



universität  
**uulm**

**Faculty of Engineering,  
Computer Science and Psychology**  
Institute for Databases  
and Information Systems

# **Generative AI for Business Process Management - Suitability of Modalities**

Master thesis at Ulm University

**Submitted by:**

Marvin Völter

marvin.voelter@uni-ulm.de

1032090

**Reviewer:**

Prof. Dr. Manfred Reichert

Prof. Dr. Rüdiger Pryss

**Supervisor:**

Marius Breitmayer

Dr. Timotheus Kampik

Dr. Raheleh Hadian

2024

Version March 21, 2024

© 2024 Marvin Völter

Set: PDF-L<sup>A</sup>T<sub>E</sub>X 2<sub>ε</sub>

## **Abstract**

Business Process Management (BPM) is a thriving discipline that helps organizations run better. Previous work suggested using Large Language Models (LLMs) to significantly reduce effort in the discovery process of BPM by automatically generating process models from existing documentation. The documentation is often available in a multimodal form containing images and texts. This thesis presents an investigation of the capabilities of Generative Pre-trained Transformers (GPTs) to auto-generate graphical process models from multi-modal (i.e., text- and image-based) inputs. More precisely, it first introduces a small dataset for setting an objective common ground. Then, it applies multimodal GPT capabilities to generate process models using zero-, one- and few-shot prompting strategies. Moreover, it introduces an evaluation framework for the carried-out generation performance. The results indicate that GPTs can potentially be useful tools for semi-automated process modeling based on multi-modal inputs. More importantly, however, the dataset, evaluation metrics, and open-source evaluation code provide a structured framework for continued systematic evaluations moving forward.

## **Acknowledgements**

I am thankful for the great cooperation between Ulm University and SAP Signavio. I would like to thank my university supervisors, Prof. Dr. Manfred Reichert and Marius Breitmayer, for their amazing support. I would also like to thank my SAP Signavio supervisors, Dr. Timotheus Kampik and Dr. Raheleh Hadian, for their incredible mentorship. It was a pleasure working with you all!

Thanks should also go to Christian Warmuth, Nataliia Klietsova, Lukas Loos, Jan Hoffbauer, Lorenz Ohly, Bruno Zirnststein, and the whole SAP Signavio Team for sharing their knowledge in enjoyable and enriching discussions.

I am also grateful for the unwavering support and balance my parents, my girlfriend, and my friends provided, enriching my life beyond academia.

# Contents

<b>1</b>	<b>Introduction</b>	<b>1</b>
1.1	Motivation . . . . .	1
1.2	Goals and Contributions . . . . .	2
1.3	Structure . . . . .	2
<b>2</b>	<b>Fundamentals</b>	<b>3</b>
2.1	Business Process Management . . . . .	3
2.1.1	BPM Lifecycle . . . . .	4
2.1.2	Business Process Modeling with BPMN . . . . .	6
2.1.3	Advanced BPMN concepts . . . . .	8
2.1.4	Most used BPMN elements . . . . .	11
2.2	Generative AI . . . . .	12
2.2.1	Context . . . . .	13
2.2.2	Deep Neural Networks . . . . .	14
2.2.3	Convolutional Neural Networks . . . . .	16
2.2.4	Recurrent Neural Networks . . . . .	16
2.2.5	Traditional Generative Models - VAEs and GANs . . . . .	17
2.2.6	Modern Generative Models - Transformers . . . . .	18
2.2.7	Large Language Models . . . . .	22
2.2.8	ChatGPT . . . . .	23
2.2.9	Multimodality . . . . .	24
<b>3</b>	<b>Related work</b>	<b>26</b>
<b>4</b>	<b>Contribution Overview</b>	<b>29</b>
<b>5</b>	<b>Contribution I: Data Set</b>	<b>30</b>
5.1	Requirements . . . . .	30

5.2	Existing data sets . . . . .	31
5.3	Creation of the data set . . . . .	32
5.4	Characteristics of the data set . . . . .	34
5.5	Evaluation of the data set . . . . .	43
<b>6</b>	<b>Contribution II: Model Generation with Gen AI</b>	<b>44</b>
6.1	Current multimodal models . . . . .	44
6.2	Current approaches . . . . .	45
6.3	Conceptual design . . . . .	47
6.4	Execution . . . . .	51
<b>7</b>	<b>Contribution III: Evaluation</b>	<b>53</b>
7.1	Requirements . . . . .	53
7.2	Existing evaluation methods . . . . .	54
7.3	Introducing the evaluation framework . . . . .	56
7.3.1	Element based breakdown . . . . .	56
7.3.2	Similarity score calculation . . . . .	61
7.3.3	Evaluation of the evaluation framework . . . . .	66
7.4	Results . . . . .	67
<b>8</b>	<b>Discussion</b>	<b>71</b>
8.1	Derived implications . . . . .	71
8.2	Limitations of the thesis . . . . .	72
<b>9</b>	<b>Conclusion</b>	<b>74</b>
9.1	Summary . . . . .	74
9.2	Outlook . . . . .	75
<b>A</b>	<b>Appendix</b>	<b>76</b>
	<b>Bibliography</b>	<b>79</b>
	<b>Glossary</b>	<b>89</b>

# 1 Introduction

## 1.1 Motivation

Recent developments in the field of generative AI and Large Language Models (LLMs), such as ChatGPT, have attracted a lot of attention [35, p. 319]. Van Dis et al. are calling the technical ability a "Game Changer for Science" [71, p. 224]. Therefore, it seems promising to tackle current challenges with these new developments while remaining objective and realistic to evaluate the technical suitability for the specific use cases scientifically.

The Business Process Management (BPM) domain is a thriving discipline that helps organizations run better [48, p. 1]. The market was valued at around 9 billion U.S. dollars in 2020, growing to 14 billion by 2025 [60]. Kampik et al. speculate that LLMs can significantly reduce effort and lower knowledge barriers in the BPM Domain [33]. Vidgof et al. [62] suggest using LLMs to significantly reduce effort in the discovery process, one of the key areas of the BPM domain. LLMs could be used to produce process models automatically from documentation, offering a similar value proposition to Process Mining [62, p. 7].

