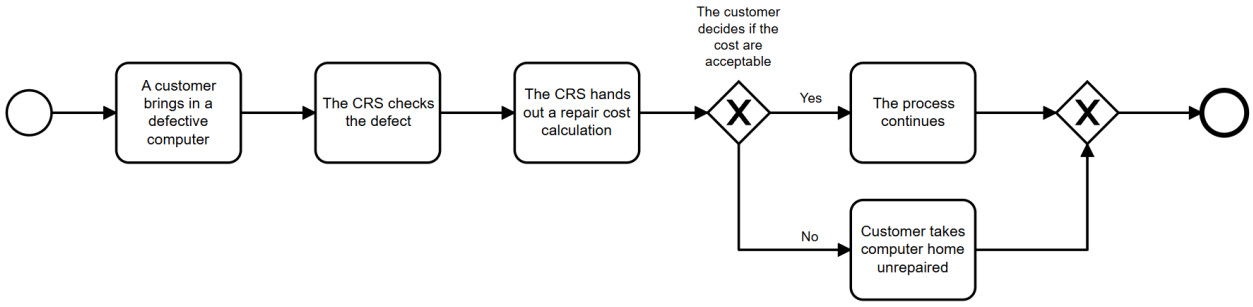


Figure 1.1 Running example of a Computer Repair Service process description and its corresponding process model. Top: the textual description of the process. Bottom: the corresponding process model.[2]

1.1 Problem Elements

Traditionally, Process Modeling has been performed manually by domain experts and business analysts who possess both process modeling expertise and understanding of the organizational context. This manual approach, while capable of producing high-quality models, presents significant practical challenges. The task is labor-intensive and time-consuming, requiring iterative refinement through stakeholder consultations and validation cycles. Moreover, it demands specialized skills that combine formal modeling proficiency with the ability to elicit, interpret, and structure domain knowledge from textual sources. As organizations increasingly seek to document and optimize their processes at scale, the limitations of manual approaches have motivated growing interest in automated or semi-automated Process Modeling techniques.

A core challenge in Process Modeling lies in the inherent ambiguity of natural language descriptions: the same text can legitimately yield multiple, structurally different yet semantically acceptable process models. Modeling decisions regarding granularity, scope, and representation vary with the modeler's expertise, the intended audience, and the specific purpose of the model. For instance, one modeler might emphasize actor responsibilities and organizational roles, while another might prioritize data flows and decision points. Similarly, a single activity mentioned in text might be represented as one coarse-grained task or decomposed into multiple fine-grained steps. Figure 1.2 illustrates this variability by presenting different process models generated for the same textual description from Figure 1.1, each capturing distinct aspects and levels of detail. Unlike many NLP tasks where a single correct answer exists, this multiplicity of valid interpretations substantially complicates rigorous evaluation and reliable automation of Process Modeling .

Recent advances in NLP, particularly the emergence of LLMs, open new perspectives for automating this complex task. These models have demonstrated promising capabilities across a wide range of text understanding and generation tasks. Their application to Process Modeling leverages both their linguistic comprehension and their ability to produce structured outputs.

Two main approaches have emerged to exploit LLM capabilities in this domain. Process Information Extraction (PIE) decomposes the modeling task into structured sub-problems of mention, entity, and relation extraction, drawing inspiration from classical information extraction techniques in NLP. This] approach identifies and extracts specific process-relevant information from text, such as activities, actors, data objects, and their relationships, which can subsequently be used to construct process models. Conversely, Process Model Generation

[2]Originally published in the BPMAI initiative [4] and part of the PET dataset [5].

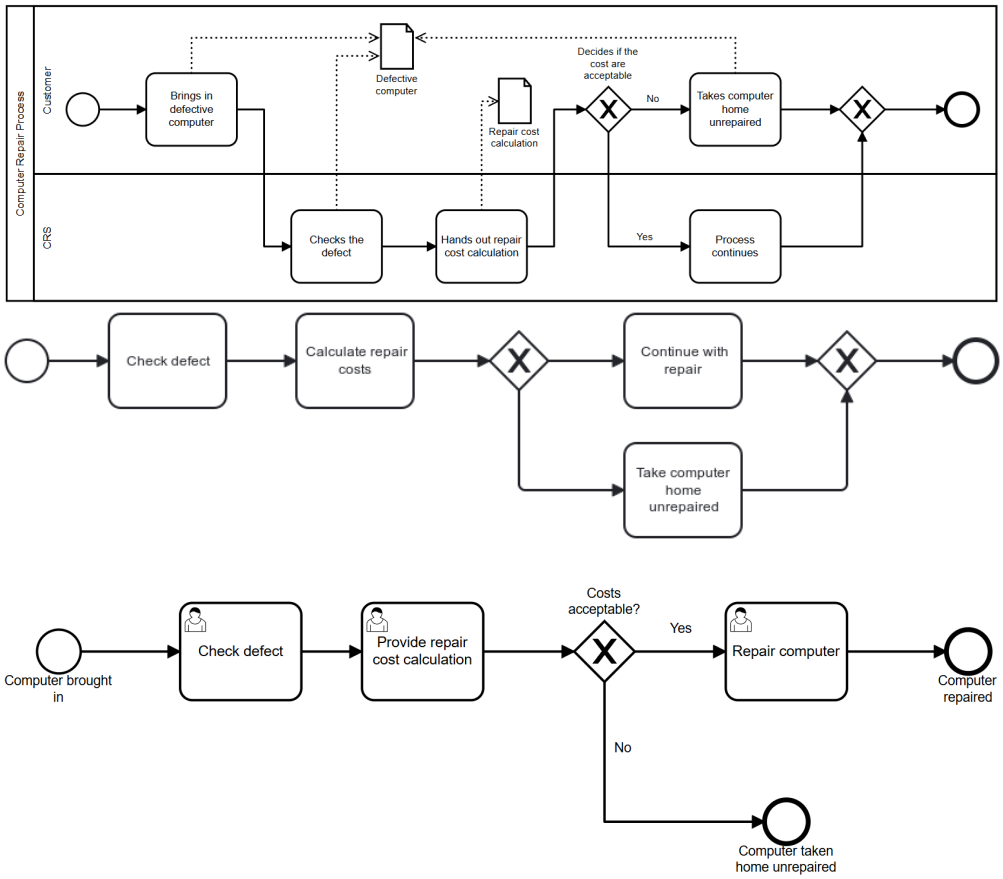


Figure 1.2 Additional process models for the same Computer Repair Service process description from Figure 1.1. The top one is modeled by an expert while the two other are generated by different LLM-based PMG approaches: PromoAI [1] (middle) and BPMN Assistant [2] (bottom).

(PMG) aims to directly generate complete process models from textual descriptions in an end-to-end manner. Most PMG approaches leverage custom-made or existing Process Model Representations (PMRs)—textual abstractions of process models such as BPMN Text [6], Mermaid diagrams [7], or specialized JSON formats [8, 9]—to represent the output process model in a form amenable to LLM generation.

However, three structural limitations persist. First, the field lacks a standardized, comprehensive evaluation framework: assessments are predominantly qualitative case analyses with heterogeneous metrics and ad hoc criteria, preventing objective measurement and robust cross-study comparison [10, 11].

Second, the current landscape is fragmented and opaque. Most approaches depend on closed proprietary models and diverge substantially in PMRs, toolchains, prompting strategies

and budgets, evaluation scripts, and the set of supported process model elements. This heterogeneity hinders transparency and fair comparability [1, 6, 7, 9, 12].

Finally, the field suffers from a shortage of quality annotated data. This limitation hinders the development and evaluation of automated approaches, particularly for techniques requiring supervised training [13].

1.2 Research Objectives

Addressing these limitations requires a systematic investigation of both the representational choices underlying PMG approaches and the methodologies used to evaluate them. The lack of standardized evaluation frameworks and the heterogeneity in PMRs, toolchains, and experimental settings make it difficult to assess which design decisions contribute most to effective Process Modeling with LLMs. Moreover, the shortage of quality annotated data necessitates evaluation methods that can leverage existing resources while providing fine-grained, reliable assessments.

This thesis addresses these challenges through a dual focus: first, by conducting a comprehensive comparison of PMRs to identify which characteristics best support LLM-based Process Modeling ; second, by developing and validating a novel evaluation framework that integrates process information to enable rigorous and fine-grained assessment of PMG outputs. This combined approach reduces the opacity and fragmentation in current research by establishing reproducible evaluation protocols and providing empirical evidence for representation design choices.

In light of these considerations, this thesis pursues three complementary objectives.

1. Establish a standardized comparison setting for existing PMG approaches, reducing evaluative subjectivity and facilitating reproducibility.
2. Identify and empirically compare alternative PMRs to determine which one best support Process Modeling with LLMs and PMG.
3. Propose and validate a hybrid evaluation method that leverages PIE-style extracted process information to produce a model-agnostic, fine-grained assessment of PMG outputs grounded directly in the original textual description.

These objectives give rise to the following research questions:

RQ1 What distinguishing features of existing PMRs most influence their effectiveness for Process Modeling and PMG, and which PMR offers the best trade-offs in practice?

RQ2 Can integrating process information into the evaluation pipeline yield a reliable, fine-grained, and model-agnostic assessment of PMG outputs?

1.3 Thesis Outline

Chapter 2 (Literature Review) synthesizes prior work on Process Modeling with LLMs, presents related Information Extraction tasks, introduces representative PMRs, and highlights open gaps in evaluation, representation choice, and transparency.

Chapter 3 (Process Model Representation Comparison) defines comparative criteria for PMRs, introduces the dataset and semantic evaluation procedure, and reports the empirical comparison leading to the selection of candidate PMRs for Process Modeling tasks including PMG.

Chapter 4 (Process Modeling Evaluation) presents the hybrid evaluation method integrating process information with PMG outputs, defines fine-grained metrics, and evaluates multiple PMG approaches under a standardized protocol.

Chapter 5 (Conclusion) recapitulates the main contributions, discusses the limitations of this work, and outlines prospective extensions.

CHAPTER 2 LITERATURE REVIEW

In this chapter, we present the key components and recent advancements related to Process Modeling . We first introduce the Business Process Management (BPM) domain and BPMN, which serves as the target representation for most Process Modeling approaches. We then review the application of LLMs in the BPM domain before focusing on Process Modeling itself. The Process Modeling section covers the two main approaches: PIE and PMG, along with their respective datasets. We also present process model evaluation methodologies, highlighting common elements and gaps in the literature. Finally, we expand our review to the broader IE field, examining recent advances in Named Entity Recognition, Coreference Resolution, and Relation Extraction that inform our experimental design.

2.1 Business Process Management

BPM is a discipline that combines management and information technology to improve organizational processes through their analysis, design, implementation, and continuous optimization [14]. At its core, BPM relies on process models that capture the sequence of activities, decisions, and flows within a business process. These models facilitate communication among stakeholders, support process analysis and improvement, enable process automation, and provide documentation for compliance and training.

2.1.1 BPMN

BPMN is the de facto standard notation used to represent process models. Originally developed by the Business Process Management Initiative and later maintained by the Object Management Group, BPMN provides a standardized graphical language that is both human-readable and machine-executable. Its widespread adoption across industry tools and academic research makes it the target representation for most Process Modeling approaches.

Figure 2.1 illustrates a typical BPMN process model. The model depicts a sequential flow starting with a start event, followed by three tasks executed in order. An exclusive gateway then splits the process based on a condition, leading to two alternative branches, each containing a specific task. These branches are later merged by a closing exclusive gateway before reaching the end event. All these elements are connected by sequence flows that define the execution order.

The process is organized using pools and lanes (collectively referred to as swimlanes). The

pool represents the overall organizational boundary labeled *Computer Repair Process*, while lanes within the pool partition activities by actor or organizational role. In this example, two lanes are visible: *Customer* and *CRS* (Customer Repair Service). The *Customer* lane contains activities performed by the customer (e.g., Brings in defective computer, Takes computer home unrepaired), while the *CRS* lane contains activities performed by the repair service (e.g., Checks the defect, Hands out repair cost calculation, Process continues). This visual separation clarifies responsibility assignment and facilitates understanding of cross-functional interactions.

Data objects are also present in the model, representing information artifacts that flow through the process. Two data objects are shown: *Defective computer* and *Repair cost calculation*. These are connected to activities via dotted lines (data associations), indicating which activities use or produce specific information items. For instance, *Repair cost calculation* is produced by the corresponding CRS activity.

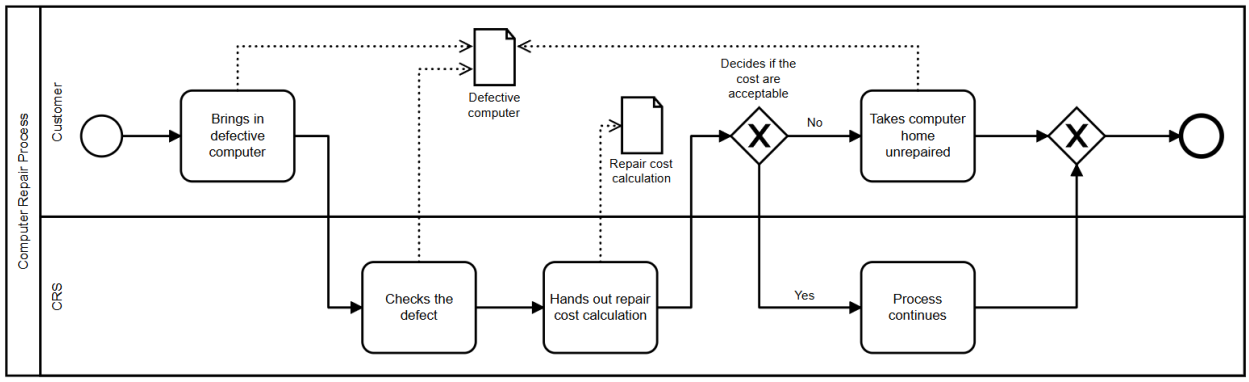


Figure 2.1 Example BPMN process model containing events (start and end), tasks (rounded rectangles), exclusive gateways (diamonds marked with X), sequence flows (solid arrows), data objects (documents), data associations (dotted lines), and swimlanes organizing activities by role (Customer and CRS lanes within the Computer Repair Process pool).

Following [15], we formally define a BPMN process model as a business process graph. Let T be a set of element types and Ω be a set of text labels. A business process graph is a 5-tuple $(N, E, \tau, \lambda, \alpha)$ where N is a finite set of nodes, $E \subseteq N \times N$ is a finite set of edges, $\tau : (N \cup E) \to T$ associates nodes and edges with types, $\lambda : (N \cup E) \to \Omega$ associates nodes and edges with labels, and $\alpha : (N \cup E) \to (T \to \Omega)$ associates nodes and edges with attributes. The set of types T encompasses activities, events, gateways and data objects while Ω includes sequence flows, message flows, and data associations. Attributes define properties such as pool and lane assignments. This formal foundation provides a precise framework for comparing and evaluating automatically generated process models.

2.1.2 LLMs for Business Process Management

The integration of LLMs into BPM has generated significant interest, with researchers highlighting their potential across various BPM tasks in position papers [16–20]. LLMs have demonstrated capabilities in knowledge extraction and translation while being versatile [21]. We organize the main research directions into four categories: process prediction, semantic process understanding, benchmarking and evaluation, and process model comprehension.

Predictive Process Management

Several approaches leverage LLMs for predicting tasks within process models. Inspired by text generation, [22] and [23] employ LLMs to predict the next element in a process model, with [22] finding that GPT-3 recommendations are indistinguishable from human-created elements. [24] assess LLM capabilities on detecting anomalies in traces and predicting the next activity, finding that fine-tuned LLMs outperform pretrained language models like BERT, but that prompting alone is insufficient (~0.8 vs 0.5 F1-score). Beyond prediction, [25] focus on explainability by using attention scores to interpret transformer-based next activity predictions.

Process Comprehension

Understanding and explaining process models represents another research direction. [26] develops a QA dataset focused on causality within processes, contributing to broader process reasoning capabilities. [27] present AIPA, a tool enabling LLMs to understand and explain process models by comparing different BPMN abstractions and prompting techniques. [28] introduce SAX4BPM for situation-aware explainability, finding that structured input improves explanation fidelity but at the cost of interpretability.

Benchmarking and Evaluation

As LLMs become more prevalent in BPM applications, standardized evaluation frameworks have emerged. [29] develop a comprehensive benchmark for evaluating LLMs in the BPM domain across multiple diverse tasks, finding that no single model performs best across all tasks. [**?**] present WONDERBREAD, the first benchmark for evaluating multimodal foundation models on BPM tasks beyond automation, covering workflow documentation, knowledge transfer, and process improvement. Their results show that while models can generate documentation effectively, they struggle with finer-grained workflow validation (F1 < 0.3).

Additionally, [19] provide a more in-depth analysis of the subject of prompt engineering for BPM. For a broader literature review on the intersection of BPM and natural language processing, we recommend referring to the work of [13], which also contains a section on the use of LLMs.

Among the diverse applications of LLMs in BPM, the Process Modeling task has received particular attention due to its potential to reduce the manual effort required in process documentation and analysis. We now turn our attention to this specific task.

2.2 Process Modeling

Process Modeling traditionally focuses on designing how a process should work, often involving expert modelers and multiple sources of information [14]. Beyond initial model creation, Process Modeling encompasses related tasks such as process model modification, question answering about existing models, and conformance checking between models and descriptions.

In this study, we focus on the task of automatically extracting a process model from a single document, such as a process description, without relying on external expertise or additional data. This specific task has also been referred to as Text-to-Process (T2P) [30] or automated process modeling [31].

The application of LLMs to Process Modeling represents a recent but rapidly growing area of research, with early studies demonstrating promising results [7, 10, 32]. While [7] specifically frame their approach as *Conversational Process Modeling* , we use the more general terms *Process Modeling with LLMs* or *LLM-based Process Modeling* to encompass the broader range of methodologies.

The first insights into the extraction of process models from textual descriptions can be traced back to the seminal work of [33], along with previous studies such as that by [34]. Building on this foundational knowledge, [35] presents a literature review that introduces the concept of Process Extraction from Text, which closely relates to our framing of Process Modeling . Additionally, [36] aims to compare the effectiveness of LLMs in the context of Process Modeling . However, this study primarily provides general guidelines and does not offer a direct evaluation of LLMs on specific Process Modeling tasks. Finally, [37] present a systematic literature review of automated Process Modeling up to 2022.

Process Modeling can be split into two main approaches: Process Information Extraction (PIE) and Process Model Generation (PMG). We present the fundamental characteristics, advantages, limitations and relevant papers of each approach in the following sections.

2.2.1 Process Information Extraction

As outlined by [13], the PET dataset [38] is the main dataset used to evaluate PIE and was the first to formalize the task; hence, we base our explanations on the PET formalization of PIE. See section B for statistics about its content.

PIE consists of extracting three different types of Process Information (PI) from a description:

Mentions. Mentions are spans of text associated with a type. The original mention types are: Activity, Activity Data, Actor, Further Specification, XOR Gateway, AND Gateway, Condition Specification [38]. In PIE, the task of extracting mentions is called Mention Detection (MD).

Entities. Entities are groups of mentions corresponding to the same thing (i.e., [Activity Data]) or person (i.e., [Actor]). By definition, all mentions in an entity have the same type, which can be either Actor or Activity Data. The task of grouping mentions into entities is called Coreference Resolution (CR). Entities were initially not present in the PET dataset [38] and were introduced in [39].

Relations. Relations are associations between two mentions. Relations can have the following types: Sequence Flow (between two Activity), Uses (between Activity and Activity Data), Actor Performer (between Activity and Actor), Actor Recipient (between Activity and Actor), Further Specification (between Activity and Further Specification), Same Gateway (between two XOR Gateway or AND Gateway). The associated task is Relation Extraction (RE). The choice of linking mentions instead of entities can seem counterintuitive, especially since mentions will be merged as entities in the final business process. This can be explained by two reasons. First, entities were initially not considered in PIE and the same annotations were kept in subsequent versions of the PET dataset. Second, it is simpler to extract relations at the mention level instead of considering entities that can span several sentences. All relations involving entities (i.e., Uses , Actor Performer and Actor Recipient) are defined within a sentence and do not require additional context to be extracted. This claim is justified by the very high results obtained for RE by [40] (see Table 2.1).

Once all the PIE tasks are performed, a process model can be built from the PI. There is no standardized way of assembling the information into an understandable and complete process model. To our knowledge, the only method available in the literature is presented in [40] with code implementation provided alongside [41][1]. This method uses the first mention of each [Actor] as the lane name in the generated BPMN model. Activity Data mentions are handled in two ways: they are added as separate data objects in the model and concatenated to the

[1] https://github.com/JulianNeuberger/pet-to-bpmn-poc/

corresponding task label. Custom logic determines the assignment of tasks to lanes and the association of data objects based on Actor Performer and Uses relations.

Formal definitions

Let D be a process description consisting of a sequence of tokens. We formally define the three types of PI as follows. A mention m is a tuple (s, e, t) where s and e denote the start and end token positions in D (with $s \leq e$), and $t \in T$ is the mention type from a predefined set T of mention types. Let $M = \{m_1, m_2, \ldots, m_n\}$ denote the set of all mentions extracted from D.

An entity ε is a non-empty subset of mentions $\varepsilon \subseteq M$ such that all mentions in ε refer to the same real-world object or person. Formally, an entity satisfies two constraints: (1) type consistency: $\forall m_i, m_j \in \varepsilon, t_i = t_j$ where t_i and t_j are the types of mentions m_i and m_j respectively, and (2) coreference: all mentions in ε refer to the same referent. Let $E = \{\varepsilon_1, \varepsilon_2, \ldots, \varepsilon_k\}$ denote the set of all entities, forming a partition of $M_E \subseteq M$ where M_E contains only mentions of types Actor and Activity Data.

A relation r is a tuple (m_i, m_j, ρ) where $m_i, m_j \in M$ are source and target mentions respectively, and ρ is the relation type. Each relation type imposes type constraints on its source and target mentions, formally defined as $\rho : T_{\text{source}} \times T_{\text{target}}$ where $T_{\text{source}}, T_{\text{target}} \subseteq T$. Let $R = \{r_1, r_2, \ldots, r_p\}$ denote the set of all relations extracted from D.

The complete PI P extracted from description D is defined as the tuple:

$$P = (M, E, R) \tag{2.1}$$

where M is the set of mentions, E is the set of entities, and R is the set of relations. This tuple captures all structured information necessary to construct a process model from the textual description.

PIE approaches

Several research efforts have explored PIE using LLMs, with two main lines of work emerging in recent years.

A first line of work in PIE is explored in [42–44]. It is aimed at building knowledge graphs using GPT-3 on PET dataset descriptions. To achieve their goal, they perform a subset of the MD and RE tasks in 3 steps: (1) extracting Activity mentions, (2) extracting mentions of *process participants* (Actors) and their associated *performs* (Actor Performer) relations

for each activity, and (3) extracting follows ($\xrightarrow{\text{Sequence Flow}}$) relations between activities. However, they regard semantically accurate annotations as valid based on their own expertise, which renders comparison with studies employing strict matching infeasible.

A second line of work is explored in [39, 40]. The first paper introduces entities and the Coreference Resolution task to complete the PET dataset and enable process model generation from the PET annotations. It also evaluates JEREX [45] a Joint Relation Extraction model (see section 2.4.3 for details) and its own specialized trained models against the PET baselines (see Table 2.1). JEREX achieves a low performance of 0.31 F1-score on the overall pipeline due to error propagation between tasks. In their subsequent work, [40] propose an ICL-based approach which prompts GPT-4 on the three PIE tasks, attaining state-of-the-art results for each. Although results on the combined task are not reported, the marginal improvements over previous baselines suggest that the overall performance remains insufficient for end-to-end PIE.

Table 2.1 presents the complete results of these 2 works on the PET dataset. The reported metric is strict F1-score meaning only perfect spans are counted as correct mentions.

Finally, acknowledging the lack of data in the PIE field, [41] highlights the need for an annotation tool to facilitate the expansion of the PET dataset and presents first work in this direction. The tool is formally introduced in [46] as TeaPie. It uses the task-specific models introduced in [39] to generate recommendations of mentions, entities and relations. These recommendations are combined with the ongoing user annotations to present a visualization of the current process model.

[47] can also be considered as a PIE approach. It consists of extracting Task, Agent, Task Information, PI and Condition mentions using a BERT model fine-tuned on 100 manually annotated examples. Exclusive and parallel gateways are then extracted using GPT models. They evaluate their approach on 31 process description-model pairs using the relative graph edit distance. Notably, this is the only PIE approach using a fine-tuned LLM.

2.2.2 Process Model Generation

The second main Process Modeling approach is Process Model Generation (PMG). While PIE splits the task into three sub-tasks, PMG consists of directly generating the whole process model using the description. To support this generation, researchers have introduced a variety of PMRs—textual abstractions used to represent the output process model.