While the idea of using LLMs to produce process models from documentation seems promising, there is currently only limited research assessing the feasibility and suitability of the proposal. Therefore, this thesis evaluates the suitability of generative AI methods for producing process models automatically from documentation. More specifically, the thesis focuses on leveraging multimodal documentation incorporating text and images because documentation frequently exists in this format. Yet, there has been no existing research exploring this multimodal area. The thesis aims to close this gap.

## 1.2 Goals and Contributions

The thesis aims to evaluate the feasibility and suitability of generative AI methods for generating process models from multimodal documents containing images and text.

A key challenge of the thesis is the currently extremely fast-moving and evolving environment. Novel and better-performing LLMs are published frequently. For example, the two promising models, GPT-4V (used in ChatGPT) and Gemini 1.5 Pro, were published during the thesis, as outlined in Chapter 6.1. Therefore, this thesis not only aims to evaluate the current state-of-the-art performance but also to offer a framework for future evaluations in this fast-changing environment.

To achieve these two goals, the thesis provides three main contributions:

1. Representative dataset: This contribution establishes the foundation for objectively measuring current and future feasibility performance.
2. Model Generation with Gen AI: The thesis examines current state-of-the-art performance by applying the most promising current model and approaches.
3. Evaluation: The thesis introduces an evaluation framework to measure the carried-out generation performance and to enable objective future evaluations. It executes the evaluation and presents the results.

Contributions 1 and 2 emphasize adherence to scientific best practices, whereas Contribution 3 strategically accepts the risk of using closed systems to attain the best state-of-the-art performance. It relies heavily on GPT-4V, a model developed by OpenAI. While it is a commercial black box with little insight, it currently offers the best performance regarding benchmarks, as detailed in Chapter 6.1.

## 1.3 Structure

This work is divided into 9 chapters. After the introduction, the fundamentals of the Business Process Management and Generative AI domain are explained in Chapter 2. Chapter 3 presents related work. Afterward, Chapter 4 provides an overview of the three main contributions. Chapters 5 - 7 contain the main contributions. Finally, derived implications and limitations are discussed in Chapter 8, and final considerations are made in Chapter 9.



## 2 Fundamentals

This chapter establishes the theoretical underpinnings crucial for a comprehensive thesis understanding. In detail, it introduces the research areas of the Business Process Management (BPM) domain and Generative AI fundamentals.

### 2.1 Business Process Management

“Business Process Management (BPM) is the art and science of overseeing how work is performed in an organization to ensure consistent outcomes and to take advantage of improvement opportunities.” [18, p. 1]

These typical improvement opportunities are concerned with reducing costs, reducing execution time, reducing error rates, as well as producing new innovations [18, p. 1]. As the name suggests, this is achieved by managing an organization’s business processes. Business processes include all tasks (including their dependencies) required to deliver a service or product. Typically the following components of a business process are sufficient to manage the business processes of an organization: Activities, events, decision points, actors, objects, and outcomes [18, pp. 1-6]. Table 2.1 describes the components exemplary for a car manufacturing process:

Process Component	Description	Example
Activities	Single units of work that someone is doing actively	Assembling the car, Testing the assembled car
Events	Things that happen passively and can trigger activities	Customer order arrived, Raw parts arrived
Continued on next page		

Process Component	Description	Example
Decision Points	When a decision occurs on which activity to perform next	Decision if test successful
Actors	Someone who performs the activities	Assembly line worker, Quality Tester
Objects	When an activity involves an interaction with them	Raw parts, Assembled car
Outcomes	The produced products	Assembled and tested car

Table 2.1: Typical components of a business process [18, pp. 1-6].

### 2.1.1 BPM Lifecycle

The BPM Lifecycle describes the different sub-fields and methods of BPM as a continuous cycle with different phases [18, pp. 22–24]:

1. Process identification: Processes relevant to a specific business objective are identified. The outcome is a process architecture, an overall picture of the processes and their relationships.
2. Process discovery: The current state of the identified processes is documented. The outcomes are as-is process models, models of the current state that are as close to reality as possible.
3. Process analysis: In this phase, the as-is process models are analyzed regarding improvement areas. The output is a structured collection of issues to solve.
4. Process redesign: This phase aims to redesign the as-is processes based on the identified issues to improve the models. These improved models are called to-be process models, the ideal state of the processes.
5. Process implementation: This phase aims to change the current as-is process models to the redesigned to-be process models. The changes are prepared and performed, resulting in actual organizational change.

6. Process monitoring: The running processes are monitored to detect any issues and determine if the change was successful. When new issues arise, the cycle repeats continuously.

The thesis's main focus lies in the discovery phase, the creation of as-is models. The literature distinguishes five discovery methods, which are often combined to get a holistic view of the actual processes [18, pp. 165–174]:

- Document analysis: Usually, documentation is already available about existing business processes. This can range from process documentation from previous modeling exercises to other documents like organizational charts, handbooks, work instructions, or system models. A key challenge is that most of the documentation is not directly available in a process-oriented way. The thesis aims to automate this process. The goal is to analyze available documents and extract process information automatically.
- Observation: Another method is to directly follow and observe the processing of individual cases to understand how the process runs. This can be performed actively (acting like the customer) or passively (watching the people doing their work).
- Automated discovery (process mining): Common enterprise systems produce process execution data, often event logs. This technique harvests the data and automatically extracts process information, allowing insights into the as-is processes.
- Interviews: Here, domain experts are interviewed to gather information about how a process is executed.
- Workshops: This technique is similar to the interviews. The difference is that workshops involve multiple domain experts simultaneously, with the advantage of resolving inconsistent views between the experts more quickly.

To summarize, the thesis tackles the discovery phase of the six BPM phases (Figure 2.1). There, the goal is to document the as-is processes. Typically, a combination of document analysis, observation, automated discovery (process mining), interviews, and workshops is used to achieve that goal. The thesis works toward automating the document analysis, in which already available documentation is used to extract the as-is processes.

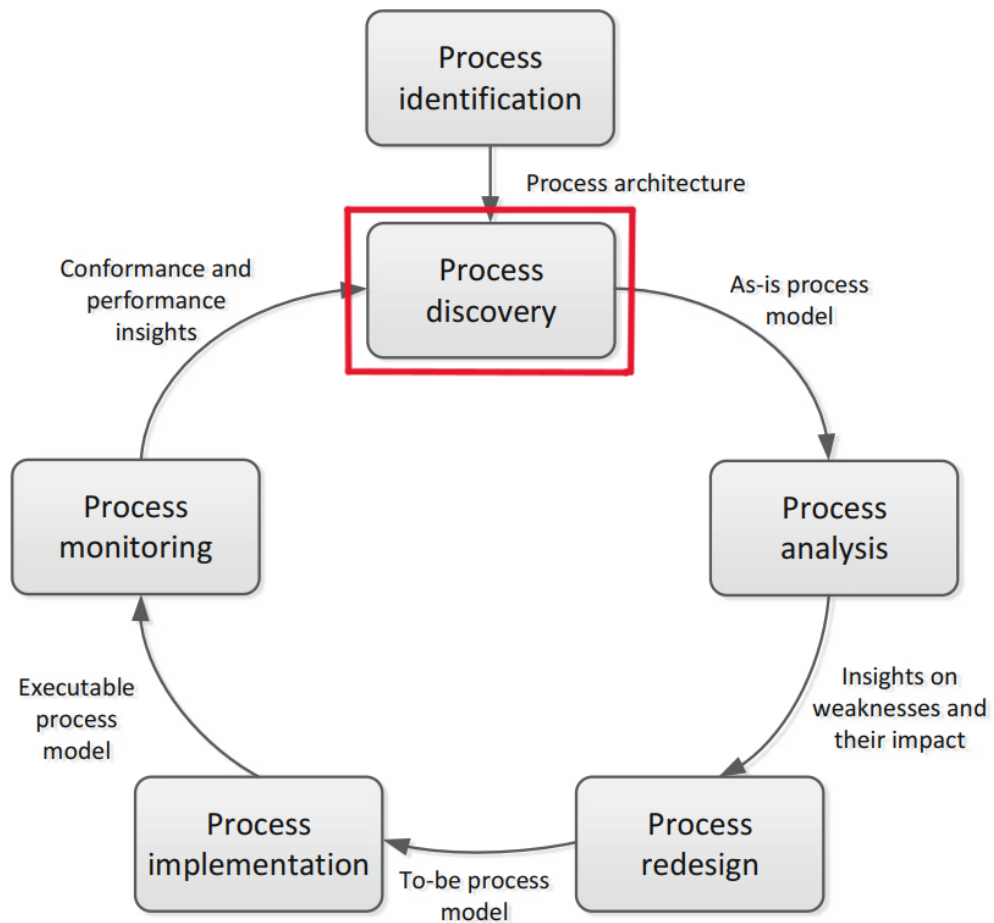


Figure 2.1: BPM Lifecycle. The thesis tackles the discovery phase and works towards automating the document analysis during that phase [18, pp. 22–24]

### 2.1.2 Business Process Modeling with BPMN

So far, the thesis has mentioned modeling as-is and improved to-be process models. This chapter introduces a specific notation for modeling the processes, the Business Process Model and Notation 2.0 (BPMN). BPMN is the leading standard for business process modeling [1, pp. 7–8]. It offers three different diagram types, with the collaboration diagram undoubtedly being the most important and most frequently used diagram [1, p. 9]. Thus, this chapter describes the basic concepts of the collaboration diagram.

Table 2.3 outlines the basic elements of BPMN. They are based on the typical

process components which were introduced in Chapter 2.1:

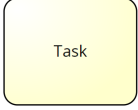

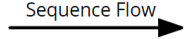


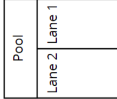
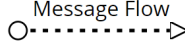
Element	Description	Notation
Tasks	A single unit of work to be performed, for example “Assembling the car”.	
Events	Things that happen passively and can trigger the start of activities, for example, “Customer order arrived”.	
Sequence Flows	Connects elements like tasks and events and determines the executing order. When there is a sequence flow from Event A to Activity B, it means that B is executed after A.	
Gateways	Decision points split and merge sequence flows. Parallel (meaning both sequence flows are executed) and Exclusive (meaning only one of the sequence flows is executed) are the most common.	  Parallel Exclusive
Pools and Lanes	Pools and Lanes model responsibilities, for example, an organization or a role. Lanes subdivide pools hierarchically.	
Message Flows	Messages Flows represent information flows between pools. It is similar to the sequence flow but only for cross-pool communication.	

Table 2.3: Basic elements of BPMN 2.0 [44, pp. 29–41]

With these main elements, modeling a basic business process is already possible. Figure 2.2 describes an example process to explain the interplay of the elements. There are two pools representing two different actors: The customer and the car manufacturer. The car manufacturer is subdivided into production and quality control lanes, representing different departments within the car manufacturer. The process starts in production when a customer order arrives. Then, the tasks *Prepare raw*

*material* and *Configure machines* are performed in parallel. That means their execution order is independent and does not matter. After both tasks are finished, the car is produced. Then, the car is tested by the quality control department. The following exclusive gateway means that just one of the outgoing paths can be selected, whether there is a detected error or not. If there is an error, the production department fixes the error, and the car is tested again. If there is no error, the car is successfully produced, and the customer gets notified.