The initial motivation for using PMRs stemmed from the limitations of LLMs, which were unable to directly generate standard BPMN models due to constraints in context length and

Task *Extracted PI*	**MD** Mentions	**CR** Entities	**RE** Relations	**All** All
Trained baselines [39]	Conditional Random Field 0.69	Neural coreference resolver 0.52	Decision tree ensemble 0.72	JEREX 0.31
GPT-4o prompting [40]	**0.74**	**0.74**	**0.89**	-

Table 2.1 PIE approaches results on the PET dataset. Metric is strict F1-score. *All* column presents results when chaining the three tasks.

suboptimal LLM performance [7, 48]. To address these challenges, alternative representations such as simplified BPMN [27] or existing diagram notations [7] were introduced. Since then, more advanced PMG approaches have emerged, proposing new techniques such as constrained generation [9, 12] or multi-agent frameworks [6], often taking advantage of their own custom PMRs.

We present the main PMRs and PMG approaches in the rest of this section, as well as additional works related to PMG.

Main PMG approaches

Conversational approaches engage users in an iterative dialogue with an LLM to create or refine process models. [49] and [50] pioneered this direction. [49] explores how closed-source LLMs such as GPT-3.5 can be used for Process Modeling and in particular extracting the tasks from process descriptions using the Mangler dataset [51] as an evaluation set. They compare different prompts and name their approach ConverMod. [52] presents the results of a user study comparing process models generated by LLMs with expert-generated models, finding that 60% of the users prefer LLM-generated models regardless of their experience in process mining. [7] regroups both previous papers and provides additional insights (e.g., evaluating GPT-4) with the same findings. Additionally, they evaluate ConverMod on a subset of 7 descriptions from the PET dataset for which they create business models (hereafter referred to as PET-7). Their main evaluation metric is the Jaccard index between the generated and gold sequence flows. They also investigate the use of Graphviz and Mermaid.js as intermediary representations for BPMN, finding that Graphviz leads to fewer formatting errors while Mermaid.js yields better results in general.

Code-based approaches generate executable code that produces process models when

run. [1, 53] introduce ProMoAI, where Python code is generated and executed to obtain process models in POWL, a modeling language that can be translated to BPMN. Designated as *POWL code*, this PMR requires extended prompts and multiple error correction rounds, which limits its practicality. [54] also adopts a code-based strategy, creating a dataset of process descriptions matched with process trees to fine-tune a small-scale LLM (Llama-3-8B). They achieve 0.56 accuracy when evaluating their method on one description from the PET dataset, highlighting the need to create higher quality data and improve the task definition.

Constrained generation approaches enforce structural validity through schema constraints during generation. [9] proposes BPMN-chatbot, an enhancement of the ProMoAI framework that introduces a more structured and practical PMR. Labeled in this study as *JSON branches*, this PMR encodes the process branching structure in JSON format and leverages OpenAI's function calling to enforce schema adherence during generation. Using PET-7 as in ConverMod, the authors conduct a qualitative evaluation and report improved performance over both previous approaches. Similarly, [12] develop a two-step constrained generation approach: the first step extracts relevant information from the description and the second creates a JSON representation of a Petri net with schema constraints. However, their approach has two main limitations: the model must handle the layout of the diagram, adding unnecessary complexity, and they qualitatively evaluate their approach on only 3 custom examples without comparing it with other methods.

Multi-agent and multimodal approaches leverage more advanced LLM capabilities. [6] uses a multi-agent framework combined with a custom PMR (BPMN text) to improve the generation process. They evaluate their results on the BPMN for research dataset [55] using BPMNDiffViz [56], a tool to calculate the structural similarity between two processes. According to these metrics, their method outperforms both human modeling and ProMoAI. [8] evaluates multimodal LLM capabilities by generating process models from process description and diagram pairs, introducing a representation based on lists of Process Model Elements (PME) for both generation and evaluation.

Table 2.2 summarizes PMG approaches presented in this section. Despite recent progress, existing studies face several limitations: reliance on closed-source LLMs limits reproducibility, evaluation sets remain small, and performance is often assessed using non-standardized metrics. While some studies have begun to compare LLMs [10], the field still lacks a systematic and reproducible evaluation of PMG pipelines.

Table 2.2 Main PMG approaches from the literature.

PMG approach	PMR	LLM	Evaluation set	Evaluation method
ConverMod [7]	Graphviz + Mermaid	GPT-4	Mangler + PET-7	Text similarity + Qualitative
PromoAI [1]	POWL code	GPT-4	2 descriptions	Qualitative
Forell [12]	JSON-Nets	GPT-4	3 descriptions	Qualitative
MAO [6]	BPMN text	GPT-4	BPMN for Research	BPMNDiff Viz
Multimodal [8]	PME	GPT-4V	SAP-SAM	PME similarity
BPMN-chatbot [9]	JSON branches	GPT-4o	PET-7	Qualitative

Evaluating LLMs for PMG

Beyond proposing new PMG approaches, several studies have focused specifically on evaluating LLM capabilities for this task. [10] provide a new benchmark consisting of 20 process descriptions associated with a ground truth process model and event logs generated from the process model. They evaluate LLMs on the PMG task by employing conformance checking to assess if the generated process models cover the event log executions. However, they have to provide the list of activities in the prompt and their evaluation is restricted to large-scale LLMs.

[32] introduce a theoretical evaluation framework, highlighting the need for more quantitative assessment methods but do not perform new evaluations.

[31] explores the LLM tendency to produce knowledge-driven hallucinations, i.e., ignoring provided data and basing responses on parametric memory. They test this behavior on the PMG task, leveraging the ProMoAI framework to generate models based on 4 shuffled or reversed descriptions from the Process Modeling Benchmark. They evaluate the generated models using behavioral-footprint similarity (see Section 2.3.3) from PM4Py [57]. They find that all LLMs are subject to these types of hallucinations, even when provided with a stricter prompt, highlighting the need for rigorous validation methods.

The evaluation methods mentioned in this section, along with additional approaches from the broader PMG literature, are examined in detail in Section 2.3. We now turn to the datasets that enable this evaluation.

2.2.3 Datasets

A significant challenge in the Process Modeling field is the scarcity of high-quality datasets, as noted by [13]. Data is critical for leveraging the full capabilities of LLMs, as they require a large number of samples to be properly trained or fine-tuned. In addition, as noted in section 2.4.1, Information Extraction datasets such as CoNLL are not similar enough to be useful directly for PIE. Transfer learning could be an avenue of research but is not straightforward to leverage as the Process Modeling tasks require a complete domain adaptation. To justify our data choices and provide valuable resources for future researchers, we summarize the main datasets available for our task and their limitations in Table 2.3.

Dataset	**Type and size of data**	**Task**	**Limitations**
bpmn-for-research [55]	4 process descriptions each associated with 1 ground truth model and 40+ alternative models	PMG	No evaluation of alternative models
Original PET [5]	45 process descriptions with mention and relation annotations	PIE (MD and RE)	Low quantity of data, missing entity annotations
PET [39]	45 process descriptions with mention, entity and relation annotations	PIE (MD, CR and RE)	Low quantity of data
MaD [58]	30,000 process descriptions with associated Graphviz models	PMG	Low diversity models and descriptions. 1 sentence is always linked to 1 task.
Mangler dataset [51]	24 process descriptions each associated with 8-11 BPMN models	PMG	Some models are not matching descriptions
ProMoAI benchmark [10]	20 process descriptions with associated BPMN model and event logs	PMG	Activity names have to be provided when evaluating LLMs

Table 2.3 Summary of the datasets available for Process Modeling . Datasets used in this study are in bold. PMG: Process Model Generation, PIE: Process Information Extraction, MD: Mention Detection, CR: Coreference Resolution, RE: Relation Extraction.

2.3 Evaluation methodologies

In this section, we focus on methodologies aimed at evaluating process models. As outlined in Section 2.2.2, most studies use expert-based qualitative approaches to grade generated

models from 1 to 5 based on either the original description or a provided ground truth model [1, 9, 12, 59–61]. This approach is problematic since the method is not systematic and the exact evaluation setup is often not provided (e.g., blind comparison between models, direct grading of a single model). Moreover, human judgment can lack objectivity and be biased. Therefore, there is a need for reproducible and automated evaluation methodologies, and some of them were introduced alongside new Process Modeling approaches.

While some methodologies were developed specifically for Process Modeling , many leverage techniques from other BPM contexts such as process repository management [62]. These techniques (graph edit distance, behavioral similarity, element-based matching) compare a generated model against a ground truth. As noted by [63], nearly all such approaches rely on matching elements between models (primarily activities), then computing an aggregated similarity score from these matches. However, a single process description can yield multiple valid models differing in granularity, naming, or structure. Comparing against one chosen ground truth penalizes valid alternatives. To address this, some methodologies evaluate the model directly against the process description.

We present both model-to-model and model-to-description evaluation approaches in the following sections. We first examine the foundational techniques for label matching and set similarity, which underpin both types of methodologies.

2.3.1 Label Matching

Both model-to-model and model-to-description evaluation require matching labels between compared elements. Due to modeling reformulations (by LLMs and human modelers), labels for tasks, activities, conditions, data objects, and other elements may differ even when referring to the same concept. Several approaches have been used to match labels or compute their similarity:

Expert-based matching: A human expert manually matches the labels [49, 64].

String-based matching: Early methodologies often leverage simple string comparison techniques, e.g., SequenceMatcher[2], Levenshtein distance [56].

Semantic-based matching: A more advanced and reliable way to match labels is to leverage semantic approaches. This can be achieved using synonym matching, but most recent approaches compute the cosine similarity between textual embeddings. These embedding can be obtained by an embedding model such as `sts-mpnet-cosine` [65] or `text-embedding-3-small` [15].

The main drawback of matching labels using similarity-based techniques is the need to

[2] `https://docs.python.org/3/library/difflib.html`

manually define a threshold above which labels are considered equal. This threshold is often defined arbitrarily [15, 65], hindering the robustness of the overall method and adding an additional hyperparameter. We perform the first systematic evaluation of both similarity metrics and matching thresholds in Section 4.1.2. We also introduce Element counts a new metric that does not require matching yet provides relevant insights when comparing models.

2.3.2 Set Similarity

Once labels are matched, evaluation methodologies aggregate individual matches into overall similarity scores. When comparing sets of elements (e.g., tasks in a generated model versus tasks in a reference model or extracted from a description), several approaches quantify their similarity.

Similarity matrix with greedy matching

The most flexible approach for comparing two sets of elements is to first construct a similarity matrix, where each entry (i, j) represents the similarity between element i from the first set and element j from the second set. The similarity can be computed using any of the label matching techniques described in Section 2.3.1.

Given two sets of elements $A = \{a_1, a_2, \ldots, a_m\}$ and $B = \{b_1, b_2, \ldots, b_n\}$, the similarity matrix S is defined as:

$$S_{ij} = \text{sim}(a_i, b_j) \quad \text{for } i \in \{1, \ldots, m\}, j \in \{1, \ldots, n\} \tag{2.2}$$

where $\text{sim}(\cdot, \cdot)$ is a similarity function (e.g. cosine similarity of embeddings, Levenshtein similarity).

Once the similarity matrix is constructed, a greedy matching algorithm is typically applied to pair elements. At each iteration, the algorithm selects the pair (i, j) with the highest similarity score S_{ij}, matches these elements, and removes the corresponding row and column from further consideration. This process continues until no more pairs can be matched (either all elements from one set are matched, or remaining similarities fall below a threshold).

The overall similarity between sets A and B can then be computed as:

$$\text{Similarity}(A, B) = \frac{\sum_{(i,j) \in M} S_{ij}}{\max(|A|, |B|)} \tag{2.3}$$

where M is the set of matched pairs. This approach is used in several Process Modeling evaluation methods [62, 65].

Jaccard similarity

The Jaccard similarity coefficient, also known as the Jaccard index, is one of the simplest and most widely used metrics for comparing sets. It measures the ratio of the intersection to the union of two sets:

$$J(A, B) = \frac{|A \cap B|}{|A \cup B|} = \frac{|A \cap B|}{|A| + |B| - |A \cap B|} \tag{2.4}$$

The Jaccard similarity ranges from 0 (no overlap) to 1 (identical sets). For process models, this metric is particularly useful when comparing sets of elements with exact string matching (e.g., when element names are standardized). For instance, it is used to compute similarity between activity pairs in [31, 49].

The main limitation of Jaccard similarity is that it requires exact matches between elements and does not account for partial similarities or semantic equivalence. Therefore, it is often combined with label matching techniques to determine whether two elements should be considered part of the intersection.

Dice-Sørensen coefficient

The Dice-Sørensen coefficient, also known as the Sørensen-Dice index or F1-score of set overlap, is another popular metric for comparing sets. It is defined as:

$$\text{DSC}(A, B) = \frac{2|A \cap B|}{|A| + |B|} \tag{2.5}$$

Compared to Jaccard similarity, the Dice-Sørensen coefficient gives more weight to the intersection by counting it twice in the numerator. This makes it more sensitive to matches and can be preferred in contexts where finding common elements is more important than penalizing differences. The coefficient also ranges from 0 to 1.

The relationship between Jaccard similarity and Dice-Sørensen coefficient can be expressed as:

$$\text{DSC}(A, B) = \frac{2 \cdot J(A, B)}{1 + J(A, B)} \tag{2.6}$$

In Process Modeling evaluation, the choice between Jaccard and Dice-Sørensen often depends on the desired sensitivity to missing elements. Dice-Sørensen tends to produce higher similarity scores and is sometimes preferred when a more lenient evaluation is desired, such as in [65].

Precision, Recall, and F1-score

When one set is considered as the ground truth (reference model) and the other as predictions (generated model), set similarity can be expressed using standard precision, recall, and F1-score:

$$\text{Precision}(A, B) = \frac{|A \cap B|}{|A|} \tag{2.7}$$

$$\text{Recall}(A, B) = \frac{|A \cap B|}{|B|} \tag{2.8}$$

$$\text{F1-score}(A, B) = \frac{2 \cdot \text{Precision} \cdot \text{Recall}}{\text{Precision} + \text{Recall}} = \frac{2|A \cap B|}{|A| + |B|} \tag{2.9}$$

where A represents the predicted (generated) set and B represents the ground truth (reference) set. Note that the F1-score is equivalent to the Dice-Sørensen coefficient.

These metrics provide a more nuanced view of set similarity by distinguishing between over-generation (low precision) and under-generation (low recall) of elements. This decomposition is particularly valuable for diagnosing specific weaknesses in Process Modeling approaches. We leverage these metrics in our evaluation methodology presented in Chapter 4.

2.3.3 Model to Model Evaluation

The first category of methods we analyze is model-to-model evaluation. Comparing two process models has been a longstanding challenge in the BPM community [62], and several approaches have been developed over the years.

Some of them are standardized and integrated into existing libraries (e.g., `pm4py.analysis`), while others have failed to be implemented consistently and suffer from small variations when used in different studies.

Following the seminal work of [62], these methodologies have been classified into three main categories: **element-based metrics** (focusing on the individual elements of the process), **structural metrics** (focusing on what the process is made of—its static components, elements, and connections), and **behavioral metrics** (focusing on what the process does—its dynamic execution, flow of work, and resulting actions or outcomes).

Element-based metrics aim to match elements of the two processes based on their attributes. Elements are grouped by type (e.g., tasks, gateways) and are then matched with reference elements using greedy matching. The principal feature used in this process is the element label, which is matched using techniques from the preceding section. Other features that can

be used to match these elements are contextual information such as incoming or outgoing connections, or attribute information such as element lane. The main advantage of these techniques is their fine-grained character, leading to precise feedback about which specific element or type is missing from the process model. Yet, this fragmented view of the model can fail to grasp the entire picture.

Structural metrics can mitigate this lack of general view by leveraging model-wise information. The most used metric in this category is the Graph Edit Distance (GED), a graph-based metric measuring the optimal number of edits necessary to convert one model into the other. As all edit types are not equal (e.g., adding a flow is less significant than adding a gateway), they are often weighted differently. The distance is also normalized to obtain a final score independent of the length of the evaluated process models. The calculations need to incorporate element matching techniques as a first step to account for similar elements. Other structural metrics include the feature-based similarity that averages the Euclidean distance between a selection of structural features (e.g., number of nodes, maximum depth, average branching, number of splits) and embeddings similarity, which is the cosine similarity between two models encoded using the PetriNet2Vec algorithm [66].

Behavioral metrics can be considered the most comprehensive as they encompass both previous dimensions. For the process behavior to be correctly represented, process model elements must be equivalent between the models, and they must be structured in a similar manner. The first behavioral metric specifically designed for model comparison was introduced in [62] and is known as causal footprint similarity. In this approach, task labels are matched and combined with control flows to define a low-dimensional embedding of the process model, which is subsequently compared with the reference model embedding using cosine similarity. However, this methodology lacks a standardized implementation and has consequently not been widely adopted. In contrast, behavioral footprint similarity is more commonly employed due to its implementation in the `pm4py` library. This metric computes the Jaccard index between sequence and parallel activity combinations of the compared models.

A more sophisticated metric was introduced in [10], which leverages established conformance checking techniques. The methodology begins by generating synthetic event logs from the compared models using standard simulation techniques. All possible execution paths in the model are enumerated, with loops constrained to a single iteration to limit the size of the event log. Subsequently, fitness (recall)—representing the extent to which the model captures the event logs—and precision—representing the specificity of the model to the event log and its exclusion of alternative execution paths—are calculated using token-based or alignment techniques originally developed for conformance checking. While this approach

benefits from leveraging well-established metrics, it requires exact matching of task labels between models. Currently, this is achieved by providing ground truth task labels as input to the PMG approach, which constitutes a significant limitation. Semantic matching methods could potentially address this issue, although to the best of our knowledge, they have not yet been applied in this context.

2.3.4 Model to Description Evaluation

While model-to-model comparison offers utility in various scenarios, the primary objective is to evaluate the conformance of a generated model to its textual description. Consequently, one could argue that constructing an arbitrary ground truth process model as an intermediary comparison point represents an unnecessary step, and that evaluation should directly assess the process model against the original description. While this reasoning is sound, such direct evaluation can be more complex than the generation task itself, which explains the paucity of methodologies in this domain.

An initial approach in this direction involves the use of an LLM-based judge to provide scalar rewards ranging from -1 to 1 to train a smaller model for PMG, as demonstrated in [67]. However, this feedback mechanism suffers from limited granularity and poor interpretability. Furthermore, this approach introduces significant inference costs for evaluation. As a standalone evaluation method, it could be more beneficial to directly use the larger LLM for model generation.

Round-tripping [68] represents another promising research direction, which has been investigated by the SAP Signavio team. This methodology involves transforming the evaluated model back into a textual description, then comparing this generated description with the original using metrics such as sentence embedding similarity. Alternatively, the process can be initiated from a ground truth model, which is used to generate a description that is subsequently transformed back into a process model. While these bidirectional transformation methods facilitate comparison between PMG approaches, they introduce a dependency on process description generation techniques. Consequently, the observed performance cannot be reliably attributed to a single component of the pipeline (model or description generation), complicating the interpretation of results.

We address this gap in model-to-description evaluation methodologies in Chapter 4 by introducing a novel approach based on Process Information.

2.3.5 Discussion

Tables 2.4 and 2.5 present a comprehensive summary of evaluation methods along with their distinguishing characteristics. A critical observation is that only a minority of these methods provide fine-grained feedback—that is, the capability to identify specific element types or localized changes in the evaluated model. This lack of granularity represents a significant limitation, as detailed, component-level feedback is essential for systematically identifying the specific deficiencies and failure modes of each Process Modeling approach. Without such diagnostic capabilities, it becomes challenging to guide targeted improvements in model generation methodologies.

Table 2.4 Summary of evaluation methods for Process Modeling with their characteristics (Part 1). M2T: Model-to-Text, M2M: Model-to-Model, FG: Fine-Grained, Exp: Expert-based, Code: Code-based.

Metric	Ref	Used in	M2T	M2M	FG	Evaluation		
						Exp.	Code	LLM
Expert evaluation	-	[1, 9, 12, 59–61]	✓	✓	✗	✓	✗	✗
Node-matching similarity	[62]		✗	✓	✓	✗	✓	✗
Graph Edit Distance similarity	[62]	[15, 33, 59]	✗	✓	✓	✗	✓	✗
Causal footprint similarity	[62]		✗	✓	✗	✗	✓	✗
Feature-based similarity	[69]		✗	✓	✗	✗	✓	✗
Graph Edit Distance	[56]	[6]	✗	✓	✓	✗	✓	✗
Element comparison	[64]		✗	✓	✓	✓	✗	✗
Jaccard index of flows	[7]	[70]	✗	✓	✗	✗	✓	✗
PME similarity	[65]	[8, 71]	✗	✓	✓	✗	✓	✗
Behavioral-footprint similarity	[31]	[67]	✗	✓	✗	✗	✓	✗
Embeddings similarity	[66]		✗	✓	✗	✗	✓	✗
Simulation + conformance	[10]	[1, 67]	✗	✓	✗	✗	✓	✗
Element counts	[71]		✗	✓	✓	✗	✓	✗
Round-tripping	[68]		✓	✓	✗	✗	✓	✗
LLM judge	[67]		✓	✗	✗	✗	✗	✓
Our approach	-	-	✓	✗	✓	✗	✓	✗

Table 2.5 Summary of evaluation methods for Process Modeling with their characteristics (Part 2). Exp: Expert-based, Str: String-based, Sem: Semantic, Given: activity labels are given, no matching is needed, Elem: Element-based, Struct: Structural, Beha: Behavioral, P.i.: Public implementation (checkmarks contain implementation links).

Metric	Label similarity				Type			P.i.
	Exp.	Str.	Sem.	Given	Elem.	Struct.	Beha.	
Expert evaluation	✗	✗	✗	✗	✗	✗	✗	-
Node-matching similarity	✗	✗	✓	✗	✓	✗	✗	✗
Graph Edit Distance similarity	✗	✗	✓	✗	✗	✓	✗	✗
Causal footprint similarity	✗	✗	✗	✗	✗	✗	✓	✗
Feature-based similarity	✗	✗	✗	✗	✗	✓	✗	✓
Graph Edit Distance	✗	✓	✗	✗	✗	✓	✗	✓
Element comparison	✓	✗	✗	✗	✓	✗	✗	-
Jaccard index of flows	✓	✗	✗	✗	✓	✗	✗	✗
PME similarity	✗	✗	✗	✗	✓	✗	✗	✓
Behavioral-footprint similarity	✗	✗	✓	✓	✗	✗	✓	✓
Embeddings similarity	✗	✗	✗	✗	✗	✓	✗	✓
Simulation + conformance	✗	✗	✗	✓	✗	✗	✓	✓
Element counts	✗	✗	✗	✗	✓	✗	✗	✓
Round-tripping	✗	✗	✗	✗	✗	✗	✗	✓
LLM judge	✗	✗	✗	✗	✗	✗	✗	✓
Our approach	✗	✗	✓	✗	✓	✗	✗	✓

2.4 Information Extraction

Given the structural similarities between PIE tasks and established Information Extraction techniques, and recognizing the limitations of current Process Modeling methodologies, we expanded the scope of our research to perform an extensive literature review on recent advances in Information Extraction. This broader perspective is particularly relevant since PIE fundamentally involves extracting structured information from text, a core objective shared with Information Extraction, and many recent advances with LLMs can inform improvements to PIE and PMG approaches.

Information Extraction encompasses several NLP tasks such as NER, RE, Event Extraction, and direct tuple extraction for knowledge graph construction. This domain has been profoundly impacted by LLMs, with the majority of recent studies leveraging generative models [72]. Even with recent advancements in NLP, Information Extraction remains a hard task and cannot be solved by simply prompting state-of-the-art LLMs [73].

In the following sections, we share our findings and highlight their impact on our experimental choices for both PIE and PMG.

2.4.1 Named Entity Recognition (NER)

At first glance, the PIE task of Mention Detection (MD) can seem very similar to NER. However, the two tasks differ in several important ways. In the NLP domain, mentions are not necessarily associated with a type, whereas PIE mentions are typed spans referring to process elements. The corresponding NLP task would be Mention Detection and Classification rather than NER. Unlike NER tasks where entities are discrete named items, MD involves tagging parts of sentences that are not necessarily proper nouns. Moreover, the usual NER classes do not overlap with PIE mention types, which also limits transfer learning between the two tasks. An example of classic NER is presented in Figure 2.2. Furthermore, large pretrained models have also been trained or fine-tuned on standard NER tasks, which hinders their performance on this non-standard setting, especially in zero-shot.

Nevertheless, we performed a literature review on recent advancements in the field and gathered relevant insights for our work. Notably, [74] presents a study that explores the integration of LLMs with human annotations, further bridging the gap between automated systems and human input. Additionally, [75] discusses data augmentation techniques that employ constraints to regulate the augmentation process, which is crucial for maintaining the quality of generated data. The construction of datasets from annotations is thoroughly examined in [76], highlighting systematic approaches to enhancing dataset quality. Furthermore, the PaDeLLM-NER model [77] significantly improves the reasoning speed for NER tasks through parallel decoding of all mentions. Lastly, we reference PromptNER, an earlier method based on fine-tuning BERT, and the self-improving capabilities illustrated in [78], which demonstrate the iterative advancements in this field.