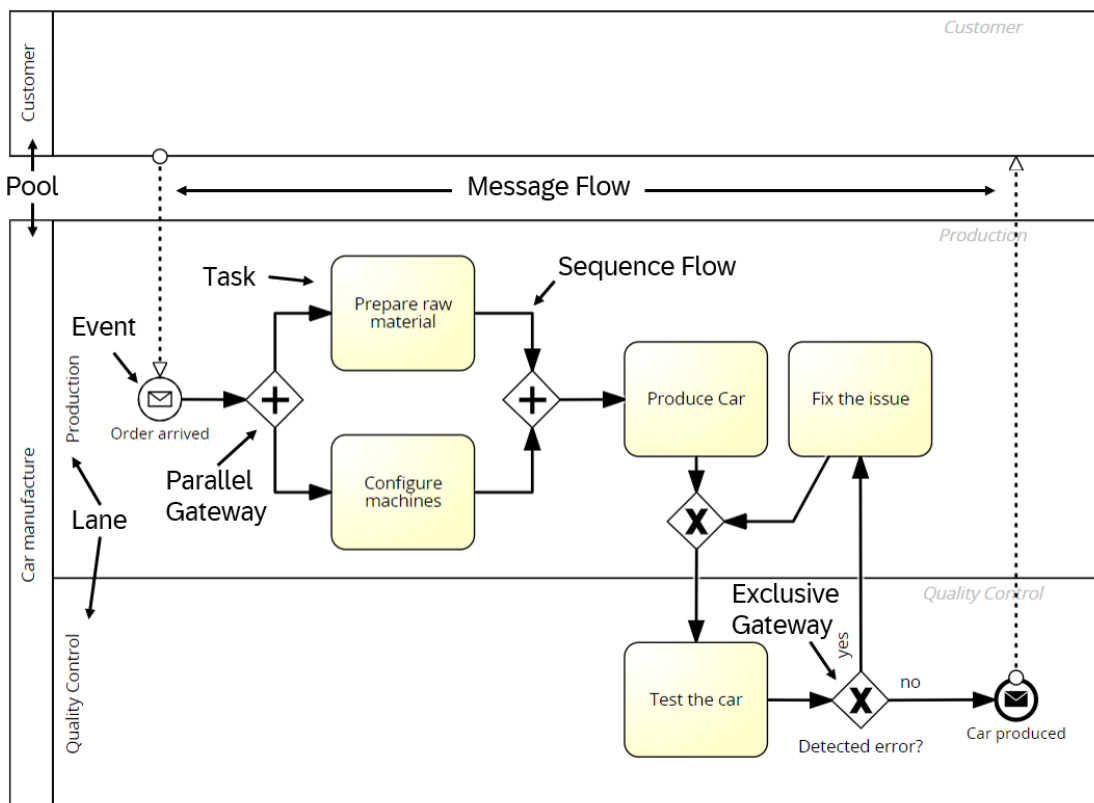


Figure 2.2: Example of a simple process in BPMN 2.0.

### 2.1.3 Advanced BPMN concepts

As shown in the previous section, modeling a simple process in BPMN is already possible with the elements mentioned in Table 2.3. More elements are available to enrich the model and increase the expressiveness [44, pp. 29–41]:

- Data Objects: Can be linked to tasks to specify data they need or produce when performing the task.
- Messages: Can be linked to a message flow to model the message explicitly.
- Text annotations: Allows a modeler to add additional textual information to document the process in more detail, like comments in a programming language.
- Associations: Links information and artifacts like the above elements with other elements. For example, a data object can be connected to a task.
- Groups: Serves the purpose of grouping other elements. Does not affect the Sequence Flows.
- Collapsed Sub-Processes: Indicates that the process can be broken down to a finer level of details not visible in the diagram.
- Expanded Sub-Processes: The process details are modeled and visible. They are grouped as an expanded sub-process. This also affects the Sequence Flow; they cannot cross the boundary.
- Activity Looping: Indicates that a task can be looped multiple times.

In addition, tasks can have different types to express more details [1, pp. 111–112]:

- Service Task: Automated task, for example, by a web service.
- Receive Task: Receives a message.
- Send Task: Sends a message.
- User Task: Needs user input, for example, entering information into a user interface dialog.
- Business Rule Task: Result or decision is derived from business rules.
- Script Task: Directly processed by the process engine.
- Manual Task: Performed without IT support.
- Abstract Task: Default when no type is defined.

The notation of the task types is depicted in Figure 2.3.

Events can have different types as well. The events can be characterized by the flow dimension [44, pp. 29–41]:

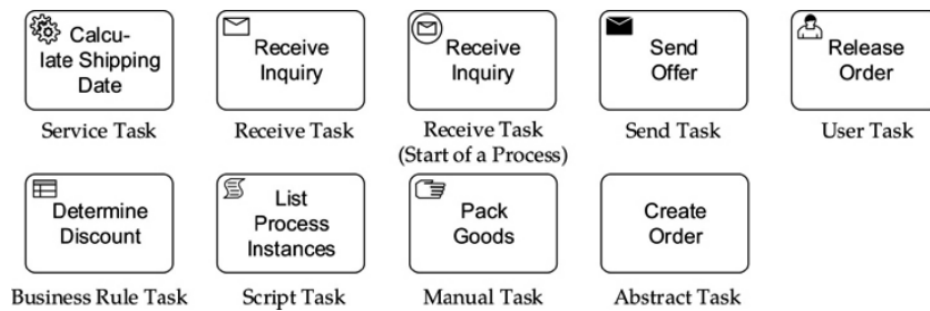


Figure 2.3: Notation of the task types [1, p. 113]. The type is indicated with an icon in the upper left corner.

- Start Event: Indicates where a process starts.
- End Event: Indicates where a process ends.
- Intermediate Event: Occurs between a start and end event. They affect the flow of the process but do not start or end it.

Each of these flow dimensions can be combined with a type dimension, for example [44, pp. 29–41]:

- Message Event: Sending or receiving messages.
- Timer Event: Waits until a timer triggers.
- Error Event: Indicates an error occurred.
- Escalation Event: Indicates an escalation occurred.
- Multiple Event: Indicates that multiple different types of events occur.

These two dimensions can be combined for events like Message Start Event, Message End Event, or Message Intermediate Event [44, pp. 29–41]. The full list of available events with their notation can be seen in Figure 2.4.