2.4.2 Coreference Resolution (CR)

In this section, we focus on Within-Document Entity CR and do not include Event CR or Cross-Document CR, as they do not match the PIE objective. It is important to note that PIE entities are not exactly the same as NLP entities. In NLP, entities typically refer to real-world objects identified through NER (e.g., *Acme Corporation*), while PIE entities are groups of coreferent mentions (e.g., {the customer, he, the client}). This distinction explains why the original PIE study uses the term Entity Resolution, which does not correspond to the relevant NLP task; this was later corrected in [40].

CR datasets and models often include MD as a sub-step. This means that very few datasets and models exist for MD or CR only, and that CR is not directly accessible in most datasets.

[79] presents one of the latest and most efficient CR approaches. It leverages DeBERTa-

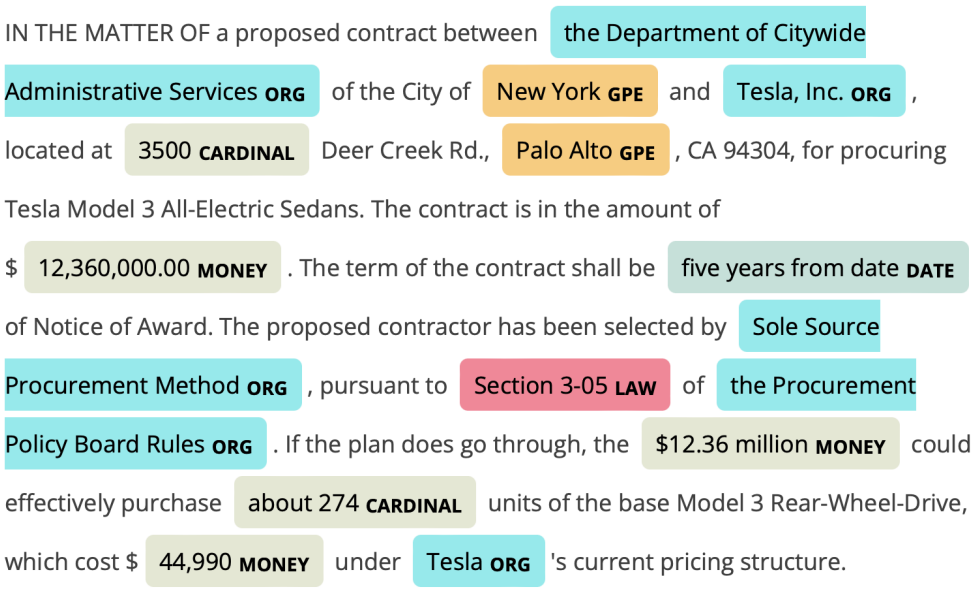


Figure 2.2 Example of classic NER task. The goal is to detect the entities in the text and classify them into predefined categories.

v3 as the LM backbone and careful design decisions (e.g., better pruning techniques when considering mentions) to achieve SOTA results on OntoNotes and other out-of-domain datasets. It especially emphasizes usability for downstream tasks, enabling their method to work with gold mentions leading to better performance on out-of-domain datasets.

2.4.3 Relation Extraction (RE)

RE can be performed as a standalone task (i.e., starting from the gold entities or mentions) or starting from the raw text. The latter case is called Joint RE and closely matches the PIE task since it involves the same three sub-steps (i.e., detecting mentions, grouping them into entities and linking them with relations).

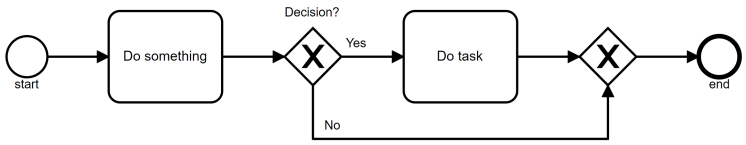
[45] presents JEREX, a Joint RE model that was SOTA at the time of publication. It was evaluated in [39] but was outperformed by PIE-specific techniques. This performance gap can be attributed to the domain-specific nature of PIE: process-related mention types and relation semantics differ substantially from general-purpose datasets like DocRED, and the limited size of the PET dataset prevents effective fine-tuning of Joint RE models.

This literature review has established the foundations for our experimental work. We examined the two main Process Modeling approaches, PIE and PMG, along with their respective evaluation methodologies and datasets. The analysis of existing evaluation methods revealed

a gap in model-to-description assessment, which we address in Chapter 4. Additionally, our review of Information Extraction techniques provides insights that inform our experimental design for PIE tasks. Before presenting our evaluation framework, we first investigate the choice of Process Model Representation, a foundational decision that directly influences PMG performance.

CHAPTER 3 PROCESS MODEL REPRESENTATION COMPARISON

The choice of Process Model Representation (PMR) is foundational in Process Modeling with LLMs, as it directly influences token efficiency, generation quality, and interpretability. This chapter presents a comprehensive comparison of nine PMRs in the context of Process Modeling with LLMs. The analysis combines both intrinsic evaluation of PMR characteristics and extrinsic evaluation through the PMG task. A subset of the evaluated PMRs is presented in Figure 3.1. These contributions have been accepted at AI4BPM 2025, a workshop of the BPM conference[1].




```
graph LR
    StartEvent_1(("start")) -->
     Task_1["Do something"]
    Task_1 --> Exclusive_1{"
     Decision?"}
    Exclusive_1 -->|Yes| Task_2["
     Do task"]
    Task_2 --> Exclusive_2{"X"}
    Exclusive_1 -->|No|
     Exclusive_2{"X"}
    Exclusive_2 --> EndEvent_1((""
     end"))
```

(a) Mermaid

```
from utils.model_generation import
      ModelGenerator

gen = ModelGenerator()

# Define activities
activity1 = gen.activity("Do
      something")
activity2 = gen.activity("Do task"
      )

# Create XOR gateway
xor_gateway = gen.xor(activity2,
      None)

# Define dependencies
dependencies = [(activity1,
      xor_gateway)]
```

(b) POWL code

```
<process>
    <activity action="Do something
     " id="activity1"/>
    <exclusiveGateway id="
     exclusivegateway1" decision=
     "Decision?">
        <branch condition="Yes" id
     ="branch1">
            <activity action="Do
     task" id="activity2"/>
        </branch>
        <branch condition="No" id=
     "branch2">
            <!-- Empty branch -->
        </branch>
    </exclusiveGateway>
</process>
```

(c) BPMN text

```
{
  "startEvent": {
    "name": "start"
  },
  "process": [
    {
      "type": "task",
      "name": "Do something"
    },
    {
      "type": "xor",
      "condition": "Decision?",
      "branches": [
        {
          "label": "Yes",
          "branch": [
            {
              "type": "task",
```

(d) JSON branches

Figure 3.1 Four Process Model Representations (PMRs) of the same process model.

The remainder of this chapter is organized as follows. Section 3.1 establishes key terminology. Section 3.2 presents the nine PMRs selected for evaluation. Section 3.3 introduces the PMo Dataset. Section 3.4 defines six requirements for an ideal PMR and evaluates each representation. Section 3.5 examines PMRs as generation targets for PMG. Section 3.7 discusses findings and limitations.

3.1 Initial Definitions

To ensure clarity throughout the rest of the chapter, we distinguish between the following concepts:

[1] A preprint is available at `https://doi.org/10.48550/arXiv.2507.11356`.

Process model: An abstract model of a business process, independent of any specific formalism or notation. For brevity, we occasionally refer to it simply as a *model*.

PMR: A notation used to express a process model (e.g., BPMN, POWL, or Mermaid). A PMR defines the syntax and semantics by which a process model is described.

PMR model: A process model represented using a PMR (e.g., a BPMN model, a Mermaid model).

Process model diagram: The graphical depiction of a process model, typically created from a PMR model (e.g., the diagram shown at the top of Figure 3.1). It offers a visual interpretation of the underlying process logic.

3.2 Process Model Representations

We choose to evaluate nine PMRs, presented in Table 3.1. Our selection is guided by two criteria. First, we include PMRs that are being used in existing PMG approaches, ensuring relevance to the current state of the art. Second, we ensure diversity in representation style, covering a range of structures and languages. In particular, we distinguish between graph-based PMRs, which explicitly define nodes and edges, and branch-based PMRs, which organize control flow through nested hierarchical structures.

Our analysis focuses on a specific set of element types commonly found in process models. These include tasks, events, gateways (with optional labels referred to as *decisions*) and sequence flows (with optional labels referred to as *conditions*). Furthermore, we consider swimlanes, which feature lanes nested within pools, and message flows. We do not include data objects, textual annotations or other special elements due to their limited prevalence in the PMo Dataset. We standardize the element identifiers used in most PMRs to the following format: *SubType_counter*, with individual counters for each subtype (e.g., *EndEvent_1*, *ParallelGateway_1*).

Hereafter, we describe the main PMRs used in this study.

BPMN is selected as our baseline given its status as a widely adopted standard. However, BPMN models vary in structure, with many including tool-specific extensions (e.g., SAP Signavio metadata). To ensure a fair evaluation within the Process Modeling context, we implement several standardization steps. These include removing superfluous attributes and elements such as text annotations and extension elements, rounding coordinates to integers within the BPMN diagram, and eliminating styling elements such as label styles.

BPMN process is a BPMN model that only defines the process and does not include the

Table 3.1 PMR main features. PMRs are ordered by recency. Exec.: Executable, Viz.: Vizualisable, Gen. schema: Generation schema.

PMR	Language	Graph based	Branch based	Exec.	Viz.	Gen. schema
BPMN	XML	✓	✗	✓	✓	✗
BPMN process	XML	✓	✗	✓	✗	✗
Graphviz	Graphviz	✓	✗	✗	✓	✗
Mermaid	Mermaid	✓	✗	✗	✓	✗
PME	JSON	✓	✗	✗	✗	✓
Simplified XML	XML	✓	✗	✗	✗	✗
POWL code	Python	✗	✗	✓	✗	✗
BPMN text	XML	✗	✓	✗	✗	✗
JSON branches	JSON	✗	✓	✗	✗	✓

BPMN diagram definition. Comparing BPMN and BPMN process allows us to directly measure the impact of specifying graphical elements.

Graphviz is a graphical representation, meaning it can be directly visualized using the DOT engine. Therefore, we pay special attention to the look of the visualized process model. The graph is structured from left to right, with tasks represented as rectangles and events as circles with names as labels. End events are depicted with a bolded outer circle, while intermediate events use a double circle, consistent with BPMN. Gateways are represented as follows: Exclusive (X or condition), Parallel (+), Event-based (E), Inclusive (O), and Complex (*). Swimlanes are represented as nested boxes.

Mermaid is another graphical representation made for diagrams. We follow the same rules as Graphviz, with the main difference being that styling properties (e.g., shape) are defined within the flow definitions rather than at the end of the PMR (see Figure 3.1).

PME [8] consists of a JSON object containing six lists of elements: tasks, events, gateways, swimlanes, sequence flows and message flows. This flat structure enables easy counting of elements and comparison between models.

Simplified XML is a simplification of XML originally proposed in [27]. Similarly to BPMN process, graphical and other non-essential elements are omitted. The XML structure is lightened, resulting in a more compact PMR.

POWL code, introduced by [1], is Python code that can be executed to create a POWL model, itself translatable to BPMN. The main advantage of using code as a PMR is to leverage LLMs' familiarity with coding tasks while enabling detailed feedback via execution of said

code. POWL code lacks support for conditions, event labels, intermediate events, swimlanes and message flows, making it the least expressive PMR.

JSON branches was developed by [9] specifically to reduce token count, be compact and provide schema following abilities for LLM generation. It only handles simple backward flows (represented as looping gateways) and cannot include multiple start and end events, message flows and swimlanes.

BPMN text [6] is similar to JSON branches, but in XML format. It has the same limitations except that it supports role and object annotations in tasks.

3.3 PMo Dataset

To enable PMR comparison across a wide range of process models, we build the PMo Dataset. PMo stands for Process Modeling. It contains 55 process descriptions paired with process models represented in all PMRs from Section 3.2. The PMo Dataset is constructed from five existing sources: 24 pairs from the Mangler dataset [51], 20 from the Process Modeling Benchmark [10], 6 from the PET-7 dataset [7][2], 4 from the BPMN for research dataset [55], and 1 from the CCC19 dataset [80].[3]

We chose these data sources because process models are handcrafted or at least validated by domain experts. This explains why we did not include the much larger MaD dataset [58], which has been criticized for lack of variability in its models and their descriptions. Similarly, even though the descriptions are human-authored, the dataset in [54] also lacks diversity due to automatic process model generation (e.g., maximum 9 activities).

The PMo Dataset underwent extensive preprocessing to ensure optimal usability and applicability. For process descriptions, this involves cleaning special characters, correcting punctuation and spacing, sentence splitting, and removing irrelevant information such as modeling instructions. The ground truth BPMN models are also refined by sanitizing label texts, improving diagram layouts and positioning decisions and conditions optimally. The Mangler dataset received special attention due to having multiple models per description, each graded from 0 to 5. As several models can be graded with the top grade, the model closest to the textual description (to our own judgment) is chosen among those, with a preference for models including only common elements (e.g., no data objects or other special elements).

Following the preprocessing phase described above, we automatically convert all BPMN models into all the other representations to obtain our ground truth. We develop converters

[2]One pair is removed because its description is already included in the Process Modeling Benchmark.
[3]The PMo Dataset is accessible at this URL: `https://doi.org/10.5281/zenodo.15857588`

for each PMR and validate our conversions by transforming models from BPMN to PMRs and back. Some process models contain information that is not supported by every PMR. In this case, we ignore the additional information (e.g., conditions are not included in POWL code). In cases where the model cannot be represented by the PMR without significant loss of information, it is left out (e.g., models including swimlanes are not converted to JSON branches). For branching PMRs, two-thirds of the BPMN models cannot be converted.

3.4 PMR for Process Modeling with LLMs

3.4.1 Requirements

To guide our evaluation, we define six key requirements for an ideal PMR in the context of Process Modeling with LLMs. These requirements aim to support various tasks within the Process Modeling pipeline, such as question answering, interactive modification and process model generation. A suitable PMR should be **compact** in terms of token count to fit within LLM context limits, reduce costs, generation time and energy consumption [9]. Strong **expressiveness** is essential to ensure that a wide range of models can be represented. This is mainly measured by element coverage, i.e., how many elements from a model are representable by the PMR. The PMR should be **human-readable** to support interpretability. More specifically, a non-expert should be able to understand the general purpose and structure of the process by just looking at the PMR model itself [7]. Length in lines, words and characters are key metrics to consider for this requirement, as a longer model is often harder to understand. Beyond pure text, a PMR should also be **vizualisable** (i.e., support direct graphical visualization), ideally without the need for additional conversions. The representation must be easily **usable**—parsable, editable, and generable by LLMs. This includes support for additional tooling such as JSON schema. Finally, it should be possible to **extend** it to fit specific modeling needs or add support for other process model elements.

In summary, our six requirements are Token compactness, Expressivity, Human readability, Visualization capabilities, Usability and Extensibility. These requirements are specific to the use case of Process Modeling with LLMs and are designed to support the whole range of associated tasks. They represent our attempt to differentiate between PMRs using the PMo Dataset. Other criteria could also be included such as compositionality, but it would require a more advanced evaluation framework and dataset.

3.4.2 Evaluation

We perform a comparison of PMRs across six dimensions that reflect the previously defined requirements. This evaluation combines both quantitative and qualitative metrics. For quantitative analysis, we measure the length of each representation in terms of lines, tokens, words, and characters to assess their compactness and efficiency. We also compute the mean element coverage over the PMo Dataset by dividing the number of process elements representable in the PMR by the total number of ground truth elements. For qualitative analysis, we evaluate human readability, visualization capabilities, usability and extensibility.

3.5 PMR for Process Model Generation

We aim to measure the performance of each PMR for the specific task of PMG. To ensure a fair comparison, we perform all experiments under a standardized setting. Specifically, we restrict process elements to the following set: standard tasks, start and end events, exclusive and parallel gateways, and sequence flows. This simplified setting allows us to compare PMRs performance on PMG independently of their element coverage, while also limiting prompt length and complexity. It would not be fair to compare a complex model generated in a PMR that supports swimlanes to a simpler model in a less expressive PMR.

3.5.1 Prompting

We design a PMG prompt that instructs the LLM to generate a process model represented in a specific PMR based on a given process description. To ensure a fair evaluation across PMRs, we construct prompts with consistent size and structure. The LLM receives the same task description and general modeling instructions regardless of the PMR. For each representation, we provide specific formatting guidelines which include Markdown sections and an example illustrating the expected output format.

3.5.2 LLM Choice and Generation Parameters

We perform our experiments with LLaMA-3.3-70B, which is a recent, open-source model with state-of-the-art performance for its size. Our choice to use an open-source LLM is motivated by transparency, adaptability, and community support, which facilitate reproducibility and further research. While it could be interesting to evaluate additional LLMs, this is out of the scope of this chapter. All experiments are performed using Google Vertex AI API. Generation parameters are set as follows: Top-K is left as default (i.e., -1), Top-P is 0.95 and Temperature

is 0.2.

3.5.3 Evaluation

We evaluate the generated PMR models through comparison with the ground-truth PMR models along two dimensions: element counts and PME similarity. Element counts capture the number of generated process elements for each type. To also take into account the semantic content of elements (i.e., task labels, decisions, conditions), we follow the evaluation methodology proposed by [8]. It consists of two steps: semantic matching and set similarity. Element pairs with semantic similarity higher than an experimentally defined threshold (0.7) are made identical using a 1-to-1 matching. To calculate this semantic similarity, we produce embeddings for each element using a sentence transformer trained for sentence similarity (`stsb-mpnet-base-v2`) and calculate their cosine similarity. Once the elements are semantically matched, the Dice-Sørensen coefficient is computed to assess similarity between the resulting sets, balancing precision and recall in a single score.

3.6 Results

3.6.1 PMR for Process Modeling with LLMs

Length comparison

Table 3.2 presents the length differences between ground-truth PMR models and our baseline, BPMN models. All PMRs lead to a significant reduction across all length metrics, successfully achieving their initial purpose. However, we observe substantial variation between PMRs: Mermaid is the most compact, achieving over 90% reduction in lines, tokens, words, and characters compared to BPMN. Looking at BPMN process, we can see that removing the BPMN diagram definition alone leads to a 70% decrease in token count. This highlights the importance of excluding non-essential elements such as graphical details.

POWL code, BPMN text, and JSON branches are reported in a separate section of the table as they do not cover all processes of the PMo Dataset, due to element coverage limitations. This explains the slight discrepancies between absolute and relative difference in the results. Nonetheless, they also achieve significant length reductions, comparable to Mermaid.

Evaluation summary

Table 3.3 summarizes our evaluation of PMRs across the six requirements defined in Section 3.4.1. We graded each requirement from 1 to 5 based on specific quantitative and

Table 3.2 Mean length difference of ground-truth PMR models compared to BPMN models. Rel. stands for Relative and Abs. for Absolute.

PMR	Line Count		Token Count		Word Count		Char Count	
	Rel.	Abs.	Rel.	Abs.	Rel.	Abs.	Rel.	Abs.
BPMN (Baseline)		384		4231		6531		19048
BPMN process	-64%	-248	-70%	-2984	-69%	-4539	-63%	-11947
Graphviz	-87%	-336	-88%	-3734	-90%	-5888	-88%	-16803
Mermaid	**-91%**	**-350**	**-93%**	**-3934**	**-93%**	**-6095**	**-92%**	**-17524**
PME	-27%	-109	-65%	-2783	-73%	-4768	-62%	-11799
Simplified XML	-60%	-228	-83%	-3502	-79%	-5111	-70%	-13207
POWL code	-87%	-280	-93%	-3268	-94%	-5107	-91%	-14471
BPMN text	-89%	-284	-89%	-3119	-93%	-5032	-88%	-13900
JSON branches	-67%	-212	-88%	-3078	-92%	-4973	-80%	-12215

qualitative metrics detailed in the same section.

Token compactness reflects the results from Table 3.2, with Mermaid and POWL code achieving the highest scores.

For expressiveness, we report the element coverage calculated on the PMo Dataset models. Branching PMRs such as POWL code and BPMN text score lower due to their limited element support (see Section 3.2).

Human readability scores are based on length results (Table 3.2) and ease of understanding the process solely from the PMR model. Branching PMRs perform best in this regard due to their concise syntax and clearer structure. We also take into account the user study from [7], where participants preferred Mermaid to Graphviz representation.

BPMN, Mermaid and Graphviz have the best visualization capabilities as they can be directly represented graphically. BPMN process and POWL code can be visualized via standard libraries such as pm4py, while the remaining PMRs need to be converted. The PMRs that are not graph-based are harder to convert, hence their reduced grade.

Usability scores are based on three aspects: parsability, editability and support for external tooling. Parsability is highest for JSON and XML-based formats, which benefit from standardized parsers. Custom formats like Mermaid and Graphviz are less straightforward, while branching PMRs are penalized due to the need for recursive parsing. We remove one point from BPMN for editability due to the need to update the associated BPMN diagram when making a change to the process. We also account for LLM schema-following capabilities (see Table 3.1) by adding an additional point to JSON PMRs.

Extensibility grades are based on the inherent limitations of PMRs and effort needed to support additional element types (e.g., it is harder to update the POWL generation logic than the PME JSON schema).

3.6.2 PMR for Process Model Generation

Element counts

Table 3.4 presents the mean element counts of the generated process models for each PMR. On average, LLMs generate significantly smaller models, with roughly eight fewer nodes (tasks, events, or gateways) compared to the ground truth. Gateways are the most affected: the number of Exclusive gateways drops by 50%, and Parallel gateways by 65%. This indicates a tendency to simplify the model by retaining a single task sequence. Despite this general trend, differences between PMRs are notable. Branching PMRs (i.e., JSON branches and BPMN text) lead to substantially higher element counts than the other representations. This suggests that the branching structure can partially mitigate LLMs' tendency to omit gateways. Interestingly, Graphviz stands out by producing more tasks and gateways than other graph-based PMRs. This may be attributed to its use of descriptive node names instead of identifiers, reducing generation complexity for the LLM.

PME similarity

Table 3.5 reports PME similarity scores obtained for different element types. Branching PMRs—particularly BPMN text—consistently achieve higher similarity scores than graph-based ones. Graphviz also shows strong performance for gateway decisions, likely benefiting once again from its use of descriptive identifiers. POWL code scores are lowered by the high number of formatting errors (40% of generated models are invalid), reflecting the PMR shortcomings with a limited prompting budget.

3.7 Discussion and Conclusion

This study offers a comprehensive analysis of PMRs in the context of Process Modeling with LLMs, with a particular focus on the PMG task.

To support this investigation, we assemble the largest gold-standard dataset to date for Process Modeling (55 pairs) coupled with various types of PMRs, enabling both qualitative and quantitative comparisons across nine PMRs drawn from the literature. Our analysis, guided by a set of functional requirements, identifies Mermaid as the most suitable PMR

Table 3.3 Evaluation summary of PMRs for LLM-based Process Modeling . All grades are from 1 to 5. Avg.: Average, Token comp.: Token compact, Exp.: Expressive, Viz.: Vizualisable, Ext.: Extensible.

PMR	Avg.	Token comp.	Exp.	Human readable	Viz.	Usable	Ext.
BPMN	3.33	1	**5 *(100%)***	1	**5**	3	**5**
BPMN process	3.50	2	**5 *(100%)***	2	3	4	**5**
Graphviz	3.67	4	4 *(89%)*	3	**5**	3	3
Mermaid	**4.00**	**5**	4 *(89%)*	4	**5**	3	3
PME	3.20	2	**5 *(100%)***	2	2	**5**	4
Simplified XML	3.50	3	**5 *(100%)***	3	2	3	**5**
POWL code	2.83	**5**	1 *(71%)*	3	3	3	2
BPMN text	2.67	4	2 *(84%)*	**5**	1	2	2
JSON branches	3.00	4	3 *(87%)*	**5**	1	3	2

for Process Modeling with LLMs, notably due to its token compactness and visualization capabilities.

We further examine the use of PMRs as generation targets for PMG. After generation based on standard prompting, we measure performance using process element counts and PME similarity. Results reveal a consistent tendency of LLMs to under-generate process elements—particularly gateways—leading to simplified process models. However, PMRs that support explicit branching structures, such as BPMN text and JSON branches, mitigate this behavior. BPMN text, in particular, achieves the strongest similarity with ground-truth elements for both raw numbers and semantic content.

Overall, our findings highlight the critical role of PMR design in enabling effective LLM-based process modeling and generation. While Mermaid appears to be the most versatile PMR, BPMN text yields better results for PMG specifically. This suggests that using different PMRs at specific stages of the Process Modeling pipeline may lead to more effective outcomes.

3.7.1 Limitations and Future Work

Several limitations must be acknowledged and provide avenues for future research.

First, our evaluation of PMRs for Process Modeling with LLMs (Table 3.3), while incorporating quantitative metrics, remains primarily subjective, as it is based on the assessment of the authors. A more rigorous approach, such as combining the grades from multiple experts, would enhance the validity of our findings.

Table 3.4 Mean difference in element counts between generated PMR models and ground truth. Nodes include tasks, events and gateways. Sequence flows are reported separately as they directly depend on the number of nodes and would artificially increase the number of elements.

PMR	**Nodes**	**Tasks**	**Events**	**Exclusive gateways**	**Parallel gateways**	**Sequence flows**
Ground Truth	23.18	12.53	2	5.91	2.67	27.67
BPMN	-10.49	-4.62	+0.18	-4.09	-1.98	-13.93
BPMN process	-9.93	-4.35	+0.14	-3.96	-1.69	-13.21
Graphviz	-7.96	-2.71	+0.04	-3.82	**-1.40**	-10.29
Mermaid	-9.16	-3.87	+0.07	-3.75	-1.53	-11.80
PME	-9.56	-4.02	+0.09	-3.73	-1.82	-12.69
Simplified XML	-9.49	-4.36	+0.09	-3.51	-1.64	-12.38
POWL code	-9.36	-4.08	**0.00**	-3.91	-1.67	-12.94
BPMN text	**-3.86**	**-1.53**	**0.00**	-0.41	-1.85	**-6.01**
JSON branches	-4.29	-2.44	**0.00**	**+0.09**	-1.87	-6.07
Average	-8.23	-3.55	+0.07	-3.01	-1.72	-11.04

Second, the structure of PMRs is inherently flexible, and our evaluation relies on standardized representations (e.g., consistent identifier formats). Alternative formatting choices may influence LLM behavior and lead to different results. Moreover, several promising PMRs, such as BPMN Sketch [81] or JSON-Nets [12], were not included in this study and represent valuable directions for future exploration.

Third, another limitation is that PME similarity, although it captures semantic content, remains highly sensitive to the number of generated elements. While most generated elements match the ground truth, omissions significantly reduce the overall score. For this reason, we also report element counts. Moreover, it enables us to better understand the under-generation problem of LLMs.

Fourth, we do not manually evaluate the LLM-generated models, which could provide better insights of their quality and practical usability.

Finally, our PMG experiments, like most existing studies, focus on a reduced subset of process elements. As such, they do not capture the full expressive range of real-world models. Future research should investigate LLMs' ability to generate richer models that include more advanced elements such as swimlanes or data objects.

Having established appropriate PMRs for Process Modeling with LLMs and identified their impact on generation performance, a fundamental question remains: how can we rigorously

Table 3.5 Mean PME similarity scores of generated PMR models compared to ground truth for each PMR under standard PMG. Gate.: Gateways, Seq.: Sequence.

PMR	Overall	Tasks overall	Events overall	Gate. overall	Gate. decisions	Gate. types	Seq. flows
BPMN	0.43	0.48	0.55	0.38	0.16	0.45	0.36
BPMN process	0.44	0.48	0.55	0.4	0.15	0.5	0.38
Graphviz	0.47	0.5	0.74	0.49	**0.6**	0.54	0.38
Mermaid	0.48	0.49	0.74	0.45	0.18	0.54	0.42
PME	0.46	0.5	0.68	0.45	0.12	0.54	0.37
Simplified XML	0.45	0.48	0.57	0.46	0.07	0.58	0.4
POWL code	0.27	0.29	0.41	0.21	0.36	0.23	0.22
BPMN text	**0.54**	**0.52**	**0.75**	0.58	0.13	0.69	0.49
JSON branches	0.53	0.51	0.58	**0.61**	0.19	**0.7**	**0.5**

evaluate whether generated process models faithfully capture the semantic content of their source descriptions? While this chapter assessed PMRs through structural metrics and element-level similarity, these measures do not directly address information preservation from textual descriptions to generated models. The following chapter introduces a novel evaluation framework grounded in Process Information extracted from source texts, enabling systematic assessment of how well PMG approaches preserve the mentions, entities, and relations present in process descriptions.

CHAPTER 4 PROCESS MODELING EVALUATION

As discussed in Chapter 2, existing PMo evaluation approaches rely on structural metrics or domain-independent quality criteria, lacking methodologies that assess how faithfully generated process models capture the semantic content of their source descriptions. We propose a fine-grained evaluation framework grounded in Process Information to address this gap. Specifically, we compare the fine-grained mentions, entities and relations extracted from textual descriptions with the information in generated process models, providing a semantic basis for evaluation that complements structural approaches.

This evaluation framework requires addressing three methodological challenges. First, we establish techniques for matching semantically equivalent but lexically different process elements, as LLMs reformulate content during PIE and PMG. Second, we match PI with generated models by either extracting PI from models or reformatting PI into a representation closer to process models. Third, we validate the reliability of our extraction and matching methodologies before evaluating PMG approaches. This chapter presents our solutions through experiments spanning PIE evaluation, process model information extraction, and assessment of three PMG approaches.

4.1 PIE Semantic Evaluation

Before using PI to evaluate generated models, we must establish how to reliably assess PI itself. Traditional PIE evaluation metrics require exact or near-exact text span matches, penalizing semantically equivalent reformulations produced by LLMs. This section addresses this limitation by developing a semantic evaluation methodology based on similarity metrics that recognize equivalence despite surface-level variations.

We focus on mention-level evaluation, as mentions are the foundational units from which entities and relations are constructed. Correctly identifying and matching mentions is crucial for the PIE evaluation pipeline. Through systematic experimentation with similarity metrics, we establish a matching methodology that forms the basis for all evaluation tasks in this chapter.

4.1.1 Existing Metrics

Existing works [44, 82, 83] use three evaluation settings to evaluate extracted mentions: *strict*, *relaxed* and *partial*.

- *strict*: The mention span must match exactly with the gold standard mention.
- *relaxed*: Small variations in the mention span are allowed. One missing or additional word at the beginning or end of the mention is considered valid (e.g., "the data manager", "data manager of" or "manager" are valid matches for "data manager"). This setting is used in the original PET paper [5].
- *partial*: Any overlap between the predicted mention and the gold annotation is counted as a true positive.

These three metrics provide a baseline but are insufficient to capture semantic equivalence of mentions.

4.1.2 Mention Matching

LLMs reformulate mentions during PIE and PMG, producing variations that are semantically identical or similar to gold standard annotations but differ in text spans. Table 4.1 illustrates examples of such reformulations.

Table 4.1 Examples of mention reformulations by LLMs

Gold mention	**LLM reformulation**
A customer	Customer
brings in	brings
a defective computer	defective computer
the data manager	data manager
is sent to	sent to

These reformulations involve linguistic transformations such as article removal (e.g., "the", "a", "an"), verb variations (e.g., "brings in" → "brings"), or noun simplification (e.g., "a defective computer" → "defective computer"). While these variations do not alter semantic meaning, existing metrics penalize them, underestimating model performance.

To address this limitation, we leverage the label matching techniques from Section 2.3.1 and adapt them to mention evaluation. Our approach computes semantic similarity between mentions and considers them equivalent when similarity exceeds a predefined threshold. Formally, given a predicted mention m_{llm} and a gold standard mention m_{gold}, we define:

$$m_{\text{llm}} \equiv m_{\text{gold}} \iff \text{sim}(m_{\text{llm}}, m_{\text{gold}}) > \theta \tag{4.1}$$

where $\text{sim}(\cdot, \cdot)$ is a similarity function and θ is a threshold parameter. The challenge is selecting a similarity metric and determining the optimal threshold, which we address through systematic experimentation.

4.1.3 Similarity Metrics

To evaluate mention equivalence, we establish a suitable similarity metric. The ideal metric should satisfy three requirements:

Semantic sensitivity with lexical independence. Mentions in the PET dataset range from single words (e.g., "I") to complex noun phrases (e.g., "according to the current files of the leading manager"). The similarity metric must detect semantic similarity while remaining independent of lexical variations. Minor surface-level changes such as article removal or reordering should not significantly affect similarity scores when core meaning remains unchanged.

Threshold stability. Small threshold variations should not drastically change matching performance. Similarity scores should be well-distributed rather than clustered around a single value, enabling robust threshold selection and consistent performance across datasets.

Computational efficiency. With approximately 35 mentions per document in the PET dataset, evaluating a single document requires computing similarity for $\binom{35}{2} = 595$ pairs. Across all 45 documents, this yields approximately 26,775 pairwise comparisons. The metric must remain computationally tractable at this scale, and scale to larger datasets where comparison counts grow quadratically.

We evaluate several categories of similarity metrics against these requirements. As baselines, we include *strict* (exact match), *relaxed* (one word difference tolerance), and *partial* (any word overlap).

Traditional NLP metrics. We consider several established string-based and n-gram-based metrics from the NLP literature. Levenshtein distance measures the minimum single-character edits to transform one string into another. BLEU [84], ROUGE [85], and METEOR [86] compute n-gram overlap with various normalization strategies. CHRF [87] operates at the character level. BERTScore [88] computes token-level semantic similarity using contextualized embeddings, then aggregates via greedy matching. While these metrics are efficient, they are either primarily character-based (Levenshtein, CHRF) or limited to token-level matching (BERTScore), lacking holistic semantic understanding for mention-level comparison.

Sentence transformers with cosine similarity. Embedding-based approaches encode text into dense vectors, with semantic similarity measured via cosine similarity. We evaluate

several pretrained models:

- `sts-mpnet-base-v2` (109M parameters): Trained on sentence similarity tasks using Sentence-BERT [89].
- `bge-base-en-v1.5` (109M parameters): From the BGE family [90], trained on diverse retrieval and similarity tasks.
- `PEARL` (109M parameters): Designed for phrase embeddings [91].
- `potion-base-8M` (8M parameters): A distilled version of `bge-base-en-v1.5` using static embeddings [92].

Cross-encoder models. Unlike bi-encoders that encode texts independently, cross-encoders process both texts jointly, enabling richer interaction modeling at increased computational cost. We evaluate `stsb-roberta-base` (125M parameters), a RoBERTa model [93] fine-tuned on the Semantic Textual Similarity Benchmark [94]. Cross-encoders require $O(n^2)$ forward passes for n mentions, compared to $O(n)$ for bi-encoders.

Our similarity metrics span from traditional string-based approaches to advanced embedding-based models.

4.1.4 Mention Similarity Dataset

To evaluate similarity metrics, we require a dataset containing mention pairs with known semantic equivalence relationships. Such a dataset comprises two sets:

Positive pairs consist of ground truth mentions paired with semantically equivalent variations.

Negative pairs consist of ground truth mentions paired with semantically distinct mentions from the same domain to establish decision boundaries and prevent false positives.

Ground truth mentions are from PET, the only source of annotated ground truth mentions.

Positive pair generation

We construct positive pairs by applying systematic perturbations to ground truth mentions, mimicking reformulations produced by LLMs during PIE and PMG:

- **Article manipulation**: Adding or removing determiners (e.g., "customer" ↔ "the customer").

- **Capitalization variation**: Modifying first character case (e.g., "data manager" ↔ "Data manager").
- **Specification removal**: Truncating prepositional phrases (e.g., "files of the manager" → "files").
- **Plural variation**: Removing or adding terminal "s" (e.g., "computers" ↔ "computer").

Each mention is subjected to these transformations, generating one noised variant per original mention.

Negative pair generation

Constructing negative pairs is challenging. A naive approach would pair each ground truth mention with all other ground truth mentions. However, different PET mentions can be semantically equivalent (e.g., check versus checks).

We employ a sampling methodology based on part-of-speech tags:

1. **Mention removal**: We identify all tokens that are part of annotated mentions and exclude them.
2. **POS-filtered sampling**: From remaining tokens, we select candidates based on part-of-speech tags provided in PET. We sample verb tokens as negative examples for Activity mentions, and noun tokens as negative examples for Activity Data and Actor mentions.

This ensures negative examples are from the same linguistic domain and lexical category while maintaining semantic distinction.

Table 4.2 illustrates examples of negative mentions extracted using this methodology alongside their corresponding PET mentions from the same document.

Table 4.2 Examples of mentions extracted using POS-tag-based sampling from a PET document

PET mentions	Noised mentions	Negative mentions
checks	Checks	manufactures
takes	take	continues
a defective computer	Defective computer	repair
the defect	Defect	activities
A customer	the Customer	process
the CRS	CRS	repair

Dataset limitations

The constructed dataset has limitations. Our negative sampling strategy produces reliable negative examples only for Activity , Activity Data , and Actor mention types. The remaining types (Further Specification , XOR Gateway , AND Gateway , Condition Specification) lack sufficient quantities or distinguishable POS patterns. Thus, similarity metric evaluation is restricted to the three most prevalent mention types, representing 83.8% of all mentions in PET (see Table B.1).

The dataset also inherits PET's limitations, notably its small size.

4.1.5 Similarity Metric Evaluation

Using the constructed dataset, we evaluate similarity metrics across multiple experiments reflecting the three requirements from Section 4.1.3.

Pair ranking

In realistic evaluation scenarios, we compute similarity between a set of LLM-generated mentions $M_{\text{pred}} = \{m_1, m_2, \ldots, m_n\}$ and ground truth mentions $M_{\text{gold}} = \{g_1, g_2, \ldots, g_k\}$, where at most one predicted mention matches each ground truth mention. To evaluate discriminative power, we simulate this using noised mentions as approximations of LLM reformulations. We test whether each metric assigns higher scores to positive pairs (ground truth and noised variant) than to negative pairs (ground truth and semantically distinct mentions).

Formally, for each ground truth mention $g_i \in M_{\text{gold}}$ with its corresponding noised variant p_i (representing a typical LLM reformulation) and a set of negative examples $N_i = \{n_{i,1}, n_{i,2}, \ldots, n_{i,m}\}$, we define a correct ranking as:

$$\text{sim}(g_i, p_i) > \max_{n \in N_i} \text{sim}(g_i, n) \tag{4.2}$$

We report ranking accuracy as the proportion of mentions for which this condition holds:

$$\text{Accuracy} = \frac{1}{|M_{\text{gold}}|} \sum_{i=1}^{|M_{\text{gold}}|} \mathbb{1}\left[\text{sim}(g_i, p_i) > \max_{n \in N_i} \text{sim}(g_i, n)\right] \tag{4.3}$$

where $\mathbb{1}[\cdot]$ is the indicator function that equals 1 when the condition is true and 0 otherwise.

Figure 4.1 presents the ranking accuracy results, providing a threshold-independent assessment

of each metric's discriminative power.

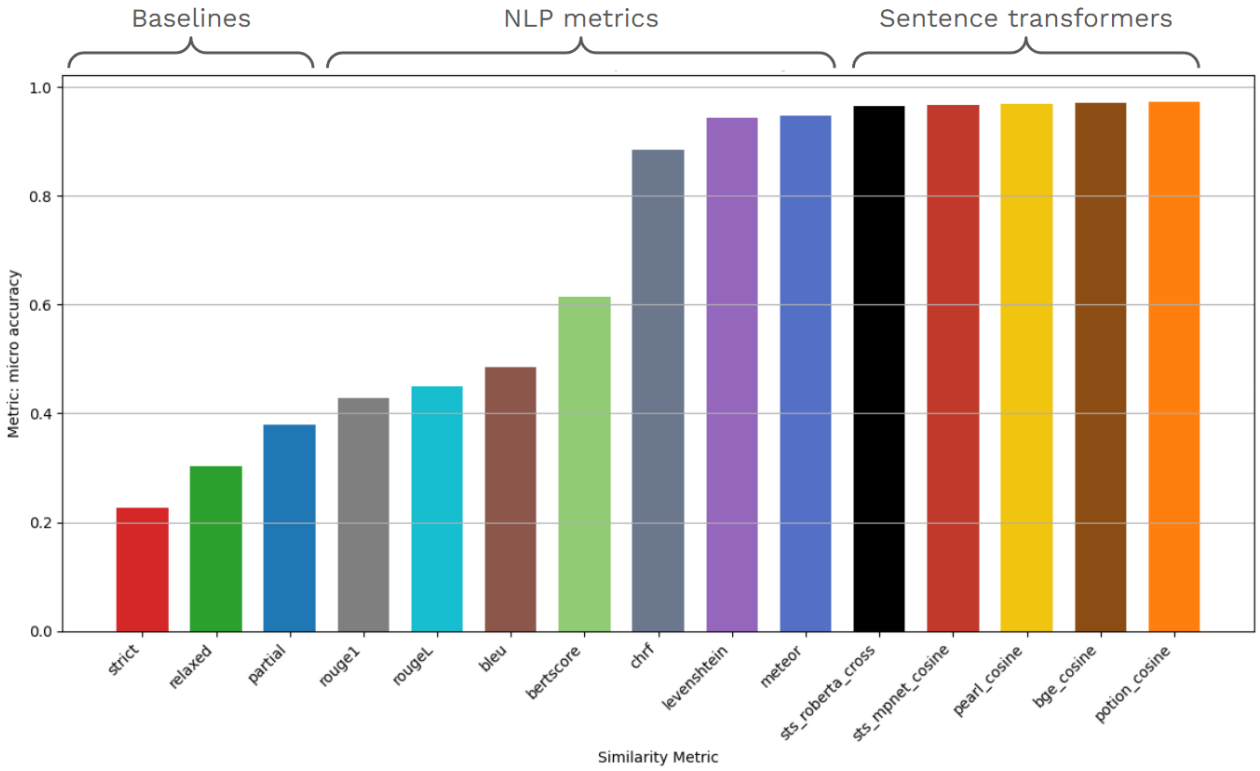


Figure 4.1 Ranking accuracy of similarity metrics

The results reveal clear performance tiers across metric categories. Baseline metrics perform poorest, with *strict* achieving 0.23 accuracy, *relaxed* reaching 0.30, and *partial* achieving 0.38. These low scores confirm that exact or near-exact text matching is insufficient for handling LLM reformulations, as even minor variations cause ranking failures.

Traditional NLP metrics show improved performance. ROUGE-L achieves 0.43 accuracy and BLEU reaches 0.45. BERTScore achieves the highest accuracy among traditional metrics at approximately 0.61, though still below embedding-based approaches. CHRF reaches approximately 0.89 while Levenshtein distance demonstrates notably higher performance at 0.93, benefiting from our noise patterns that introduce small character-level changes while preserving most characters. Finally METEOR is the best performing traditional NLP metric at 0.94.

Sentence transformers consistently achieve the highest ranking accuracies, clustering between 0.96 and 0.97. The four pretrained sentence transformers demonstrate near-perfect discriminative power: `sts-roberta-base` achieves 0.96, `sts-mpnet-base-v2` reaches 0.97, `bge-base-en-v1.5` attains 0.97, and `PEARL` achieves 0.97. The distilled model `potion-base-8M`

achieves the highest ranking accuracy at 0.97, demonstrating that model distillation preserves discriminative capabilities while reducing computational requirements.

The cross-encoder `stsb-roberta-base` achieves 0.96 accuracy, slightly below the best sentence transformers. This suggests that bi-encoders with high-quality pretrained embeddings can match or exceed cross-encoder performance for mention matching while offering superior computational efficiency.

Threshold selection

Beyond ranking accuracy, we determine the optimal threshold θ above which mentions are considered equivalent. Since thresholds depend on metric scale and distribution, we evaluate multiple values for each metric.

Our evaluation reproduces the label matching methodology from Section 2.3.1. We construct candidate mentions $M_{\text{cand}} = M_{\text{pred}} \cup M_{\text{neg}}$ (noised and negative mentions) and target mentions $M_{\text{target}} = M_{\text{gold}} \cup M'_{\text{neg}}$ (PET and additional negative mentions). We then perform:

1. Group mentions by type (e.g., Activity , Activity Data)
2. Compute the pairwise similarity matrix between M_{cand} and M_{target}
3. Sort all mention pairs by similarity score in descending order
4. For each pair, match the candidate mention to a target mention if their similarity exceeds the threshold (greedy matching)

This setup ensures performance degrades at inappropriate thresholds: if too high, correct matches are missed (decreasing recall); if too low, incorrect matches occur (decreasing precision). This mirrors actual usage where LLM-generated mentions are matched to ground truth in the presence of confusing alternatives. We compute precision, recall, and F1-score across threshold values to identify optimal operating points. We report F1-score as it balances precision and recall. The three baseline metrics are threshold-independent, achieving F1-scores of 0.56 (*strict*), 0.60 (*relaxed*), and 0.67 (*partial*), serving as lower bounds.

Figures 4.2 and 4.3 present the F1-scores across different threshold values for traditional NLP metrics and sentence transformers respectively.

Among traditional NLP metrics, BLEU peaks at F1 = 0.70. BERTScore peaks at $\theta = 0.9$ with low performance. CHRF achieves limited performance (F1 = 0.60) due to its character-only basis. ROUGE achieves F1 = 0.68. Levenshtein distance achieves F1 = 0.93 at $\theta = 0.5$,

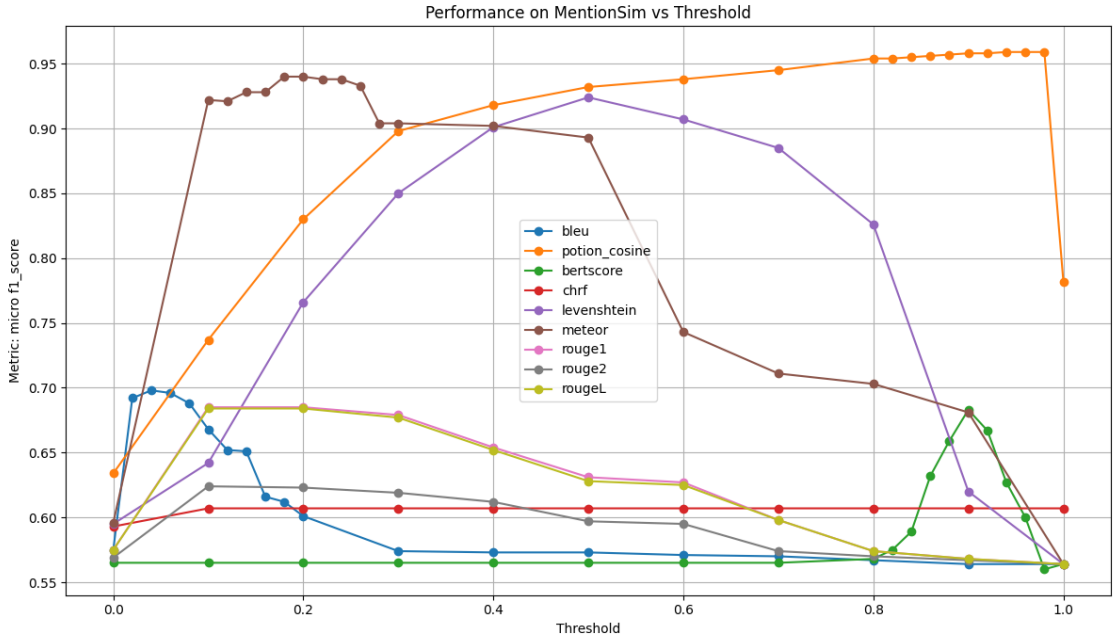


Figure 4.2 Traditional NLP similarity metrics F1-score on mention matching task across multiple thresholds. Best performing metric `potion-base-8M` is kept for comparison.

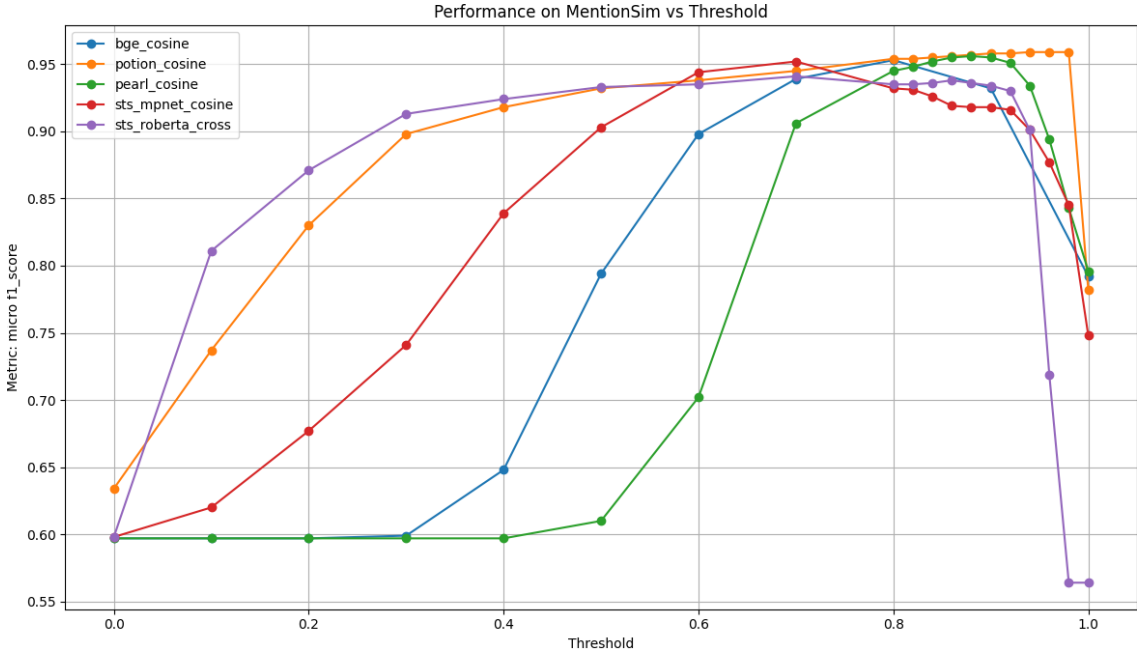


Figure 4.3 Sentence transformer similarity metrics F1-score on mention matching task across multiple thresholds.

benefiting from noise patterns introducing small character-level changes. METEOR achieves the best performance among traditional metrics (F1 = 0.94 at $\theta = 0.2$) through synonym matching and paraphrase recognition.