The gateways have different types as well [44, pp. 29–41]:

- Parallel: The paths are executed independently, and the order does not matter.
- Exclusive: Only one path is executed.
- Inclusive: One or multiple paths are executed.
- Event-Based and Parallel Event-Based: The execution path is determined by an event.



- Complex: For complex conditions and decisions.

The gateway type is noted as an icon within the diamond shape (see Figure 2.5).

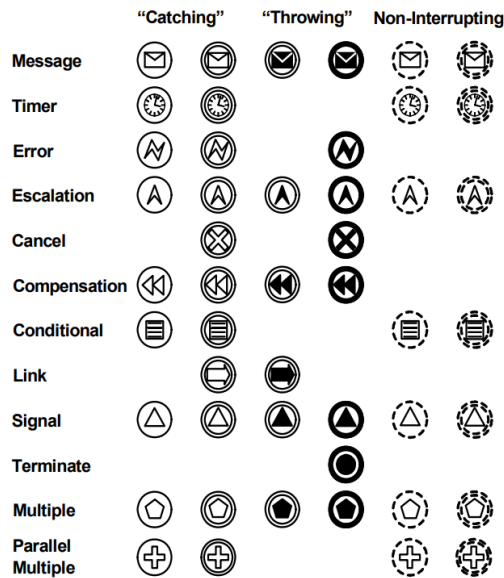


Figure 2.4: Task types and their notation [44, pp. 29–41]

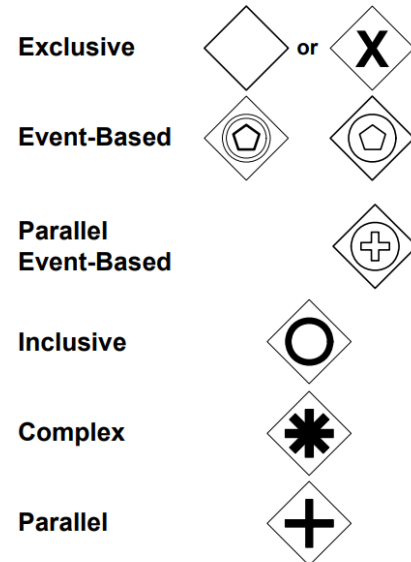


Figure 2.5: Gateway types and their notation [44, pp. 29–41].

For flows besides message and control flow, further specialized types exist: Conditional flow (only executed when the condition is true), default flow (used if no other flow can be executed), and exception flow (from an attached intermediate event to a boundary of a task). The different types of flows are noted as variations on the arrows [44, pp. 29–41].

In addition, some more special elements are not discussed here because they are rare. More about rareness and frequency of use in Section 2.1.4.

### 2.1.4 Most used BPMN elements

As described in the last section, BPMN has, in theory, many different elements. However, the thesis focuses on the practical applications of BPMN. Therefore, looking at the current frequency of use of the elements is interesting. Compagnucci et al. [13] have investigated the most used elements. Table 2.5 summarizes their

findings.

Usage Percentage	Elements
More than 50%	Sequence Flow, End None Event, Start None Event, Task, Exclusive Gateway
50% - 25%	Pool, User Task, Lane, Parallel Gateway, Message Flow, Start Message Event
25% - 10%	Association, Service Task, Intermediate Catch Timer Event, Intermediate Catch Message Event, End Terminate Event, Event Based Gateway, Collapsed Sub Process, Conditional Event, Text Annotation
10% - 3% (3% excluded)	Six special types of tasks, six special types of events, Message, Default Flow, Inclusive Gateway, Data Object, Group
3% or less	Mainly related to different types of tasks, events, and some special elements

Table 2.5: Distribution of BPMN Element Usage in Models. The categorization is based on their frequency of usage across different models [13, pp. 87–88].

To summarize, BPMN is the leading modeling standard for business processes. It consists of various elements to allow expressive modeling. However, only a subset of the elements is used.

## 2.2 Generative AI

This section aims to provide a clear overview of Generative AI and its significant accomplishments. First, it briefly introduces the concept of Artificial Intelligence (AI) to set the context. Then, it explains deep neural networks in detail. Afterward, it delves into the traditional generative models and the latest generation of generative models. Lastly, it discusses Large Language Models, ChatGPT, and multimodal models.

### 2.2.1 Context

Artificial Intelligence (AI) is a computer science field that deals with artificial machines that mimic the cognitive functions of humans. Machine Learning (ML) is a sub-field of AI that focuses on machines that can learn by themselves without being explicitly programmed [45]. There exist multiple levels of learning [45] [25, pp. 104–106]:

- Supervised learning: The machine learns by using labeled data, meaning data that consists of input data and the expected output. The expected output is referred to as the label or ground truth.
- Unsupervised learning: The machine learns by using unlabeled data and searches patterns in this data without further information about the ground truth.
- Reinforcement learning: The dataset is not fixed. The machine learns while it interacts with the environment.

Deep Learning (DL) is a subset of ML that focuses on using deep neural networks as a learning technique [45]. Generative AI is part of this discipline [2, pp. 20–21]. Figure 2.6 illustrates the field of AI and in which sub-fields Gen AI is placed.

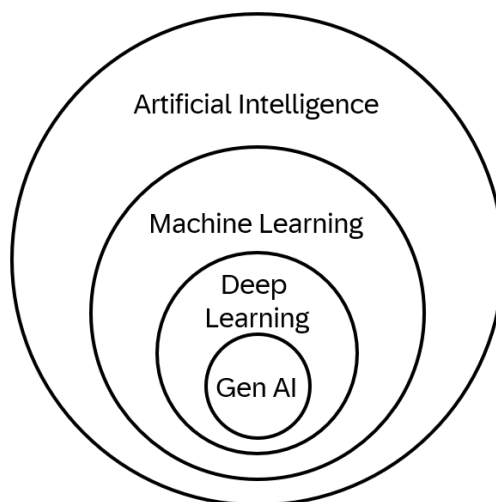


Figure 2.6: Generative AI is part of Deep Learning, which is part of Machine Learning, which is a sub-field of Artificial Intelligence [2, pp. 20–21].