Sentence transformers demonstrate superior performance. `potion-base-8M` achieves the highest F1-score (0.97 at $\theta = 0.86$) with the most stable performance across thresholds. `PEARL`, `bge-base-en-v1.5`, and `sts-mpnet-base-v2` achieve F1 = 0.96. `PEARL` and `bge-base-en-v1.5` peak around $\theta = 0.86$ with greater threshold sensitivity, while `sts-mpnet-base-v2` peaks at $\theta = 0.7$, validating findings from [8]. The cross-encoder `stsb-roberta-base` achieves F1 = 0.93 at $\theta = 0.7$ but exhibits sharp performance degradation outside this range.

Computational efficiency

We measure execution time for the threshold selection experiment. Figure 4.4 presents results.

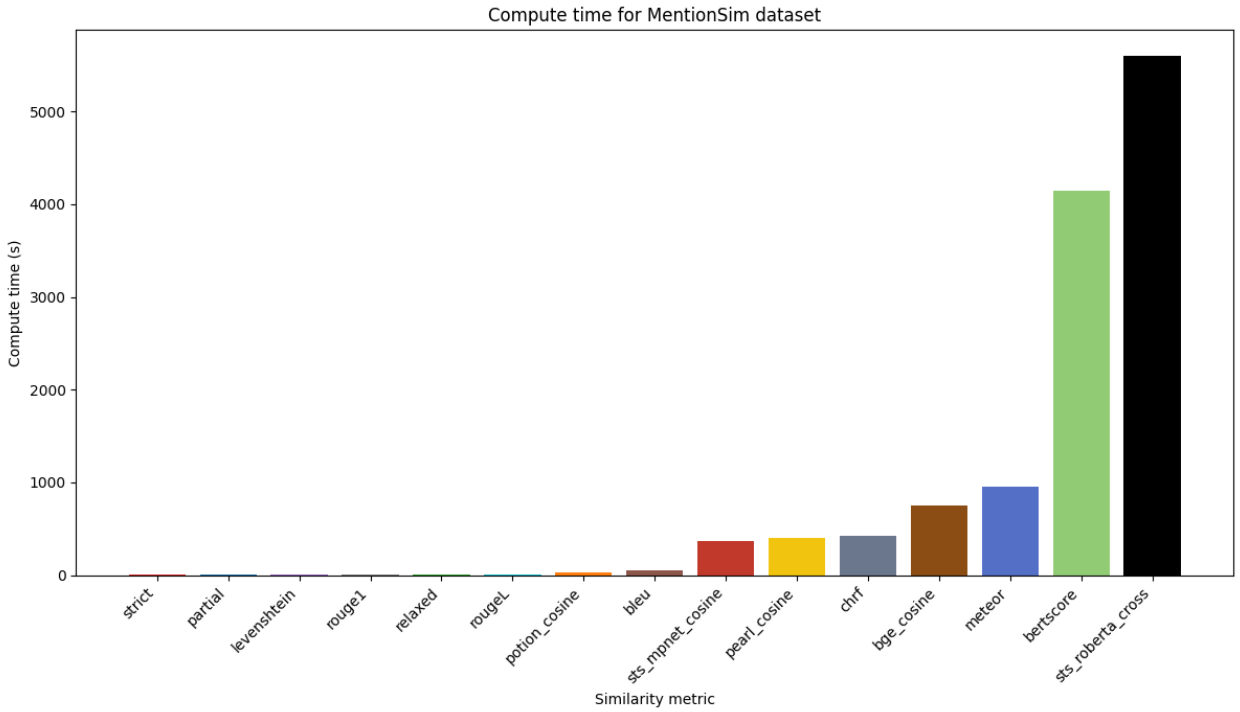


Figure 4.4 Computation time for similarity metrics on the mention matching task

Traditional string-based metrics demonstrate exceptional computational efficiency, with *relaxed*, *strict*, *partial*, Levenshtein, ROUGE-L, and BLEU all completing in under 50 seconds. CHRF shows slightly increased computation time at approximately 200 seconds due to word splitting operations. METEOR exhibits longer execution time at approximately 900 seconds due to preprocessing steps including word splitting and lemmatization. BERTScore is substantially

slower at approximately 4,200 seconds, requiring BERT embedding computation for each mention pair without precomputation benefits.

Among embedding-based models, the sentence transformers demonstrate a clear efficiency advantage over the cross-encoder. `potion-base-8M` in 90 seconds, `sts-mpnet-base-v2` in 310 seconds, and `PEARL` in 350 seconds. `BGE` reaches 750 seconds, approximately eight times slower than `potion-base-8M`. The cross-encoder `stsb-roberta-base` exhibits significantly higher computation time at approximately 5,850 seconds due to its $O(n^2)$ complexity, requiring forward passes for all mention pairs rather than precomputed embeddings.

The efficiency results strongly favor `potion-base-8M`, providing that mention embeddings are cached and reused for all comparisons. Indeed this drastically limits the computational complexity and enable scaling to much larger datasets.

4.1.6 Discussion

We select `potion-base-8M` with $\theta = 0.86$ for mention matching based on its superior performance across all three requirements. The model achieves the highest F1-score (0.97) and ranking accuracy (0.97) with the most stable performance across thresholds, while completing the threshold selection experiment in 90 seconds. While METEOR (F1 = 0.94) and Levenshtein (F1 = 0.93) achieve competitive F1-scores, they demonstrate substantially lower ranking accuracy (0.49 and 0.62 respectively) and lack semantic understanding for diverse reformulations beyond character or word-level variations. The model's static embeddings and distilled architecture make it well-suited for large-scale evaluation, offering optimal performance at low computational cost.

For the remainder of this work, we employ a semantic evaluation methodology based on this matching approach. Rather than requiring exact text span matches as in *strict* evaluation, we match mentions using `potion-base-8M` cosine similarity with $\theta = 0.86$, then compute precision, recall, and F1-score. This methodology better reflects semantic equivalence of LLM-generated mentions while maintaining rigor through validated threshold-based decision boundaries.

4.2 Evaluated PMG Approaches

Having established a robust methodology for evaluating PI, we now apply it to assess PMG approaches. This section presents the three evaluated approaches, selection rationale, and implementation choices.

4.2.1 Selection Rationale

We select three PMG approaches: BPMN-chatbot [9], ConverMod [7], and ProMoAI [1]. These were chosen based on criteria ensuring validity and comprehensiveness.

Availability and reproducibility. All three approaches provide publicly available implementations or detailed documentation. Each supports multiple LLMs as backends, allowing evaluation under consistent conditions using Gemini 2.0 Flash.

Temporal and technical diversity. The approaches span different development periods and employ fundamentally different PMRs and generation techniques:

- **BPMN-chatbot** [9] uses *JSON branches*, encoding process branching in JSON. It leverages LLM function calling for schema adherence with explicit validation.
- **ConverMod** [7] employs *Mermaid* or *Graphviz* with standard prompt engineering. The authors investigate multiple prompt variations and select the best-performing one.
- **ProMoAI** [1] generates *POWL code*, Python code executed to produce process models in POWL, translatable to BPMN. It uses extended prompts and iterative error correction based on execution feedback.

Continued relevance and adoption. All three approaches remain actively used, demonstrating their value. BPMN-chatbot serves as foundation for BPMN-chatbot++ [95]. ConverMod's methodology is integrated into AutoBPMN.AI [96]. ProMoAI is widely adopted as a benchmark [10] and used to explore techniques such as reinforcement learning [67].

This diversity in PMRs and generation techniques makes these approaches ideal candidates for evaluation using our PI-based methodology.

4.2.2 Implementation and Experimental Setup

We establish a consistent experimental setup for fair and reproducible comparison.

Language model selection. We employ Gemini 2.0 Flash for all three approaches. Gemini 2.0 represents state-of-the-art performance among publicly available LLMs and provides robust tool support including function calling required by BPMN-chatbot. While closed-source, this aligns with the design philosophy of the evaluated approaches, which leverage state-of-the-art closed-source LLMs. Using a consistent backend isolates the impact of different PMRs and generation strategies.

Implementation adaptations. We adapted original implementations to support Gemini 2.0 while preserving core methodologies. For BPMN-chatbot, we used Gemini's function

calling for constrained generation with JSON schema validation. For ProMoAI, we preserved iterative error correction and POWL code generation, adjusting only API calls.

Evaluation dataset. We evaluate all approaches on the complete PET dataset (45 descriptions), substantially larger than subsets used in original papers (PET-7 in ConverMod and BPMN-chatbot, 2 descriptions in ProMoAI). This enables robust statistical analysis and captures variability in description complexity.

Reproducibility considerations. All experimental code, adapted implementations, and evaluation scripts are available. Consistent use of Gemini 2.0 combined with our deterministic PI-based methodology ensures findings can be independently verified.

4.3 Process Model Information Extraction

Our first approach to evaluating PMG consists of extracting Process Information (i.e. mentions, entities and relations) from generated process models and comparing it with ground truth Process Information from textual descriptions. To achieve this, we develop Process Model Information Extraction (PMIE), a methodology derived from PIE. This section describes the PMIE methodology, evaluates extraction performance, and presents evaluation results on PMG approaches.

4.3.1 Methodology

PMIE consists of two stages: first, converting process models from their native PMR into Process Model Elements; second, extracting Process Information (PI) from these elements. Process Model Elements are the core semantic components of process models: tasks, events, gateways and sequence flows, stripped of format-specific syntax. For example, a Mermaid node `Task1["Manager reviews application"]` becomes a task element with label "Manager reviews application", from which PMIE extracts the Activity mention reviews, the Actor mention Manager, and the Activity Data mention application. This two-stage approach enables consistent information extraction across PMRs while leveraging LLM semantic understanding. An overview of the PMIE methodology is presented in Figure 4.5.

Process Model Elements

Process Model Elements (PME) serve as a unified representation abstracting structural and graphical details of PMRs while preserving semantic content. This addresses challenges in cross-PMR information extraction.

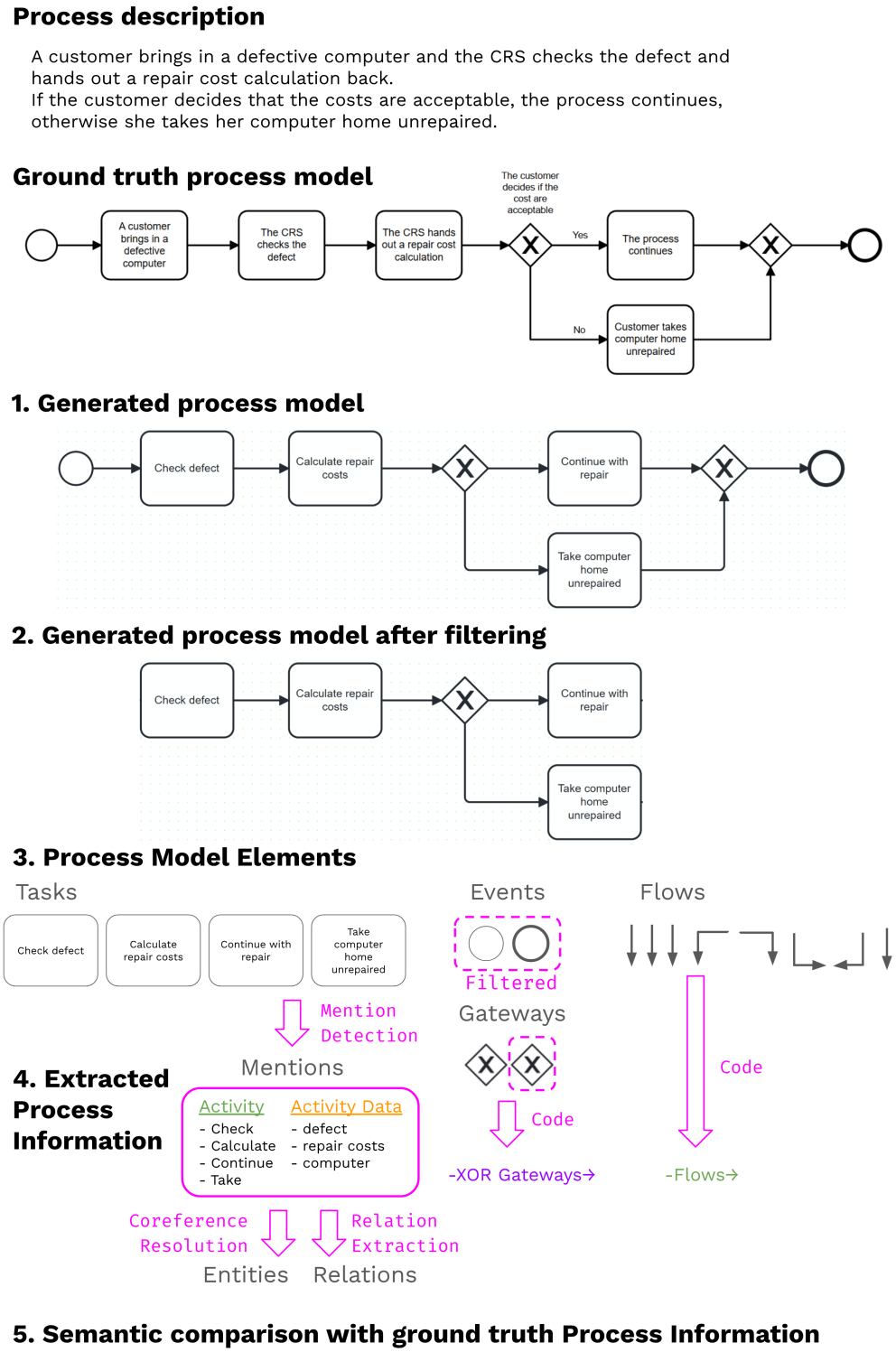


Figure 4.5 Overview of the PMIE evaluation process. Extraction steps are in pink.

Motivation. Different PMRs encode process models in different formats: BPMN uses XML, Mermaid uses a domain-specific language, JSON branches uses nested JSON, and POWL code uses Python. These variations create two problems: LLMs must parse complex formats (e.g., BPMN with graphical coordinates), and developing separate extraction pipelines for each PMR is impractical and prevents comparative evaluation.

PME provide a format-agnostic representation that simplifies parsing, enables unified processing, and facilitates bidirectional conversion. PME reduces structural complexity by focusing on semantic content, removing layout information, namespace declarations, and format-specific artifacts irrelevant for information extraction. A single extraction pipeline processes models from all PMRs after conversion to PME, ensuring consistent evaluation across PMG approaches. PME can be converted from any PMR and back to standard representations, making it suitable for analysis and generation.

Conversion from PMRs. We implement deterministic conversion procedures from each PMR to PME. We parse Mermaid (ConverMod) to extract task labels, gateway types, and sequence. For JSON branches (BPMN-chatbot), we traverse the nested structure and flatten it into sequential elements with explicit control flow. For POWL code (ProMoAI), we execute Python code to obtain a POWL object, then extract activities and control flow dependencies. These conversions are preprocessing steps transforming any input model into PME before extraction.

PI extraction from PME

Given a process model in PME format, we extract PI through LLM-based and rule-based operations. The extraction pipeline mirrors PIE (see Section 2.2.1) but operates on structured model elements rather than free-form text.

Filtering. We apply structural simplifications to focus on semantically meaningful elements. Start and end events are excluded as they carry no process information. Closing gateways (merging parallel or exclusive branches) are removed since they serve only structural purposes. This reduces noise and focuses extraction on information-bearing elements. An example of process model before and after filtering is presented in Figure 4.5.

Mention extraction from tasks. For each task, we extract mention types from its label using LLM prompting:

- Activity mentions: core action or verb phrase describing what is done
- Activity Data mentions: data objects, documents, or artifacts involved

- Actor mentions: roles, persons, or organizational units performing the activity
- Further Specification mentions: temporal, spatial, or modal qualifications

For example, given "Manager reviews submitted application", extraction identifies: reviews, application, and Manager. We employ few-shot prompting with explicit definitions and JSON-formatted output constraints (see prompt in **??**.

Mention extraction from gateways. Opening gateways (exclusive or parallel splits) with condition labels are processed to extract gateway-specific mentions. For exclusive gateways, we extract XOR Gateway mentions from gateway labels and Condition Specification mentions from outgoing sequence flow labels. For example, given an exclusive gateway labeled "Decides if the costs are acceptable" with outgoing flows labeled "Yes" and "No", we extract Decides if the costs are acceptable as a XOR Gateway mention and Yes, No as Condition Specification mentions. This preserves branching logic information crucial for faithful representation.

Entity construction. Following mention extraction, we group Actor and Activity Data mentions (the only entity types in the PET annotation schema) into entities through an additional prompting step. We prompt the LLM to cluster mentions referring to the same real-world entity, producing unique entities for each type. This addresses coreference issues and ensures relations are established between distinct entities.

Sequence flow relation extraction. We extract $\overrightarrow{\text{Sequence Flow}}$ relations directly from PME by linking Activity mentions of tasks connected by sequence flows. For example, given a sequence flow connecting "Brings in defective computer" to "Checks the defect", we extract a $\overrightarrow{\text{Sequence Flow}}$ relation between Brings in and Checks. This leverages explicit control flow representation in PME, producing relations corresponding to $\overrightarrow{\text{Sequence Flow}}$ relations in PIE.

Intra-task relation extraction. For relations within tasks, we employ a hybrid approach combining deterministic rules with LLM-based classification. Two relation types use straightforward rules based on mention co-occurrence: $\overrightarrow{\text{Uses}}$ relations are established between each Activity and Activity Data mention within the same task, and $\overrightarrow{\text{Further Specification}}$ relations are established between each Activity and Further Specification mention within the same task. These rule-based extractions assume all mentions within a task are semantically related to the activity, which holds for well-formed process models where task labels are self-contained descriptions.

The most complex case is Actor Relation Classification, distinguishing between $\overrightarrow{\text{Actor Performer}}$ (the actor performs the activity) and $\overrightarrow{\text{Actor Recipient}}$ (the actor receives or is affected by the activity). This distinction cannot be made through lexical rules alone as it depends on

semantic understanding. We employ LLM-based classification, prompting the model with the task label and extracted mentions to determine the relation type. The prompt includes definitions and few-shot examples (see Appendix D).

The complete PMIE pipeline produces mentions, entities, and relations that can be directly compared against ground truth PI from PET, enabling quantitative evaluation of generated process models through our semantic matching methodology (Section 4.1.2).

4.3.2 Evaluation

Before applying PMIE to evaluate PMG approaches, we establish its reliability. This evaluation serves two purposes: validating that PMIE accurately extracts PI, and identifying limitations affecting downstream evaluation. We conduct this evaluation in two stages: first Actor Relation Classification, then the complete PMIE pipeline on ground truth process descriptions.

Actor Relation Classification evaluation

We validate Actor Relation Classification using 477 actor-activity relations from PET (both $\overrightarrow{\text{Actor Performer}}$ and $\overrightarrow{\text{Actor Recipient}}$ types). For each relation, we extract the sentence containing both mentions, then prompt the LLM to classify the relation type. Experiments use zero-shot, 1-shot, and 3-shot prompting with LLaMA 3.3 70B.

Table 4.3 presents F1-scores across prompting settings. Results demonstrate strong classification performance, with F1-scores ranging from 0.947 (zero-shot) to 0.972 (3-shot).

The high zero-shot performance (F1 = 0.947) indicates the distinction between performer and recipient roles is sufficiently clear and can be reliably identified by LLMs without extensive prompting. Incremental improvements with few-shot examples (1.6% gain from zero-shot to 1-shot, 0.9% gain from 1-shot to 3-shot) suggest examples help handle edge cases and ambiguous formulations.

These results validate Actor Relation Classification as highly reliable for PMIE. Near-perfect performance (F1 = 0.972) in 3-shot setting demonstrates LLMs can accurately perform this classification with appropriate task framing and minimal examples. For all subsequent PMIE evaluations, we adopt 3-shot prompting to maximize accuracy.

PMIE evaluation

Having validated Actor Relation Classification, we assess the complete PMIE pipeline to establish performance ceiling. This isolates PMIE extraction capabilities from errors introduced

Table 4.3 Actor Relation Classification results on the 477 Actor Performer and Actor Recipient relations of the PET dataset. F1-scores are reported for zero-shot, 1-shot, and 3-shot prompting settings using LLaMA 3.3 70B.

Actor Relation Classification		
Zero-shot	**1-shot**	**3-shot**
0.947	0.963	0.972

by PMG approaches, providing a baseline for interpreting downstream results.

Experimental setup. We evaluate PMIE on the 45 process descriptions from the PET dataset by treating each sentence as a simulated process model task. For example, the sentence "The data manager checks the files" is treated as a task label from which PMIE extracts mentions: checks, data manager, and files. This tests extraction methodology using ground truth descriptions while bypassing actual process model generation complexity. For each sentence, we apply PMIE to extract mentions, then group all mentions of the same description into entities. We compare results against PET annotations using semantic matching (Section 4.1).

This evaluation has inherent limitations. Since sentences lack explicit control flow structure, we cannot evaluate Sequence Flow relation extraction, which relies on PME structure. Gateway mentions (XOR Gateway , AND Gateway , Condition Specification) typically appear as structural elements rather than within task descriptions, making them impossible to extract from isolated sentences. Despite constraints, sentence-level evaluation covers the most prevalent mention types (83.8% of PET mentions) and provides insight into core PMIE capabilities.

Results. Table 4.4 presents the F1-scores for mention detection and coreference resolution tasks. We employ LLaMA 3.3 70B with 3-shot prompting for all extraction steps to ensure consistency with the Actor Relation Classification evaluation.

Mention detection performance. The three primary mention types exhibit moderate to strong performance. Activity mentions achieve highest F1-score (0.828), reflecting their central role and clear linguistic markers (action verbs). Actor mentions (F1 = 0.724) and Activity Data mentions (F1 = 0.704) show slightly lower but acceptable performance. Both types require identifying noun phrases with specific semantic roles, which can be challenging in complex sentences or when expressed implicitly.

Low performance on Further Specification mentions (F1 = 0.199) reveals a significant PMIE limitation. This type encompasses diverse linguistic phenomena—temporal expressions

Table 4.4 PMIE extraction performance on PET sentences. F1-scores are reported for mention detection and coreference resolution using semantic matching with LLaMA 3.3 70B using 3-shot. Gateway mentions cannot be extracted from isolated sentences (indicated by dash).

Task	Type	F1-score
Mention Detection	Activity	0.828
	Activity Data	0.704
	Actor	0.724
	Further Specification	0.199
	XOR Gateway	-
	AND Gateway	-
	Condition Specification	-
Coreference Resolution	Activity Data	0.717
	Actor	0.387

("after approval"), spatial specifications ("in the warehouse"), modal qualifications ("if possible")—making consistent identification difficult without extensive examples. Furthermore, Further Specification mentions are often optional contextual details that may be omitted or reformulated in process models, reducing utility for evaluation.

Coreference resolution performance. Entity construction shows mixed results. [Activity Data] entities maintain stable performance (F1 = 0.717) compared to mention detection (F1 = 0.704), indicating grouping mentions into entities does not introduce substantial errors. However, [Actor] entities show dramatic performance drop to F1 = 0.387, a 33.7-point decrease from mention-level performance.

This degradation stems from error propagation. An entity is considered correct only if all constituent mentions are correctly identified and grouped. A single missed or misclassified mention marks the entire entity as incorrect, leading to disproportionate penalty accumulation. For actor mentions, which frequently appear multiple times across documents with various referring expressions (pronouns, definite descriptions, full names), the probability of at least one mention error is high, cascading into entity-level failures.

Implications for PMIE evaluation. These results establish PMIE capabilities and limitations. The methodology demonstrates reliable extraction of core mention types (Activity , Activity Data , Actor) with F1-scores above 0.7, sufficient for meaningful PMG evaluation. However, poor performance on Further Specification mentions and severe error propagation affecting entities suggest PMIE-based evaluation should focus on mention-level metrics rather than entity or relation-level metrics. Inability to extract gateway mentions and sequence flow relations from sentences further constrains PMIE applicability to full process model

evaluation.

4.3.3 Results on PMG Approaches

Having established PMIE reliability, we apply it to assess the three selected PMG approaches on the complete PET dataset. We generate process models for all 45 PET descriptions using BPMN-chatbot, ConverMod, and ProMoAI, then extract Process Information from generated models using PMIE and compare against ground truth PET annotations using semantic matching.

Table 4.5 presents F1-scores for MD, CR and RE across approaches using LLaMA 3.3 70B in 3-shot setting.

Table 4.5 PMIE evaluation results on 3 PMG approaches using LLaMA 3.3 70b in 3-shot setting. Semantic micro F1-scores are reported for each mention, entity and relation type.

Task	PI type	BPMN-chatbot	Convermod	ProMoAI
MD	Activity	0.367	**0.392**	**0.392**
	Activity Data	**0.518**	**0.518**	0.501
	Actor	**0.647**	0.62	0.615
	Further Specification	**0.166**	0.106	0.153
	XOR Gateway	0	0	0
	AND Gateway	0	0	0
	Condition Specification	**0.082**	0	0
CR	[Activity Data]	0.118	**0.129**	0.027
	[Actor]	**0.184**	0.17	0.118
RE	Sequence Flow	0.053	**0.077**	0.054
	Uses	0.216	**0.217**	0.2
	Actor Performer	0.22	**0.355**	0.202
	Actor Recipient	0.177	**0.289**	0.23
	Same Gateway	0	0	0
	Further Specification	0.042	0.047	**0.056**

MD performance. The three primary mention types show substantial degradation compared to PMIE baseline (Table 4.4). Activity mentions, which achieved F1 = 0.828 when extracting from PET sentences, drop to F1 = 0.367–0.392 when extracting from generated models (53% decrease). Activity Data mentions decline from F1 = 0.704 to F1 = 0.501–0.518 (27% decrease), while Actor mentions drop from F1 = 0.724 to F1 = 0.615–0.647 (11–15% decrease). This suggests generated models systematically omit or deform information from original

descriptions, with activities most affected.

Gateway-related mentions (XOR Gateway , AND Gateway , Condition Specification) achieve near-zero F1-scores. This reflects representational mismatch: process models encode branching logic through structural gateway elements rather than textual mentions. An exclusive gateway in BPMN is a diamond-shaped node with multiple outgoing edges, not text containing "if" or "otherwise." PET annotations mark such textual indicators as gateway mentions, creating inherent incompatibility between annotation schema and process model representation.

CR performance. Entity construction reveals severe error propagation, with F1-scores dropping to 0.027–0.184 compared to 0.387–0.717 in PMIE baseline. Dramatic decline—particularly for Activity Data entities in ProMoAI (F1 = 0.027)—stems from compounding mention-level errors.

RE performance. Relation F1-scores range from 0.042 to 0.355, with most below 0.22. $\xrightarrow{\text{Sequence Flow}}$ relations achieve particularly low performance (F1 = 0.053–0.077), despite being directly extractable from PME through deterministic rules. This poor performance results from prerequisite dependency on accurate Activity mention detection: a $\xrightarrow{\text{Sequence Flow}}$ relation can only be extracted correctly if both connected activities are accurately identified. With activity mention F1-scores around 0.37–0.39, the probability of correctly identifying both endpoints is approximately $(0.38)^2 \approx 0.14$, aligning with observed flow relation performance.

Comparison across PMG approaches. While Table 4.5 shows marginal differences, they are insufficiently large for confident conclusions about relative performance. Absolute F1-scores remain low across all approaches and PI types, suggesting the performance ceiling is constrained not primarily by PMG approach choice.

Implications for PMIE-based evaluation. These results reveal critical PMIE limitations for evaluating PMG approaches. Although the three PMG approaches employ fundamentally different PMRs and generation strategies, producing structurally different process models, they achieve nearly identical PMIE scores. This score convergence despite model diversity suggests PMIE itself is the evaluation bottleneck rather than PMG quality. Systematic performance degradation compared to PMIE baseline, combined with near-zero scores for gateway mentions and severe error propagation affecting entities and relations, confirms that direct PI extraction from process models cannot reliably capture semantic content of original descriptions. The root cause is representational mismatch: PET annotations mark fine-grained textual spans in natural language, while process models encode information through structured graphical elements with simplified labels. This prevents meaningful comparison using PMIE alone.

These findings motivate developing an alternative evaluation strategy presented in the next

section, reorganizing PI into a representation more aligned with process model structure.

4.4 Task-based Evaluation Framework

PMIE evaluation results revealed fundamental limitations in directly extracting and comparing fine-grained PI from process models. Low F1-scores across all mention, entity, and relation types—combined with complete failure to extract gateway-related information—indicate the representational gap between textual PET annotations and structural process models is too large for direct comparison. This section addresses these limitations by developing an alternative evaluation framework reorganizing PI into task-based representations more closely aligned with how process models encode information.

We begin by identifying specific PET annotation schema limitations contributing to representational mismatch, then present TaskPET, a new dataset version structuring PI around process tasks rather than isolated mentions. This task-centric representation enables more robust evaluation through two complementary methodologies: coarse-grained task-level matching assessing overall task coverage, and fine-grained attribute-level evaluation identifying specific information omissions within matched tasks.

4.4.1 PET Annotation Schema Limitations

PMIE evaluation exposed structural limitations in PET annotation schema hindering application to process model evaluation. These fall into three categories: representational mismatches, annotation constraints, and semantic inconsistencies.

Mention-based relations without entity-level grouping. PET annotations define relations between individual mention spans rather than entities. If an actor is mentioned three times using different expressions (e.g., "the manager", "she", "the department head"), each activity relation must be annotated separately for each mention. This creates redundancy and complicates comparison with process models, where an actor typically appears once in a task label. To evaluate process models, we must first map mentions to entities, then aggregate relations—introducing an error-prone preprocessing step not originally required by PET schema.

Non-overlapping mention constraint. PET annotation guidelines prohibit overlapping mention spans, meaning a single text span can only be assigned one mention type. This creates problems when Further Specification or Condition Specification mentions contain embedded Activity , Activity Data , or Actor mentions. For instance, "after the manager approves the invoice" might be annotated as a single Further Specification mention, losing information that

it contains an Activity (approves), an Actor (manager), and Activity Data (invoice). Process models typically decompose such compound expressions into separate tasks, making direct comparison impossible without extracting embedded information.

Missing process boundary annotations. PET does not explicitly annotate process start and end points. While process models typically include explicit start and end events, textual descriptions may use implicit cues (e.g., "The process begins when...") or omit boundary information entirely. This prevents systematic evaluation of whether generated models correctly identify process entry and exit points.

Semantic inconsistencies in condition representation. PET schema defines both Condition Specification mentions (textual indicators like "if" or "otherwise") and $\overrightarrow{\text{Sequence Flow}}$ relations between activities. However, guidelines replace condition-based branching with $\overrightarrow{\text{Sequence Flow}}$ relations, creating inconsistencies. For instance, "If approved, proceed to payment" should theoretically involve a Condition Specification mention and a conditional flow relation, but is annotated only with a $\overrightarrow{\text{Sequence Flow}}$ relation. This complicates evaluation of branching logic in generated models.

These limitations collectively prevent PET schema from serving as ideal reference for process model evaluation. The schema was designed for PIE from natural language, not comparison with structured process models. Addressing these limitations requires reorganizing PI into a representation better reflecting process model structure while preserving semantic content of original annotations.

4.4.2 TaskPET Construction

To address the limitations identified above, we construct TaskPET, a new version of the PET dataset that structures PI around process tasks rather than isolated mentions. The core principle of TaskPET is to group related mentions into task representations that correspond more directly to the task elements found in process models, thereby reducing the representational mismatch observed in PMIE evaluation.

Task definition and structure

A PI Task in TaskPET represents a single process activity along with its associated participants, data objects, and contextual qualifications. Formally, we define a PI Task as a tuple:

$$\text{PI Task} = (\text{Activity}, D, P, R, F) \tag{4.4}$$

where:

- Activity is a single Activity mention (mandatory)
- $D = \{d_1, d_2, \ldots, d_w\}$ is a set of [Activity Data] entities $(0 \leq w \leq 4)$
- $P = \{p_1, p_2, \ldots, p_x\}$ is a set of performer [Actor] entities $(0 \leq x \leq 4)$
- $R = \{r_1, r_2, \ldots, r_y\}$ is a set of recipient [Actor] entities $(0 \leq y \leq 4)$
- $F = \{f_1, f_2, \ldots, f_z\}$ is a set of Further Specification mentions $(0 \leq z \leq 4)$

This structure mirrors the typical composition of process model tasks, which consist of an action (activity) performed by an actor (performer), potentially involving data objects and recipients. The inclusion of further specifications accommodates contextual information that may appear in task labels.

We refer to the five components as **task attributes**. Each task must contain exactly one activity, but all other attributes are optional. The upper bounds $(w, x, y, z \leq 4)$ reflect practical constraints observed in the PET dataset, where tasks never involve more than four entities of any single type.

Task construction methodology. We construct PI Tasks from PET annotations through the following procedure, illustrated with the example sentence: "The manager reviews the application and forwards it to the HR department."

1. **Activity-centric grouping**: Each activity mention in PET becomes the nucleus of a task. We identify all relations involving that activity mention. In this example, activities reviews and forwards yield two tasks.
2. **Attribute aggregation**: For each activity, we collect all Activity Data mentions connected via Uses relations, all Actor mentions connected via Actor Performer and Actor Recipient relations, and all Further Specification mentions connected via Further Specification relations. For reviews, we collect manager (performer) and application. For forwards, we collect manager (performer), it, and HR department (recipient).
3. **Entity-level resolution**: Since PET relations reference mention spans, we first resolve coreferences to identify unique entities, then assign entities (not mentions) to task attributes. This addresses the mention-versus-entity limitation identified in the previous subsection. In this example, coreference resolution links it to application, yielding entity [application, it]. The final task for forwards thus references this entity rather than the pronoun mention.

This construction process transforms the 501 activity entities in the PET dataset into 501 PI Tasks, maintaining a one-to-one correspondence that preserves the semantic content while restructuring it into a task-centric format.

Dataset statistics. Table 4.6 presents summary statistics for TaskPET, showing task and attribute distribution across 45 process descriptions.

The dataset contains 501 tasks across 45 descriptions, averaging 11.13 tasks per description. With 1.2 tasks per sentence on average, most sentences describe a single task, though some contain multiple activities. The attribute distribution shows Activity Data entities are most common (0.93 per task), followed by Actor performers (0.62 per task), recipients (0.33 per task), and Further Specification (0.13 per task). This reflects typical process description structure, which prioritizes specifying what is done (activity) and what is manipulated (data), while sometimes omitting who performs the action or contextual qualifications.

Task verbalization

To enable semantic matching between ground truth tasks and tasks extracted from generated process models, we convert the structured task representation into natural language text. This verbalization transforms the formal task tuple into a human-readable phrase that can be compared using the similarity metrics established in Section 4.1.

Entity verbalization. Before verbalizing complete tasks, we determine a canonical textual representation for each entity, as entities consist of multiple coreferent mentions (e.g., "the manager", "she", "the department head"). We adopt a longest common subsequence approach: we filter out all pronouns from the mention set, then identify the longest contiguous word sequence that appears in any two mentions of the entity. If no common sequence exists (i.e., all mentions are unique), we use the first mention. This balances specificity with generality—longer common sequences capture core semantic content while avoiding overly specific details.

Analysis of PET reveals 72.3% of entities have all mentions with identical text, making verbalization straightforward. For remaining entities, the longest common subsequence typically extracts the head noun phrase, removing articles, modifiers, and pronouns. Examples from TaskPET are presented in Table 4.7.

Task verbalization. We verbalize complete tasks using a template-based approach that concatenates attributes in a fixed order:

`Actor Performer` + `Activity` + `Activity Data` + `Actor Recipient` + `Further Specification`

Table 4.6 TaskPET dataset statistics showing task distribution and average attribute counts per task.

Metric	Value
Task count	501
Task per description	11.13
Task per sentence	1.20
Entity per task	2.01
Activity Data	0.93
Actor Performer	0.62
Actor Recipient	0.33
Further Specification	0.13

Table 4.7 Entity verbalization examples.

Entity	Verbalization
[“the invoice with payment advice”, “the invoice of the MPOO”]	“the invoice”
[“a new order”, “the order”]	“order”
[“his data”, “The customer data”, “it”]	“data”
[“an accounting employee”, “she” (4×), “the accounting employee” (4×)]	“accounting employee”

Optional attributes are omitted if empty. This format mirrors natural language descriptions while maintaining consistent structure for comparison. Table 4.8 presents examples from TaskPET.

Table 4.8 Task verbalization examples showing attributes and resulting representations.

Task verbalization examples
the Support Officer produces minutes
the accounting employee forwards the copy of the invoice the commercial manager searched The Police Report
the grid operator transmits the meter data the customer service, the old supplier

These verbalizations may lack grammatical fluency (e.g., missing prepositions), but semantic similarity matching captures equivalence regardless.

These verbalized tasks serve as the basis for both task-level and attribute-level evaluation methodologies described in the following subsections.

4.4.3 Task-level Evaluation

Having established TaskPET and task verbalization, we now evaluate how well generated process models capture tasks described in the original texts. This evaluation provides a coarse-grained assessment of task coverage, identifying which tasks are correctly represented, which are missing, and which are spuriously added by PMG approaches.

Methodology. Ground truth TaskPET tasks are verbalized using the methodology described in the previous subsection, while generated task labels are used as-is. We then apply the semantic matching methodology established in Section 4.1 to match ground truth and generated tasks.

This evaluation setting offers an advantage over PMIE: since we use generated task labels directly without extraction, we eliminate dependency on an additional extraction methodology that would require validation. This makes task-level evaluation more reliable, as evaluation errors cannot stem from extraction failures. An overview of the task-level evaluation process is presented in Figure 4.6.

Results. Table 4.9 presents the task-level evaluation results for the three PMG approaches on the complete TaskPET dataset.

Table 4.9 Task-level evaluation results on TaskPET for three PMG approaches. Support refers to ground truth tasks (501), Predicted refers to tasks generated by each approach, Matched refers to predicted tasks that successfully matched ground truth tasks.

	BPMN-chatbot	**Convermod**	**ProMoAI**
Support		501	
Predicted	450	416	**486**
Matched	304	261	**293**
Precision	**0.69**	0.64	0.61
Recall	**0.61**	0.52	0.58
F1-score	**0.64**	0.57	0.6

The results demonstrate improvement over PMIE-based evaluation, with F1-scores ranging from 0.57 to 0.64 compared to 0.05–0.35 for individual mention types in Table 4.5. This improvement validates our hypothesis that task-level comparison reduces representational mismatch between textual descriptions and process models.

Task coverage analysis. All three approaches generate between 416 and 486 tasks (83%–97% of ground truth task count), indicating generated models generally contain a slightly lower number of activities than original descriptions, validating findings from the previous chapter.

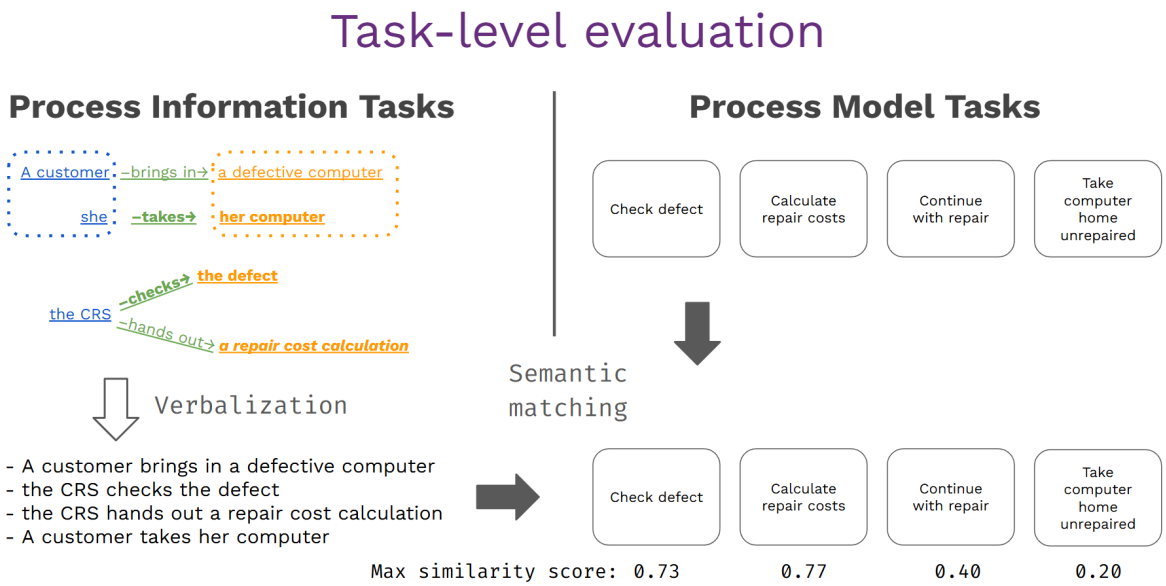


Figure 4.6 Overview of the task-level evaluation process.

The matched task counts (261–304) reveal approximately 52%–61% of ground truth tasks are successfully matched. Precision scores of 0.61–0.69 indicate 31%–39% of generated tasks do not match ground truth tasks, representing either reformulations too distant from the original or spurious additions. Recall scores of 0.52–0.61 show 39%–48% of ground truth tasks are missing from generated models, suggesting systematic information loss during PMG.

BPMN-chatbot achieves the highest precision (0.69) and recall (0.61), yielding F1 = 0.64. This may stem from its constrained generation mechanism coupled with a well-designed PMR (see previous chapter). ConverMod achieves the lowest performance (F1 = 0.57) with the fewest generated tasks (416) and lowest matched count (261), suggesting its reformulations are less aligned with ground truth or that it systematically omits certain task types. ProMoAI generates the most tasks (486), closest to ground truth count, achieving moderate performance (F1 = 0.60) with 293 matched tasks.

Limitations of task-level evaluation. While task-level matching provides improved performance over PMIE and enables comparative assessment of PMG approaches, it remains a coarse-grained evaluation treating tasks as atomic units. A task may be counted as correctly matched even if it contains attribute-level errors—for example, "manager reviews application" would match "employee reviews application" if semantic similarity exceeds the threshold, despite actor mismatch. Conversely, a task may fail to match despite correctly capturing most attributes if a single critical component differs substantially. This motivates development of fine-grained attribute-level evaluation presented in the next subsection.

4.4.4 Task Attribute Evaluation

Task-level evaluation provides insights into overall task coverage but obscures details about which specific information is correctly captured or omitted within matched tasks. A task may successfully match at the holistic level while still containing errors in individual attributes—missing actor information, incorrect data objects, or omitted contextual specifications. This subsection addresses this limitation by developing a fine-grained evaluation methodology assessing each task attribute independently.

The attribute-level approach enables precise diagnostic analysis of PMG performance. Rather than merely identifying a missing task, we can determine whether the issue stems from an incorrect activity, missing data object, or omitted performer. This granularity is essential for understanding strengths and weaknesses of different PMG approaches and for guiding future improvements. An overview of the task-level evaluation process is presented in Figure 4.7.

Task Attribute Extraction

To evaluate attributes individually, we must first extract them from the verbalized task strings generated by the PMG approaches. This requires a new extraction methodology analogous to Mention Detection in PIE, which we term Task Attribute Extraction (TAE).

Extraction methodology. Given a verbalized task string (e.g., "the manager reviews the application"), TAE identifies and extracts text spans corresponding to each of five task attributes: Activity , Activity Data , Actor Performer , Actor Recipient , and Further Specification . We employ LLM-based extraction using few-shot prompting, providing the model with examples demonstrating the extraction task. The prompt instructs the model to output a JSON structure containing extracted attributes, with empty lists for attributes not present in the task.

TAE differs from PMIE's mention detection in two ways. First, TAE operates on simplified task strings rather than full natural language sentences, reducing linguistic complexity. Second, TAE extracts entities rather than mentions—since generated task labels are self-contained and do not reference other entities, there are no coreference issues to resolve. These simplifications make TAE more reliable than PMIE while maintaining necessary granularity for attribute-level evaluation.

We implement TAE using LLaMA 3.3 70B in 3-shot setting, consistent with other LLM-based extraction tasks in this chapter. The prompt template is provided in Appendix D.

Evaluation scope. A critical consideration is that attribute-level evaluation is performed at the entity level rather than at the task level. We count the total number of entities of each

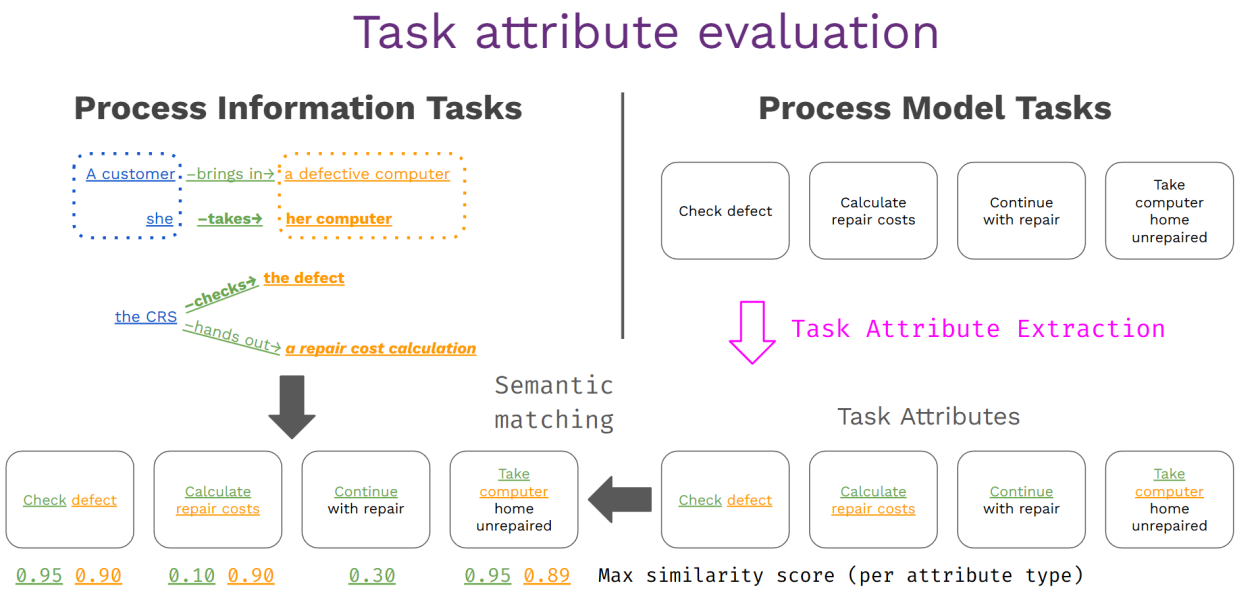


Figure 4.7 Overview of the attribute-level evaluation process.

type across all ground truth tasks (e.g., all [Activity Data] entities, all [Actor] performers) and compute precision, recall, and F1-scores based on these entity-level counts. Attributes from unmatched generated tasks (false positives in task-level evaluation) cannot be evaluated because there is no ground truth to compare against. Attributes from missing ground truth tasks are counted as false negatives in attribute-level evaluation, contributing to reduced recall.

This scoping means attribute-level metrics reflect entity coverage across the entire dataset. The recall scores thus combine two sources of error—missing tasks and missing attributes within matched tasks—while precision scores reflect only attribute-level errors within matched tasks.

Results

Table 4.10 presents the attribute-level evaluation results for the three PMG approaches on TaskPET. For each attribute type, we report the total number of ground truth entities (Support), the number of predicted entities extracted by TAE (Predicted), and Precision, Recall, and F1-score computed using semantic matching at the entity level.

The results reveal how different attribute types are captured by PMG approaches, with performance varying across attributes.

Activity extraction. Activity attributes show moderate performance (F1 = 0.33–0.35)

Table 4.10 Task attribute evaluation results on TaskPET. Task Attribute Extraction is performed using LLaMA 3.3 70B in 3-shot setting. Semantic micro precision, recall and F1-scores are reported for each task attribute. Bold indicates highest value.

		BPMN-chatbot	**Convermod**	**ProMoAI**
Task	Support		501	
	Predicted	450	416	**486**
	Matched	**304**	261	293
Activity	Support		501	
	Predicted	**302**	260	290
	Precision	0.44	**0.5**	0.47
	Recall	**0.27**	0.26	**0.27**
	F1-score	0.33	0.34	**0.35**
Activity Data	Support		468	
	Predicted	**291**	240	282
	Precision	**0.69**	0.67	0.63
	Recall	**0.43**	0.34	0.38
	F1-score	**0.53**	0.45	0.47
Actor Performer	Support		313	
	Predicted	114	106	**117**
	Precision	0.93	**0.97**	0.94
	Recall	0.34	0.33	**0.35**
	F1-score	0.5	0.49	**0.51**
Actor Recipient	Support		163	
	Predicted	115	79	**117**
	Precision	0.68	**0.73**	**0.73**
	Recall	0.48	0.36	**0.52**
	F1-score	0.56	0.48	**0.6**
Further Specification	Support		64	
	Predicted	**72**	51	78
	Precision	**0.25**	0.18	0.15
	Recall	**0.28**	0.14	0.19
	F1-score	**0.26**	0.16	0.17

given that every task must contain an activity by definition. The modest recall (0.26–0.27) reflects both missing tasks and reformulation issues. PMG approaches extract 260–302 activity attributes (52%–60% of ground truth), suggesting many task labels use formulations too distant from ground truth to match semantically. Precision ranges from 0.44 to 0.50, indicating approximately half of extracted activities fail to match ground truth, likely due to substantial reformulations.