### 2.2.2 Deep Neural Networks

The idea of neural networks is to mimic the behavior of the human brain, namely the functionality of neurons to activate each other and make intelligent decisions based on the activation. A neuron is thereby defined as a single unit. It takes inputs, multiplies them with weights, sums them up, and adds a bias. The final output then goes through an activation function. Figure 2.7 illustrates an example neuron. The neuron takes the inputs  $x_i$ , multiplies them with the weights  $w_i$ , and sums them up. This is written as a matrix multiplication  $wx$ . Then, a bias  $b$  is added. The final output is calculated using the sigma as a non-linear activation function [25, pp. 168–177].

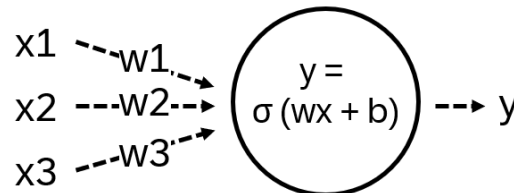


Figure 2.7: Example of a neuron. [25, pp. 168–177].

A neural network consists of multiple neurons grouped together as different layers. First, the input is processed by the input layer. Then, it goes through the hidden layers and finally to an output layer, which makes sure the needed output is produced. Figure 2.8 illustrates the three different types of layers. The hidden layers are what make the deep neural networks so powerful and also give them their name. Deep refers to the deepness of the hidden layers. They can be used to extract different features. For example, for image recognition, one layer might extract edges, while another might extract facial parts, which are then united in recognizing the whole face [25, pp. 168–171].

The neurons and layers can be represented as a computational graph. Then, with given inputs, the final output can be calculated. This process of calculating the output is called feed-forward. The computed output is then compared to the expected output (ground truth) with a loss function that computes the difference. Then, derivations of the loss function with respect to parameters are calculated, and the parameters are updated to minimize the difference. This process is called backward propagation and allows the network to learn. The process is scalable to many neurons and layers, making it possible to build different networks with different building blocks

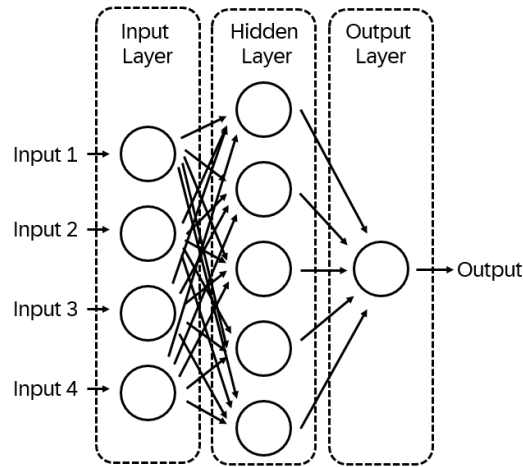


Figure 2.8: Example of a neural network with multiple neurons grouped together in three layers. The input goes through the input layer, then through a hidden layer, and finally through the output layer. In contrast to this simplified example, NNs often consist of multiple hidden layers [25, pp. 168–171].

using the same learning mechanism. Figure 2.9 shows a simple computational graph illustrating its feed-forward and back-propagation. It takes  $x$ ,  $y$ , and  $z$  as input and calculates  $g$  and the final output,  $f$ , for the feed-forward calculation. The backpropagation calculates the gradients backwards:  $\frac{df}{dx} = \frac{df}{dg} \cdot \frac{dg}{dx} = z \cdot 1 = -4$ ,  $\frac{df}{dy} = \frac{df}{dg} \cdot \frac{dg}{dy} = z \cdot (-1) = 4$ , and  $\frac{df}{dz} = g = 3$ . The neural network's weights are then updated based on the calculated gradients and deviations, which allows the network to learn [25, pp. 205–227].

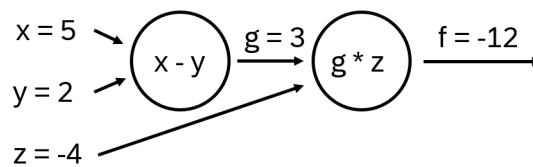


Figure 2.9: Example of a computational graph with the feed-forward calculation. The backpropagation results in these gradients:  $\frac{df}{dx} = \frac{df}{dg} \cdot \frac{dg}{dx} = z \cdot 1 = -4$ ,  $\frac{df}{dy} = \frac{df}{dg} \cdot \frac{dg}{dy} = z \cdot (-1) = 4$ , and  $\frac{df}{dz} = g = 3$ . [25, pp. 205–227].

### 2.2.3 Convolutional Neural Networks

Convolutional Neural Networks (CNNs) are neural networks that use filters to process spatial hierarchies in data. That makes them powerful for tasks like image recognition. There exist a variety of different filtering techniques. Figure 2.10 shows an example of one filter technique. It consists of a kernel placed over the input like a mask and takes the weighted sum. The first step:  $-5 \cdot 0 + 3 \cdot 2 + 4 \cdot 1 + 0 \cdot 3 = 10$  (blue). Then, the kernel moves one number to the right (green), one line down (red), and again one number to the right (purple). This allows the capture of spatial information with far fewer weights than it would have been with a fully connected layer. It is common to use multiple filters like these across layers to handle multiple various spatial hierarchies [25, pp. 330–335].