Activity Data extraction. Activity Data attributes achieve moderate performance (F1 = 0.45–0.53), with precision (0.63–0.69) higher than recall (0.34–0.43). The moderate precision indicates that when PMG approaches include data objects in task labels, they tend to use terminology reasonably consistent with original descriptions. However, low recall reveals systematic omission: only 240–291 data attributes are extracted (51%–62% of the 468–470 ground truth data entities), indicating nearly half of process models omit data objects from task labels.

Actor Performer extraction. Actor Performer attributes demonstrate the highest precision among all attribute types (0.93–0.97) combined with low recall (0.33–0.35), yielding F1 = 0.49–0.51. The near-perfect precision suggests generated models rarely misidentify performers when they include them. However, only 106–117 performers are extracted from 313–315 ground truth instances (34%–37%), indicating approximately two-thirds of performer information is omitted. This pattern suggests PMG approaches tend to generate role-agnostic task labels, deferring actor assignment to BPMN lanes or pools rather than encoding it in task labels. For instance, given the description "The manager reviews the application", a PMG approach might generate a task labeled "Review application" and place it in a "Manager" lane, rather than labeling the task "Manager reviews application". However, since the evaluated approaches do not generate lane or pool information, this omission results in lost performer information in the generated process model.

Actor Recipient extraction. Actor Recipient attributes achieve the strongest overall performance (F1 = 0.48–0.60) among all evaluated attributes, particularly in BPMN-chatbot and ProMoAI. Precision remains moderate to high (0.68–0.73), while recall (0.36–0.52) shows variation across approaches. BPMN-chatbot and ProMoAI capture approximately half of recipient information (48%–52%), while ConverMod captures only 36%. The relatively better recall for recipients compared to performers may reflect that recipients are more integral to certain activities (e.g., "send document to customer") and thus more likely to be retained in task labels.

Further Specification extraction. Further Specification attributes exhibit the poorest performance across all metrics (F1 = 0.16–0.26), with particularly low precision (0.15–0.25).

This reflects two issues: first, contextual qualifications are rarely included in process model task labels, which prioritize core actions; second, when such specifications do appear, they are often reformulated substantially, reducing semantic similarity below the matching threshold. The very low ground truth count (63–64 instances) also limits statistical significance of these results.

Comparative analysis. BPMN-chatbot achieves the highest F1-scores for Activity Data (0.53) and Actor Recipient (0.56), and Further Specification (0.26). ConverMod consistently achieves lower scores, particularly for Activity Data (0.45) and Actor Recipient (0.48), suggesting more aggressive information reduction in task labels. ProMoAI shows strong performance, achieving the highest Activity F1-score (0.35), Actor Recipient F1-score (0.60), and Actor Performer F1-score (0.51), demonstrating balanced attribute preservation.

Evaluation methodology interpretation. The attribute-level evaluation quantifies the total amount of missing information across the entire dataset, not just within matched tasks. By counting all ground truth entities and comparing them against all extracted entities (from matched tasks only), we capture both task-level failures (missing tasks contribute missing attributes) and attribute-level failures (matched tasks with missing or incorrect attributes). This provides a comprehensive view of information loss during PMG, showing substantial information is systematically omitted even when tasks are successfully generated.

4.5 Discussion

This chapter presented an evaluation of PMG approaches grounded in PI. We developed two methodologies: PMIE, which extracts PI from generated models, and a task-based framework that reorganizes PI into representations aligned with process model structure. Our evaluation of three PMG approaches revealed substantial information loss during generation, with F1-scores ranging from 0.33 to 0.60. This section reflects on key findings, discusses limitations, and outlines future research.

4.5.1 Main Findings

Our evaluation revealed patterns in how PMG approaches preserve and transform PI during generation. These findings emerge from task-level matching and attribute-level analysis, providing complementary perspectives on information loss.

Systematic information loss and evaluation bias in current literature. All three PMG approaches exhibit substantial information loss. Task-level evaluation shows only 52%–61% of ground truth tasks are matched, indicating 39%–48% of activities are omitted or reformulated

beyond recognition. Attribute-level evaluation reveals similar losses, with 50% of attributes like Activity Data and Actor Performer missing. This loss is consistent across approaches (F1 = 0.57–0.64), suggesting it stems from fundamental PMG constraints and LLM biases rather than implementation choices. These findings reveal a discrepancy with existing literature, where studies report high ratings through evaluations providing activity labels or accepting substantial reformulations. This suggests bias in current literature, where LLM capabilities are praised while information loss remains unacknowledged. Our results highlight the need for evaluation methodologies grounded in source descriptions rather than arbitrary ground truth or human judgment, enabling objective assessment of information preservation.

Complementary insights from coarse-grained and fine-grained evaluation. Task-level and attribute-level evaluation provide complementary views. Task-level evaluation treats activities as atomic units (F1 = 0.57–0.64), while attribute-level evaluation decomposes tasks and reveals lower performance (F1 = 0.33–0.60). This gap indicates holistic task matching can succeed even when attributes are incorrect or omitted. For instance, a task may match based on Activity and Activity Data similarity while omitting Actor Performer information. Attribute-level analysis exposes losses that task-level evaluation would obscure, demonstrating the value of multi-granularity evaluation.

Differential attribute preservation patterns. Attribute-level evaluation reveals differences in how information types are preserved. Activity attributes (F1 = 0.33–0.35) show moderate performance despite being mandatory, reflecting substantial reformulation. Activity Data attributes (F1 = 0.45–0.53) achieve moderate performance with recall (0.34–0.43) lower than precision (0.63–0.69), indicating systematic omission in half of tasks. The most dramatic pattern appears in actor information: Actor Performer attributes exhibit near-perfect precision (0.93–0.97) but low recall (0.33–0.35), meaning performers are rarely included but accurate when present. Actor Recipient attributes achieve more balanced performance (F1 = 0.48–0.60) with higher recall. Further Specification attributes show poorest performance (F1 = 0.16–0.26), confirming contextual qualifications are rarely preserved.

Comparative performance across PMG approaches. While all approaches exhibit similar patterns, differences emerge. BPMN-chatbot achieves strongest task-level performance (F1 = 0.64), validating the superiority of branching PMRs for PMG. ConverMod achieves lower scores (task-level F1 = 0.57), indicating more aggressive information reduction due to less advanced generation and prompting. ProMoAI generates the most tasks (486 vs. 501 ground truth), suggesting their compact PMR mitigates omission. However, differences remain modest, with all approaches operating within similar ranges, reinforcing that information loss is a systemic challenge rather than approach-specific limitation.

Methodological implications. The gap between PMIE-based evaluation (F1 = 0.05–0.35) and task-based evaluation (F1 = 0.57–0.64) underscores the importance of methodology choice. PMIE's poor performance stems from representational mismatch between fine-grained textual annotations and process models, while task-based evaluation's improved performance reflects better alignment. This validates our hypothesis that evaluation frameworks must accommodate process model characteristics rather than directly applying text-based metrics. Future evaluation should prioritize methodologies aligning with process model structure while maintaining rigor through validated semantic matching.

4.5.2 Limitations

Several limitations constrain generalizability and completeness of our findings.

Dependence on PET dataset. Our evaluation relies exclusively on PET for ground truth PI. While PET is the most comprehensive manually annotated corpus for process information extraction, it remains the only dataset with sufficiently fine-grained annotations. Alternative datasets such as DECON [97] and ATDP [98] lack detailed mention-level and relation-level annotations. This limits our ability to assess how results generalize across domains, description styles, or annotation schemes. Furthermore, since PET descriptions originate from the Friedrich dataset [33], there is potential for data contamination if evaluated LLMs were exposed to these texts during training.

Limited Task Attribute Extraction validation. While we conducted qualitative evaluation of TAE to assess reliability, more extensive validation may be required. The assessment examined extraction outputs on task samples but systematic quantitative evaluation remains constrained by absence of ground truth attribute annotations for generated tasks. Errors in TAE would propagate to attribute-level results, potentially distorting our understanding of which attributes PMG approaches capture. More comprehensive validation, including larger-scale manual annotation, would strengthen confidence in attribute-level findings.

Task-level semantic matching configuration. Based on experimental findings, we used `sts-mpnet` with $\theta = 0.7$ for task-level evaluation rather than `potion-base-8M` with $\theta = 0.86$ used for mention matching. This reflects structural differences between short mention spans and longer task descriptions, which exhibit different similarity distributions. While informed by empirical experimentation, task-level threshold selection did not undergo the same systematic validation as mention-level matching. Absence of comprehensive threshold validation introduces uncertainty about whether the configuration optimally balances precision and recall.

Limited scope of evaluated approaches. Our evaluation focuses on three approaches selected based on availability, reproducibility, and PMR diversity. While these represent different paradigms and remain actively used, newer approaches increasingly generate additional elements beyond core activities, gateways, and flows. Recent approaches support swimlanes, pools, and organizational constructs that encode actor information structurally rather than in task labels [2, 60, 95, 96]. Our framework does not assess these elements, potentially affecting validity when applied to recent approaches. Extending evaluation to capture swimlane and pool information would be necessary to fairly assess newer approaches and provide a complete picture of actor-related PI preservation.

Absence of structural evaluation. Our methodology focuses on task-based content assessment and does not account for structural aspects. While we evaluate which tasks are present and what information they contain, we do not assess task sequence correctness, control flow logic capture, or overall process structure matching. A generated model could achieve high scores while exhibiting structural deficiencies such as incorrect ordering, missing parallelism, or erroneous branching. Evaluating gateways based on the PET annotations would be a first step in this direction, Combining our semantic matching with structural metrics such as GED or behavioral similarity (e.g., alignment-based conformance checking) would enable comprehensive evaluation capturing both content preservation and structural correctness. Such hybrid frameworks would provide a complete picture of PMG performance.

4.5.3 Future Research

The evaluation methodology and findings open several directions for future research in PMG evaluation and improvement.

Expansion to recent PMG approaches. PMG continues evolving, with new approaches incorporating advanced prompting, retrieval-augmented generation, and fine-tuned models. Extending our framework to assess emerging approaches would provide insights into whether recent advances address identified information loss patterns. Particular attention should be given to approaches explicitly leveraging PI or structured process knowledge, as these may demonstrate improved performance on our PI-based metrics.

Comparative evaluation metric validation. Our methodology employs semantic matching based on embedding similarity, motivated by its ability to handle reformulations while maintaining rigor. However, alternative paradigms exist, including strict matching, graph-based structural comparison, and human judgment. Systematic comparison would clarify how methodology influences apparent performance and help establish consensus on evaluation standards. Such validation would identify scenarios where different metrics capture

complementary aspects of generation quality.

Dataset expansion and diversification. Reliance on a single dataset motivates expanding ground truth resources. Future work should annotate additional process descriptions with fine-grained PI, prioritizing domains and styles not well-represented. Multilingual annotations would extend capabilities beyond English, while real-world organizational documentation would complement academic texts. Expanded datasets would enable robust assessment of PMG generalization and identification of domain-specific patterns.

Evaluation with proprietary LLMs. Our extraction pipeline relies on LLaMA 3.3 70B for reproducibility and accessibility. However, proprietary models such as GPT-5 or Claude typically achieve higher performance on complex NLP tasks. Future work should compare extraction performance across model families to quantify how LLM choice affects evaluation results.

CHAPTER 5 CONCLUSION

This thesis investigated fundamental challenges in Process Modeling with LLMs, focusing on PMR comparison and evaluation methodology development. Through systematic empirical studies, we addressed gaps in standardization, transparency, and assessment practices that have hindered progress in automated Process Modeling . This chapter synthesizes the main contributions, acknowledges limitations, and outlines future research directions.

5.1 Summary of Works

The research was motivated by three limitations: the absence of standardized evaluation frameworks, fragmentation across heterogeneous approaches and PMRs, and the shortage of quality annotated data. We developed solutions that reduce evaluative subjectivity, establish reproducible comparison protocols, and provide evidence-based guidance for Process Modeling methodologies.

Research Question 1 examined which features of existing PMRs most influence their effectiveness for Process Modeling and PMG. Through comparison of nine PMRs using the Process Modeling Dataset, we established six key requirements: token compactness, expressivity, human readability, visualization capabilities, usability, and extensibility. The evaluation revealed substantial variation in token efficiency, with branching PMRs (BPMN text, JSON branches) achieving the greatest compactness but limited expressivity. Graph-based PMRs demonstrated superior expressiveness and better supported the full range of BPMN elements, with BPMN process and PME emerging as well-balanced choices. The extrinsic evaluation through PMG confirmed that PMR choice significantly impacts generation quality, with branching representations generally outperforming graph-based approaches. We observed a consistent simplification bias across all PMRs, where LLMs systematically generated models with fewer elements and reduced complexity compared to ground truth. These findings provide guidance for selecting PMRs for specific Process Modeling tasks.

Research Question 2 investigated whether integrating PI into the evaluation pipeline could yield reliable, fine-grained, and model-agnostic assessment of PMG outputs. We developed two complementary evaluation methodologies based on process descriptions annotated with PI. The first approach, PMIE, attempted direct extraction of mentions, entities, and relations from generated process models for comparison with ground truth PI. This methodology revealed fundamental representational mismatches between fine-grained textual annotations

and structured process models, achieving F1-scores of only 0.05 to 0.35 for mention types. The second approach, task-based evaluation using TaskPET, reorganized PI into representations aligned with process model structure. This methodology demonstrated improved performance (F1 = 0.57–0.64 for task-level matching) and enabled systematic comparative assessment of PMG approaches. Through validation experiments establishing semantic similarity thresholds and matching procedures, we demonstrated that our task-based evaluation framework provides reliable, fine-grained assessment complementing existing structural metrics.

The evaluation of three representative PMG approaches (BPMN Chatbot, Convermod, and ProMoAI) using our methodology revealed substantial information loss during process model generation. Task-level F1-scores ranged from 0.57 to 0.64 across approaches, indicating even the best-performing systems capture only approximately 60% of the tasks in source descriptions. Attribute-level analysis identified systematic patterns in this information loss, with actor information (F1 = 0.45–0.53) and further specification attributes (F1 = 0.16–0.26) particularly challenging for all approaches. These findings establish empirical baselines for PMG performance and identify specific dimensions requiring improvement.

Beyond these primary contributions, this research produced several methodological advances. The development and validation of semantic similarity-based matching for process mentions addresses a fundamental challenge in process model comparison, enabling evaluation despite lexical variation in LLM outputs. The Process Modeling Dataset provides a curated resource spanning diverse process domains and element types, supporting future comparative studies. The task-centric reorganization of PI demonstrated in TaskPET offers a template for adapting fine-grained annotations to process model evaluation.

5.2 Limitations

While this thesis makes significant contributions to Process Modeling evaluation and PMR comparison, several limitations warrant acknowledgment. Chapter-specific limitations have been discussed in their respective sections; here we address overarching constraints.

Limited BPMN element coverage. Our analysis focused on a core subset of BPMN elements commonly found in textual process descriptions: tasks, events, gateways and sequence flows. More specialized elements such as swimlanes, data objects, textual annotations, boundary events, and timer events received limited attention due to their sparse representation in available datasets. This constraint reflects the current state of Process Modeling datasets and limitations of our evaluation methodology. Recent PMG approaches have begun addressing these additional elements, and our evaluation framework should accommodate them as

appropriately annotated data becomes available. The extensibility requirement identified in our PMR comparison anticipated this need for broader element coverage.

Dataset constraints. Our research design was constrained by two complementary but incomplete datasets. The PMR comparison in Chapter 3 uses the Process Modeling Dataset containing description-model pairs but lacking PI annotations, preventing application of our PI-based evaluation methodology to assess PMRs directly. Conversely, the evaluation methodology in Chapter 4 relies on PET providing rich PI annotations but lacking corresponding ground truth models, precluding traditional model-to-model comparison. This limitation is dataset-based: our evaluation methodology could have been applied to PMR comparison if the Process Modeling Dataset had been annotated with PI, enabling unified assessment across both studies. The data fragmentation necessitated separate evaluation protocols rather than a unified framework. Moreover, both datasets remain relatively small compared to resources available for general NLP tasks, potentially limiting generalizability.

LLM diversity. Our empirical evaluations predominantly employed Llama 3.3 70b for PIE and Gemini 2.0 Flash for PMG across all approaches to ensure fair comparison. While this choice controls for model-specific effects, it constrains our ability to generalize findings across available LLMs. Different model families and sizes exhibit distinct capabilities in structured output generation, instruction following, and domain-specific reasoning. The relative performance of PMRs and evaluation methodologies might vary across models. Future work should evaluate these approaches across multiple LLMs to establish robustness.

Single-shot generation paradigm. Our research framed Process Modeling as a single-shot generation task, where the LLM produces a complete process model from a textual description in one step. This framing may impose constraints that do not reflect how human process modeling occurs. Human modelers typically engage in iterative refinement through stakeholder consultations. They clarify ambiguities through dialogue, request missing information, and incrementally improve models based on feedback. By evaluating only single-shot generation, we may underestimate the potential of LLM-based approaches using conversational interaction, iterative refinement, or multi-step generation. While our evaluation methodology could support assessment of such interactive approaches by measuring information capture across dialogue turns, the current datasets and experimental protocols do not include conversational context or incremental model versions necessary for such analysis.

5.3 Future Research

The contributions and limitations of this work suggest several directions for future research in Process Modeling with LLMs.

Addressing data scarcity. The shortage of quality annotated data represents the most critical bottleneck for advancing Process Modeling research. Two complementary strategies could alleviate this constraint. First, synthetic pair generation using Model2Text transformation techniques [99] could produce a large-scale dataset of description-model pairs, though careful validation would be required to ensure linguistic naturalness and diversity. Second, semi-automated annotation using tools such as TeaPIE [46] could enrich existing process model collections with fine-grained PI annotations. Combining these approaches could yield a dataset containing process descriptions, ground truth models, and detailed PI annotations, enabling unified evaluation across multiple dimensions. Prioritizing diverse domains, multilingual content, and real-world organizational documentation would enhance ecological validity.

Validation against human judgment. While our evaluation methodology demonstrates internal consistency and enables fine-grained assessment, its correlation with human quality judgments remains unestablished. Future work should collect expert ratings of generated process models across multiple quality dimensions (content completeness, structural correctness, understandability, fitness for purpose) and analyze correlations with our task-based metric. Such validation would identify which metrics best predict human-perceived quality and clarify whether different evaluation approaches measure distinct quality dimensions or provide redundant information.

Improving PMG approaches. Our evaluation results identified systematic patterns in information loss. These findings suggest targeted improvement opportunities. Approaches incorporating explicit actor tracking mechanisms, using coreference resolution for actor mentions might achieve more complete information capture. Similarly, our PMR comparison revealed clear advantages in some representations. Crafting a custom PMR and PMG approach based on these findings should enable improved performance.

These research directions aim toward more capable, reliable, and practically deployable Process Modeling systems. By addressing current limitations while building on the evaluation infrastructure and empirical insights established in this thesis, future work can advance Process Modeling toward robust tools supporting real-world BPM practices.

REFERENCES

[1] H. Kourani, A. Berti, D. Schuster, and W. M. P. van der Aalst, "ProMoAI: Process Modeling with Generative AI," Mar. 2024.

[2] J. T. Licardo, N. Tankovic, and D. Etinger, "BPMN Assistant: An LLM-Based Approach to Business Process Modeling," Sep. 2025.

[3] J. Carmona, B. van Dongen, and M. Weidlich, "Conformance Checking: Foundations, Milestones and Challenges," in *Process Mining Handbook*, ser. Lecture Notes in Business Information Processing, W. M. P. van der Aalst and J. Carmona, Eds. Cham: Springer International Publishing, 2022, pp. 155–190.

[4] M. Weske, G. Decker, M. Dumas, M. La Rosa, J. Mendling, and H. A. Reijers, "Model Collection of the Business Process Management Academic Initiative," Apr. 2020.

[5] P. Bellan, H. van der Aa, M. Dragoni, C. Ghidini, and S. P. Ponzetto, "PET: An Annotated Dataset for Process Extraction from Natural Language Text Tasks," in *Business Process Management Workshops*, C. Cabanillas, N. F. Garmann-Johnsen, and A. Koschmider, Eds. Cham: Springer International Publishing, 2023, pp. 315–321.

[6] L. Lin, Y. Jin, Y. Zhou, W. Chen, and C. Qian, "MAO: A Framework for Process Model Generation with Multi-Agent Orchestration," Aug. 2024.

[7] N. Klievtsova, J.-V. Benzin, T. Kampik, J. Mangler, and S. Rinderle-Ma, "Conversational Process Modeling: Can Generative AI Empower Domain Experts in Creating and Redesigning Process Models?" Jan. 2024.

[8] M. Voelter, R. Hadian, T. Kampik, M. Breitmayer, and M. Reichert, "Leveraging Generative AI for Extracting Process Models from Multimodal Documents," Jun. 2024.

[9] J. Kopke and A. Safan, "Efficient LLM-Based Conversational Process Modeling," in *BPM2024*, Sep. 2024.

[10] H. Kourani, A. Berti, D. Schuster, and W. M. P. van der Aalst, "Evaluating large language models on business process modeling: Framework, benchmark, and self-improvement analysis," *Software and Systems Modeling*, Sep. 2025.

[11] A. Wimmer, A. Costa, and L. Pufahl, "Natural Language Processing for BPMN Model Generation with LLMs: A Systematic Literature Review," in *BPM2025*. Seville: Springer, Sep. 2025.

[12] M. Forell and S. Schüler, "Modeling meets Large Language Models," in *Modellierung 2024 Satellite Events*. Gesellschaft für Informatik e.V., 2024, p. 10.18420/modellierung2024.

[13] W. Van Woensel and S. Motie, "NLP4PBM: A systematic review on process extraction using natural language processing with rule-based, machine and deep learning methods," *Enterprise Information Systems*, vol. 18, no. 11, p. 2417404, Nov. 2024.

[14] M. Dumas, M. La Rosa, J. Mendling, and H. A. Reijers, *Fundamentals of Business Process Management*. Berlin, Heidelberg: Springer, 2018.

[15] A. Costa, A. Wimmer, and L. Pufahl, "LLM4BPMN: A Prompting Approach with LLMs for BPMN Model Generation," in *EDOC*, Lisboa, Sep. 2025.

[16] T. Kampik, C. Warmuth, A. Rebmann, R. Agam, L. N. P. Egger, A. Gerber, J. Hoffart, J. Kolk, P. Herzig, G. Decker, H. van der Aa, A. Polyvyanyy, S. Rinderle-Ma, I. Weber, and M. Weidlich, "Large Process Models: Business Process Management in the Age of Generative AI," Sep. 2023.

[17] M. Vidgof, S. Bachhofner, and J. Mendling, "Large Language Models for Business Process Management: Opportunities and Challenges," Apr. 2023.

[18] M. Dumas, F. Fournier, L. Limonad, A. Marrella, M. Montali, J.-R. Rehse, R. Accorsi, D. Calvanese, G. De Giacomo, D. Fahland, A. Gal, M. La Rosa, H. Völzer, and I. Weber, "AI-Augmented Business Process Management Systems: A Research Manifesto," *ACM Transactions on Management Information Systems*, vol. 14, no. 1, pp. 1–19, Mar. 2023.

[19] K. Busch, A. Rochlitzer, D. Sola, and H. Leopold, "Just Tell Me: Prompt Engineering in Business Process Management," in *Enterprise, Business-Process and Information Systems Modeling*, ser. Lecture Notes in Business Information Processing, H. van der Aa, D. Bork, H. A. Proper, and R. Schmidt, Eds. Cham: Springer Nature Switzerland, 2023, pp. 3–11.

[20] Y. Rizk, P. Venkateswaran, V. Isahagian, A. Narcomey, and V. Muthusamy, "A Case for Business Process-Specific Foundation Models," in *Business Process Management Workshops*, ser. Lecture Notes in Business Information Processing, J. De Weerdt and L. Pufahl, Eds. Cham: Springer Nature Switzerland, 2024, pp. 44–56.

[21] W. X. Zhao, K. Zhou, J. Li, T. Tang, X. Wang, Y. Hou, Y. Min, B. Zhang, J. Zhang, Z. Dong, Y. Du, C. Yang, Y. Chen, Z. Chen, J. Jiang, R. Ren, Y. Li, X. Tang, Z. Liu, P. Liu, J.-Y. Nie, and J.-R. Wen, "A Survey of Large Language Models," Nov. 2023.

[22] V. Farkas, "Towards a Machine Learning-Based Approach for Recommending Next Elements in BPMN Models," in *2nd Workshop on Natural Language Processing for Business Process Management*, 2023.

[23] A. Beheshti, J. Yang, Q. Z. Sheng, B. Benatallah, F. Casati, S. Dustdar, H. R. M. Nezhad, X. Zhang, and S. Xue, "ProcessGPT: Transforming Business Process Management with Generative Artificial Intelligence," May 2023.

[24] A. Rebmann, F. D. Schmidt, G. Glavaš, and H. van der Aa, "Evaluating the Ability of LLMs to Solve Semantics-Aware Process Mining Tasks," Jul. 2024.