Input	Kernel	Output
<div style="display: inline-block; border: 1px solid black; padding: 5px; margin: 2px;"> <div style="border: 1px solid green; padding: 2px; display: inline-block;"> -5   3   2 </div> </div>	$\begin{matrix} 0 & 2 \\ 1 & 0 \end{matrix}$	$=$
<div style="display: inline-block; border: 1px solid black; padding: 5px; margin: 2px;"> <div style="border: 1px solid red; padding: 2px; display: inline-block;"> 4   3   2 </div> </div>	$*$	$=$
<div style="display: inline-block; border: 1px solid black; padding: 5px; margin: 2px;"> <div style="border: 1px solid purple; padding: 2px; display: inline-block;"> 1   0   3 </div> </div>	$*$	$=$
		$\begin{matrix} 10 & 7 \\ 7 & 4 \end{matrix}$

Figure 2.10: Example of applying a kernel for filtering within a CNN. The kernel is placed over the input like a mask to take the weighted sum. It starts on the top left (blue), moves one number to the right (green), one line down (red), and again one number to the right (purple) [25, pp. 330–335].

### 2.2.4 Recurrent Neural Networks

Recurrent Neural Networks (RNNs) are networks that can handle sequential data, for example, multiple sentences or time series data. As the name suggests, RNNs achieve that with recurrent connections and shared parameters. The output is based on the last input and a combination of all past inputs. Through this technique, RNNs can handle sequential data while memorizing the previous part of the sequence. For instance, when it comes to a sentence, the network takes the last word and the entire previous sequence of words into account. Figure 2.11 illustrates an RNN neuron. It takes inputs  $x_t$  over time  $t$  and has two activation functions. The tanh function calculates the current cell state  $c_t$  based on the current input  $x_t$  with its

weight  $u$  and based on the previous cell state  $c_{t-1}$  with its weight  $w$ . The final output  $y_t$  is calculated with the softmax function of  $c_t$  and the weight  $v$ . Incorporating the past states in the current calculations makes the network recurrent and allows it to handle data sequences. While this is a promising approach, RNNs face challenges learning long-range dependencies between the sequences and training issues with vanishing and exploding gradients [25, pp. 373–416].

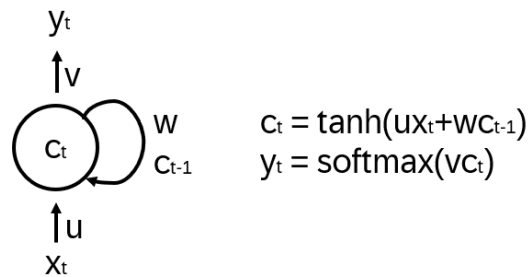


Figure 2.11: An RNN neuron remembers inputs from past sequences over time  $t$  within the cell state  $c_t$  [25, pp. 373–416].

The Long Short-Term Memory (LSTM) approach was introduced to improve these difficulties. This more advanced RNN has concepts for forgetting and remembering part of the data, making it better at learning long-term dependencies. While the concept brings improvements, LSTM still faces challenges with long sequences of data [25, pp. 408–413].

## 2.2.5 Traditional Generative Models - VAEs and GANs

Generative Models have the ability to generate new data, for example, text or images. Traditionally, Variational Autoencoders (VAEs) and Generative Adversarial Networks (GANs) are prominent models within that category. Both models work unsupervised. [25, pp. 696–703].

VAEs transform the input into a lower-dimensional representation (encoding) and then reconstruct it into the original output (decoding). The encoding phase maps input data to a probability distribution in a latent space, while the decoding phase samples from this distribution to generate outputs that are similar to the original input data. For example, an image of a face might be encoded in probability distributions about gender, hairstyle, skin color, facial expression, nose type, etc. The probability

distributions are then decoded into an image of a face again to generate an exact replica of the original face. CNNs are often used to decrease and increase the dimensions. One advantage is that VAEs operate unsupervised, with the encoder teaching the decoder the data's expected structure, minimizing reconstruction error and KL divergence. After training, the decoder can be used to generate data, for example, images [25, pp. 696–699]. Figure 2.12 illustrates the idea.

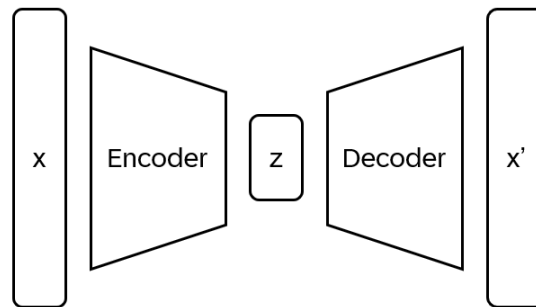


Figure 2.12: VAEs consist of an encoder that transforms the input  $x$  into a lower-dimensional representation  $z$  and a decoder that tries to generate an output  $x'$  that is the same as the original input  $x$ . After the self-supervised training, the decoder can be used to generate data [25, pp. 696–699].

The main idea behind GANs is that they consist of two models that compete with each other, a Generator and a Discriminator. Figure 2.13 illustrates the idea. Generator  $G$  generates images from input noise. Then Discriminator  $D$  gets the generated images and real images and tries to classify the images as generated or real. Thereby,  $G$  and  $D$  compete against each other and train themselves. If  $D$  identifies the generated image,  $G$  gets updated. Otherwise,  $D$  gets updated. No labeled images are needed as input, just the images themselves and some noise. CNNs play a crucial role in the architecture of both the generator and discriminator in image-related tasks [25, pp. 699–703].

### 2.2.6 Modern Generative Models - Transformers

The current breakthroughs of LLMs like ChatGPT are mainly based on the transformer architecture introduced by Vaswani et al. in the paper “Attention is All You Need” [61]. Transformers are designed to handle sequential data, like text,



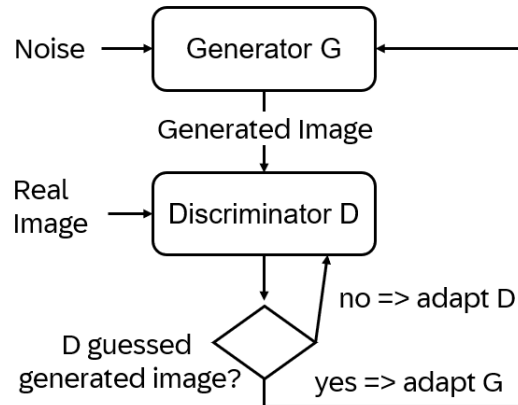


Figure 2.13: GANs consist of a Generator G that generates images and a Discriminator D that tries to classify images as generated or real. G and D compete against each other to train themselves. [25, pp. 699–703].

more efficiently than RNNs or CNNs by relying on attention mechanisms instead of sequence-aligned recurrence. Figure 2.14 shows the transformer’s architecture. It consists of an encoder and a decoder. Both start by creating embeddings with positional encoding. The positional encoding is important because, in contrast to CNNs, the data is passed in parallel, and therefore, the sequential positional data has to be memorized in the encoding. Then, the attention layers are applied, the fundamental innovation of the transformer. Furthermore, the transformer consists of well-known components of NNs like normalization layers, skip connections, a linear layer, and a softmax output layer [61].