[25] M. Käppel, L. Ackermann, S. Jablonski, and S. Härtl, "Explaining transformer-based next activity prediction by using attention scores," *Process Science*, vol. 2, no. 1, p. 11, Jun. 2025.

[26] F. Fournier, L. Limonad, and I. Skarbovsky, "Towards a Benchmark for Causal Business Process Reasoning with LLMs," Jul. 2024.

[27] H. Kourani, A. Berti, J. Hennrich, W. Kratsch, R. Weidlich, C.-Y. Li, A. Arslan, D. Schuster, and W. M. P. van der Aalst, "Leveraging Large Language Models for Enhanced Process Model Comprehension," Sep. 2024.

[28] D. Fahland, F. Fournier, L. Limonad, I. Skarbovsky, and A. J. E. Swevels, "How well can large language models explain business processes?" Jan. 2024.

[29] K. Busch and H. Leopold, "Towards a Benchmark for Large Language Models for Business Process Management Tasks," Oct. 2024.

[30] H.-A. López-Acosta, B. Feng, J. Lindner, M. Franceschetti, and A. Abbad-Andaloussi, "Ambiguity Detection in Business Process Descriptions: An Evidence and an Automated Approach," *Proceedings of the 23rd International Conference on Business Process Management (BPM 2025)*, 2025.

[31] A. Berti, H. Kourani, A. Antonov, and W. M. P. V. D. Aalst, "Knowledge-Driven Hallucination in Large Language Models: An Empirical Study on Process Modeling," Sep. 2025.

[32] P. Fettke and C. Houy, "Evaluating the Process Modeling Abilities of Large Language Models – Preliminary Foundations and Results," Mar. 2025.

[33] F. Friedrich, "Automated generation of business process models from natural language input," Ph.D. dissertation, School of Business and Economics. Humboldt-Universität zu Berlin, 2010.

[34] H. van der Aa, H. Leopold, and H. A. Reijers, "Detecting Inconsistencies Between Process Models and Textual Descriptions," in *Business Process Management*, H. R. Motahari-Nezhad, J. Recker, and M. Weidlich, Eds. Cham: Springer International Publishing, 2015, pp. 90–105.

[35] P. Bellan, M. Dragoni, C. Ghidini, H. van der Aa, and S. P. Ponzetto, "Process Extraction from Text: Benchmarking the State of the Art and Paving the Way for Future Challenges," Oct. 2023.

[36] A. Berti, H. Kourani, H. Hafke, C.-Y. Li, and D. Schuster, "Evaluating Large Language Models in Process Mining: Capabilities, Benchmarks, and Evaluation Strategies," Apr. 2024.

[37] S. Schüler and S. Alpers, "State of the Art: Automatic Generation of Business Process Models," in *Business Process Management Workshops*, J. De Weerdt and L. Pufahl, Eds. Cham: Springer Nature Switzerland, 2024, pp. 161–173.

[38] P. Bellan, M. Dragoni, C. Ghidini, S. Ponzetto, and H. van der Aa, "Process Extraction from Natural Language Text: The PET Dataset and Annotation Guidelines," in *NL4AI 2022: Sixth Workshop on Natural Language for Artificial Intelligence*, vol. 3287. CEUR, 2022, pp. 177–191.

[39] J. Neuberger, L. Ackermann, and S. Jablonski, "Beyond Rule-Based Named Entity Recognition and Relation Extraction for Process Model Generation from Natural Language Text," in *Cooperative Information Systems*, M. Sellami, M.-E. Vidal, B. van Dongen, W. Gaaloul, and H. Panetto, Eds. Cham: Springer Nature Switzerland, 2023, pp. 179–197.

[40] 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," Jul. 2024.

[41] J. Neuberger, H. van der Aa, L. Ackermann, D. Buschek, J. Herrmann, and S. Jablonski, "Assisted Data Annotation for Business Process Information Extraction from Textual Documents," Oct. 2024.

[42] P. Bellan, M. Dragoni, and C. Ghidini, "Leveraging pre-trained language models for conversational information seeking from text," Mar. 2022.

[43] ——, "Assisted Process Knowledge Graph Building Using Pre-trained Language Models," in *AIxIA 2022 – Advances in Artificial Intelligence*, A. Dovier, A. Montanari, and A. Orlandini, Eds. Cham: Springer International Publishing, 2023, pp. 60–74.

[44] ——, "Process Knowledge Extraction and Knowledge Graph Construction Through Prompting: A Quantitative Analysis," in *Proceedings of the 39th ACM/SIGAPP Symposium on Applied Computing*, ser. SAC '24. New York, NY, USA: Association for Computing Machinery, May 2024, pp. 1634–1641.

[45] M. Eberts and A. Ulges, "An End-to-end Model for Entity-level Relation Extraction using Multi-instance Learning," Dec. 2021.

[46] J. Neuberger, J. Herrmann, M. Käppel, and S. Jablonski, "TeaPie: A Tool for Efficient Annotation of Process Information Extraction Data," *CoopIs 2024*, 2024.

[47] J. T. Licardo, N. Tanković, and D. Etinger, "A Method for Extracting BPMN Models from Textual Descriptions Using Natural Language Processing," *Procedia Computer Science*, vol. 239, pp. 483–490, Jan. 2023.

[48] M. Grohs, L. Abb, N. Elsayed, and J.-R. Rehse, "Large Language Models can accomplish Business Process Management Tasks," Jul. 2023.

[49] N. Klievtsova, J.-V. Benzin, T. Kampik, J. Mangler, and S. Rinderle-Ma, "Conversational Process Modelling: State of the Art, Applications, and Implications in Practice," in *Business Process Management Forum*, C. Di Francescomarino, A. Burattin, C. Janiesch, and S. Sadiq, Eds. Cham: Springer Nature Switzerland, 2023, pp. 319–336.

[50] Y. Fontenla-Seco, S. M. Winkler, A. Gianola, M. Montali, M. Lama, and A. Bugarín-Diz, "The Droid You're Looking For: C-4PM, a Conversational Agent for Declarative Process Mining," in *Proceedings of the Best Dissertation Award, Doctoral Consortium, and Demonstration and Resources Forum at BPM 2023*, vol. 3469. CEUR, Sep. 2023, pp. 112–116.

[51] J. Mangler and N. Klievtsova, "Textual Process Descriptions and Corresponding BPMN Models," Mar. 2023.

[52] N. Klievtsova, J.-V. Benzin, J. Mangler, T. Kampik, and S. Rinderle-Ma, "Process Modeler vs. Chatbot: Is Generative AI Taking over Process Modeling?" in *Process Mining Workshops*, A. Delgado and T. Slaats, Eds. Cham: Springer Nature Switzerland, 2025, pp. 637–649.

[53] H. Kourani, A. Berti, D. Schuster, and W. M. P. van der Aalst, "Process Modeling With Large Language Models," Apr. 2024.

[54] K. Apaydin and Y. Zisgen, "Local Large Language Models for Business Process Modeling," in *Process Mining Workshops*, A. Delgado and T. Slaats, Eds. Cham: Springer Nature Switzerland, 2025, pp. 605–609.

[55] Camunda, "BPMN for research dataset," Dec. 2015.

[56] S. Ivanov, A. Kalenkova, and W. Aalst, van der, "BPMNDiffViz : A tool for BPMN models comparison," in *Proceedings of the Demo Session of the 13th International Conference on Business Process Management (BPM 2015, Innsbruck, Austria, August 31-September 3, 2015)*, ser. CEUR Workshop Proceedings, F. Daniel and S. Zugal, Eds., 2015, pp. 35–39.

[57] A. Berti, S. van Zelst, and D. Schuster, "PM4Py: A process mining library for Python," *Software Impacts*, vol. 17, p. 100556, Sep. 2023.

[58] X. Li, L. Ni, R. Li, J. Liu, and M. Zhang, "MaD: A Dataset for Interview-based BPM in Business Process Management," in *2023 International Joint Conference on Neural Networks (IJCNN)*, Jun. 2023, pp. 1–8.

[59] R. Sonbol, G. Rebdawi, and N. Ghneim, "A Machine Translation Like Approach to Generate Business Process Model from Textual Description," *SN Computer Science*, vol. 4, no. 3, p. 291, Mar. 2023.

[60] A. Nour Eldin, N. Assy, O. Anesini, B. Dalmas, and W. Gaaloul, "Nala2BPMN: Automating BPMN Model Generation with Large Language Models," in *Cooperative Information Systems*, M. Comuzzi, D. Grigori, M. Sellami, and Z. Zhou, Eds. Cham: Springer Nature Switzerland, 2025, pp. 398–404.

[61] Q. Nivon and G. Salaün, "Automated Generation of BPMN Processes from Textual Requirements," in *Service-Oriented Computing*, W. Gaaloul, M. Sheng, Q. Yu, and S. Yangui, Eds. Singapore: Springer Nature, 2025, pp. 185–201.

[62] R. Dijkman, M. Dumas, B. van Dongen, R. Käärik, and J. Mendling, "Similarity of business process models: Metrics and evaluation," *Information Systems*, vol. 36, no. 2, pp. 498–516, Apr. 2011.

[63] A. Schoknecht, T. Thaler, P. Fettke, A. Oberweis, and R. Laue, "Similarity of Business Process Models—A State-of-the-Art Analysis," *ACM Comput. Surv.*, vol. 50, no. 4, pp. 52:1–52:33, Aug. 2017.

[64] S. Sholiq, R. Sarno, and E. S. Astuti, "Generating BPMN diagram from textual requirements," *Journal of King Saud University - Computer and Information Sciences*, vol. 34, no. 10, Part B, pp. 10 079–10 093, Nov. 2022.

[65] M. Voelter, "Generative AI for Business Process Management - Suitability of Modalities," Ph.D. dissertation, Ulm University, Ulm, 2024.

[66] J. G. Colonna, A. A. Fares, M. Duarte, and R. Sousa, "Process mining embeddings: Learning vector representations for Petri nets," *Intelligent Systems with Applications*, vol. 23, p. 200423, Sep. 2024.

[67] A. Berti, X. Wang, H. Kourani, and W. van der Aalst, "Specializing Large Language Models for Process Modeling via Reinforcement Learning with Verifiable and Universal Rewards," Sep. 2025.

[68] S. SAP, "SAP-samples/llm-round-trip-correctness," SAP Samples, Aug. 2025.

[69] Z. Yan, R. Dijkman, and P. Grefen, "Fast business process similarity search," *Distributed and Parallel Databases*, vol. 30, no. 2, pp. 105–144, Apr. 2012.

[70] N. Klievtsova, J. Mangler, T. Kampik, J.-V. Benzin, and S. Rinderle-Ma, "How Can Generative AI Empower Domain Experts in Creating Process Models?" in *Wirtschaftsinformatik 2024 Proceedings*, 2024.

[71] A. Brissard, F. Cuppens, and A. Zouaq, "What is the Best Process Model Representation? A Comparative Analysis for Process Modeling with Large Language Models," Jul. 2025.

[72] D. Xu, W. Chen, W. Peng, C. Zhang, T. Xu, X. Zhao, X. Wu, Y. Zheng, Y. Wang, and E. Chen, "Large Language Models for Generative Information Extraction: A Survey," Jun. 2024.

[73] R. Han, T. Peng, C. Yang, B. Wang, L. Liu, and X. Wan, "Is Information Extraction Solved by ChatGPT? An Analysis of Performance, Evaluation Criteria, Robustness and Errors," May 2023.

[74] Y. Naraki, R. Yamaki, Y. Ikeda, T. Horie, and H. Naganuma, "Augmenting NER Datasets with LLMs: Towards Automated and Refined Annotation," Mar. 2024.

[75] C. K. R. Evuru, S. Ghosh, S. Kumar, R. S, U. Tyagi, and D. Manocha, "CoDa: Constrained Generation based Data Augmentation for Low-Resource NLP," Mar. 2024.

[76] M. Josifoski, M. Sakota, M. Peyrard, and R. West, "Exploiting Asymmetry for Synthetic Training Data Generation: SynthIE and the Case of Information Extraction," Oct. 2023.

[77] J. Lu, Z. Yang, Y. Wang, X. Liu, B. Mac Namee, and C. Huang, "PaDeLLM-NER: Parallel Decoding in Large Language Models for Named Entity Recognition," Feb. 2024.

[78] T. Xie, Q. Li, Y. Zhang, Z. Liu, and H. Wang, "Self-Improving for Zero-Shot Named Entity Recognition with Large Language Models," Mar. 2024.

[79] G. Martinelli, E. Barba, and R. Navigli, "Maverick: Efficient and Accurate Coreference Resolution Defying Recent Trends," Jul. 2024.

[80] J. Munoz-Gama, R. R. de la Fuente, M. M. Sepúlveda, and R. R. Fuentes, "Conformance Checking Challenge 2019 (CCC19)," Feb. 2019.

[81] A. Ivanchikj, S. Serbout, and C. Pautasso, "Live process modeling with the BPMN Sketch Miner," *Software and Systems Modeling*, vol. 21, no. 5, pp. 1877–1906, Oct. 2022.

[82] Y. Heng, C. Deng, Y. Li, Y. Yu, Y. Li, R. Zhang, and C. Zhang, "ProgGen: Generating Named Entity Recognition Datasets Step-by-step with Self-Reflexive Large Language Models," Mar. 2024.

[83] W. Zhou, S. Zhang, Y. Gu, M. Chen, and H. Poon, "UniversalNER: Targeted Distillation from Large Language Models for Open Named Entity Recognition," Jan. 2024.

[84] K. Papineni, S. Roukos, T. Ward, and W.-J. Zhu, "Bleu: A Method for Automatic Evaluation of Machine Translation," in *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics*, P. Isabelle, E. Charniak, and D. Lin, Eds. Philadelphia, Pennsylvania, USA: Association for Computational Linguistics, Jul. 2002, pp. 311–318.

[85] C.-Y. Lin, "ROUGE: A Package for Automatic Evaluation of Summaries," in *Text Summarization Branches Out*. Barcelona, Spain: Association for Computational Linguistics, Jul. 2004, pp. 74–81.

[86] S. Banerjee and A. Lavie, "METEOR: An Automatic Metric for MT Evaluation with Improved Correlation with Human Judgments," in *Proceedings of the ACL Workshop on Intrinsic and Extrinsic Evaluation Measures for Machine Translation and/or Summarization*, J. Goldstein, A. Lavie, C.-Y. Lin, and C. Voss, Eds. Ann Arbor, Michigan: Association for Computational Linguistics, Jun. 2005, pp. 65–72.

[87] M. Popović, "chrF: Character n-gram F-score for automatic MT evaluation," in *Proceedings of the Tenth Workshop on Statistical Machine Translation*, O. Bojar, R. Chatterjee, C. Federmann, B. Haddow, C. Hokamp, M. Huck, V. Logacheva, and P. Pecina, Eds. Lisbon, Portugal: Association for Computational Linguistics, Sep. 2015, pp. 392–395.

[88] T. Zhang, V. Kishore, F. Wu, K. Q. Weinberger, and Y. Artzi, "BERTScore: Evaluating Text Generation with BERT," Feb. 2020.

[89] N. Reimers and I. Gurevych, "Sentence-BERT: Sentence Embeddings using Siamese BERT-Networks," Aug. 2019.

[90] S. Xiao, Z. Liu, P. Zhang, N. Muennighoff, D. Lian, and J.-Y. Nie, "C-Pack: Packed Resources For General Chinese Embeddings," Sep. 2024.

[91] L. Chen, G. Varoquaux, and F. M. Suchanek, "Learning High-Quality and General-Purpose Phrase Representations," Feb. 2024.

[92] S. Tulken and T. van Dongen, "Model2Vec: Turn any sentence transformer into a small fast model," 2024.

[93] Y. Liu, M. Ott, N. Goyal, J. Du, M. Joshi, D. Chen, O. Levy, M. Lewis, L. Zettlemoyer, and V. Stoyanov, "RoBERTa: A Robustly Optimized BERT Pretraining Approach," Jul. 2019.

[94] D. Cer, M. Diab, E. Agirre, I. Lopez-Gazpio, and L. Specia, "SemEval-2017 Task 1: Semantic Textual Similarity - Multilingual and Cross-lingual Focused Evaluation," in *Proceedings of the 11th International Workshop on Semantic Evaluation (SemEval-2017)*, 2017, pp. 1–14.

[95] A. Safan and J. Köpke, "BPMN-Chatbot++: LLM-Based Modeling of Collaboration Diagrams with Data," in *BPM 2025 Best Dissertation Award, Doctoral Consortium, and Demonstration & Resources Forum*, Sep. 2025.

[96] N. Klievtsova, M. Ehrendorfer, J. Mangler, and S. Rinderle-Ma, "AutoBPMN.AI: Conversational Process Modeling and Automation," in *BPM 2025 Best Dissertation Award, Doctoral Consortium, and Demonstration & Resources Forum*, Sep. 2025.

[97] H. van der Aa, C. Di Ciccio, H. Leopold, and H. A. Reijers, "Extracting Declarative Process Models from Natural Language," in *Advanced Information Systems Engineering*, ser. Lecture Notes in Computer Science, P. Giorgini and B. Weber, Eds. Cham: Springer International Publishing, 2019, pp. 365–382.

[98] L. Quishpi, J. Carmona, and L. Padró, "Extracting Annotations from Textual Descriptions of Processes," in *Business Process Management*, D. Fahland, C. Ghidini, J. Becker, and M. Dumas, Eds. Cham: Springer International Publishing, 2020, pp. 184–201.

[99] N. Klievtsova, J. Mangler, T. Kampik, and S. Rinderle-Ma, "Utilizing Process Models in the Requirements Engineering Process Through Model2Text Transformation," in *2024 IEEE 32nd International Requirements Engineering Conference (RE)*, Jun. 2024, pp. 205–217.

APPENDIX A BPMN ELEMENTS SUMMARY

Table A.1 BPMN Elements Summary - Flow Objects

BPMN Element	**Definition**	**Main subtypes**	**Subtype definition**	**Visual**
Task	An atomic activity within a process flow. It represents a single piece of work that cannot be broken down to a finer level of detail in the process model.			
Gateway	A decision point that controls the divergence (splitting) and convergence (merging) of sequence flows.	Exclusive gateway	An exclusive (XOR) gateway is a decision point where only one path can be taken out of two or more alternatives.	
		Parallel gateway	A parallel (AND) gateway is a synchronization point where multiple concurrent paths of execution are created.	

Table A.2 BPMN Elements Summary - Connecting Objects and Swimlanes

BPMN Element	**Definition**	**Main subtypes**	**Subtype definition**	**Visual**
Event	Something that happens during the course of a process. Events affect the flow of the process and usually have a cause (trigger) or an impact (result).	Start event	Indicates where a process begins.	
		Intermediate event	Occurs between start and end events during process execution.	
		End event	Indicates where a process terminates.	
Sequence flow	An arrow that shows the order in which activities (tasks, events, gateways) will be performed. It connects the flow elements in a process.			
Swimlane	Containers used to organize and categorize activities based on who is responsible for performing them.	Pool	A separate process boundary that represents an independent participant (like a company, customer, or system).	
		Lane	A section within a Pool that organizes activities according to the specific role responsible for those tasks.	
Message flow	A dashed arrow used to show the flow of messages (information) between two separate Pools (participants).			

APPENDIX B PET STATISTICS

Tables B.1, B.2 and B.3 present additional PET statistics for each type of PI.

Table B.1 PIE mentions and their statistics in the PET dataset.

Mention type	PET dataset statistics [38]					
	Absolute count	Relative count	Per description	Per sentence	Average length	Standard dev. (lenght)
Activity	501	30.16%	11.13	1.2	1.1	0.48
Activity Data	452	27.21%	10.04	1.08	3.49	2.47
Actor	439	26.43%	9.76	1.05	2.32	1.11
Further Specification	64	3.86%	1.42	0.15	5.19	3.4
XOR Gateway	117	7.04%	2.6	0.28	1.26	0.77
AND Gateway	8	0.48%	0.18	0.02	2.12	1.54
Condition Specification	80	4.82%	1.78	0.19	6.04	3.04

Table B.2 PIE entities and their statistics in the PET dataset.

Entity type	PET dataset statistics [39]		
	Absolute count	Intra-entity max token distance (avg.)	Type-token ratio (lexical diversity) (avg.)
[Activity Data]	75	31.93	0.48
[Actor]	88	54.84	0.60

Table B.3 PIE relations and their statistics in the PET dataset.

Relation type	Source mention	Target mention	PET dataset statistics [38]			
			Absolute count	Relative count	Per description	Per sentence
Sequence Flow	Activity or XOR Gateway or AND Gateway or Condition Specification	Activity or XOR Gateway or AND Gateway or Condition Specification	689	39.37%	15.31	1.65
Uses	Activity	Activity Data	477	27.26%	10.6	1.14
Actor Performer	Activity	Actor	313	17.89%	6.96	0.75
Actor Recipient	Activity	Actor	164	9.37%	3.64	0.39
Further Specification	Activity	Further Specification	64	3.66%	1.42	0.15
Same Gateway	XOR Gateway AND Gateway	XOR Gateway AND Gateway	43	2.46%	0.96	0.1

APPENDIX C PET DATASET SAMPLES

PET document example

The process starts when a customer submits a claim by sending in relevant documentation.
The Notification department at the car insurer checks the documents upon completeness and registers the claim.
Then, the Handling department picks up the claim and checks the insurance.
Then, an assessment is performed.
If the assessment is positive, a garage is phoned to authorise the repairs and the payment is scheduled (in this order).
Otherwise, the claim is rejected.
In any case (whether the outcome is positive or negative), a letter is sent to the customer and the process is considered to be complete.

PET sentence example

As a basic principle, ACME AG receives invoices on paper or fax.

Figure C.1 PET samples. Mentions are annotated as follows : *Activity*, *Activity Data*, *Actor*, *Further Specification*, *XOR Gateway*, *Condition Specification*.

APPENDIX D PROMPTS

Mention Detection zero-shot prompt for activity mentions

```
# Task

You are a business process modelling expert, tasked with identifying mentions of
    process relevant activities (tasks, assignments) in textual descriptions of
    business processes. You can find a definition below.

# Definition

**Activity**: Focus on verbs and nouns, that describe an active task or action
    executed during the business process. Never contains the Actor that executes it,
    nor the Activity Data that's used during this Activity, is just the verb or noun
    as in "checked" and not "is checked"!

## Restrictions

Rule 1. Always include the determiner, if it is present.

Rule 2. Don't extract auxiliary verbs, such as be, is, are, will etc.

Rule 3. Don't extract Activities, if they describe something that is not being done (
    negations!).

Rule 4: Don't extract Activities, that are not process relevant, i.e., they describe a
     state someone or something is in (e.g. waiting).

Rule 5: Don't extract an Activity, if they describe the process lifecycle (e.g.
    process end, start, etc.)

# Format

Provide the output in JSON. Each mention should be a JSON object with a "text" field
    containing the mention text.

## Format Example

The first step is to complete the report .
Then the clerk will review it .

{
    "mentions": [
        {"text": "complete"},
        {"text": "review"}
    ]
}

# Notes

Do not change the text you extract, i.e., do not correct typos, or change spaces, or
    add punctuation.
Do not use any code formatting.
```

Figure D.1 Mention Detection zero-shot prompt for *Activity* mentions. Used in PMIE.

Coreference Resolution zero-shot prompt

```
# Task

You are a business process modelling expert, tasked with
    identifying which mentions of certain process relevant elements
    in textual descriptions of business processes belong to the same
     entity. These mentions are spans of text and are marked in the
    text with xml-style tags containing their id.
A mention refers to the same real world entity, e.g. via anaphoric
    (e.g. pronouns) or co-referent mentions (abbreviations, same
    text, etc.), or if it simply has very similar surface-form (
    similar text).

# Format

Provide the output in JSON format. Each entity should be a list of
    mention ids.

## Format Example

Given the following input:

The LLM extracts <m2>multiple mentions</m2> in <m3>the text</m3> .
    It outputs <m6>the mentions</m6> in json . <m7>They</m7> are
    hidden inside <m8>sentences</m8>.

You will extract these entities:
{
    "entities": [
        ["m2", "m6", "m7"]
    ]
}


# Restrictions

Only list entities that have at least two mentions!
```

Figure D.2 Coreference Resolution zero-shot prompt. Used in PMIE to create [Activity Data] and [Actor] entities.

Actor Relation Classification zero-shot prompt

```
You are an expert annotator in process modeling. Your task is to
   determine the type of the relation between an activity and an
   actor in a sentence.
The relation can either be "actor performer" (the actor is
   performing the activity) or "actor recipient" (the actor is
   passively affected by the activity).

Provide your answer in json using the following template.
```

Figure D.3 Actor Relation Classification zero-shot prompt. Used in PMIE.

Task Attribute Extraction zero-shot prompt

```
You are an expert annotator in process modeling. You need to
   extract the exact words from the following task description
   corresponding to:
- the activity
- the associated activity data
- the actor performing the action
- the actor receiving the action
- the further specification of the activity

If any information is not available, mark it as "N/A". Each word
   can only have one label.

Provide your answers in json using the following template.
```

Figure D.4 Task Attribute Extraction zero-shot prompt.