As stated above, the attention layer is the pivotal innovation of the transformer’s architecture. It models the relationships between words in a sentence. The attention mechanism uses the vectors  $Q$ ,  $K$ , and  $V$  to calculate an attention score, which states how much focus to put on specific parts of the input data when processing a part of the data.  $Q$  is the query that represents a possible relation question for the current item.  $K$  is the key and represents the strongness of relations to answer the query.  $V$  is the value and represents some more context for the  $K$ , for example, how exactly it helps to solve the query. Figure 2.15 gives an example of what the vectors might represent. For the word “mouse” in the sentence “The cat chased the mouse into the box, but it escaped” the query might be about the action of the mouse (Q5). The attention mechanism identifies different levels of relevance across the sentence: “chased” is somewhat relevant to the action of the mouse (weak hint),

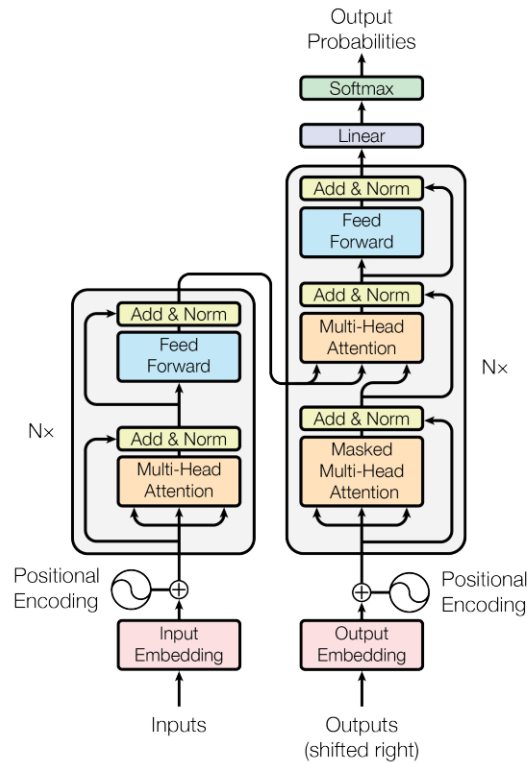


Figure 2.14: The transformer architecture consists of an encoder (left) and a decoder(right), both of which create embeddings with positional encoding. Special attention blocks are applied, along with other NN components like normalization layers, skip connections, a linear layer, and a softmax output layer [61].

while “escaped” is highly relevant (strong hint). The value associated with “chased” (V3) might encode information about the mouse being forced into action due to the cat’s chase. The value associated with “escaped” (V11) captures the mouse performing an action by itself, directly answering the question. This example shows that the attention mechanism helps understand relations within the sentences and which words are relevant to each other. Instead of performing a single attention operation, the transformer uses multiple sets of  $Q$ ,  $K$ , and  $V$  weight matrices to project the input embeddings into different representational spaces. This is called multi-head attention and allows the model to capture different representations of the attention relationships [61].

The concepts of the transformer have significant advantages over RNNs and CNNs [61]:

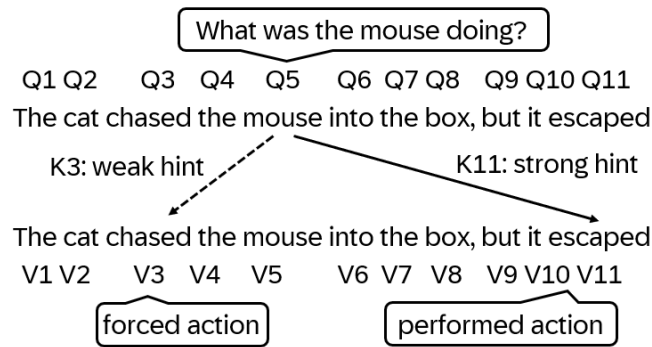


Figure 2.15: Example of what the values Q, K, and V might represent in the transformer's attention mechanism. Q5 might be a query about the action of the mouse. K3 and K11 might be identified as weak and strong hints to answer the query. Value V3 may represent the mouse being forced into action, while V11 might capture the mouse acting by itself [61].

- **Parallel Processing:** Unlike RNNs that process data sequentially, transformers can process entire data sequences in parallel, which significantly speeds up training and allows for more effective handling of longer sequences.
- **Handling Long-Range Dependencies:** RNNs struggle with long-range dependencies even with LSTM variants due to vanishing and exploding gradient problems. Transformers, through self-attention, can directly model relationships between words in a sentence, regardless of their positions, making them more effective at understanding context.
- **Scalability:** The parallel processing and long-range-dependency handling allow the transformer architecture to scale more effectively by adding more parameters and training data. This is crucial for building models with billions or even trillions of parameters like many LLMs.
- **Adaptability:** The transformer's architecture is highly adaptable, allowing for various configurations (e.g., encoder-only, decoder-only, or encoder-decoder pairs) to suit different Natural Language Processing (NLP) tasks, from text generation to understanding and translation.

The original transformer model consists of an encoder to process the input text and a decoder to generate the output text. Generative Pre-trained Transformer (GPT) and Bidirectional Encoder Representations from Transformers (BERT) are two prominent models based on the transformer model. GPT uses the transformer's decoder only,

while BERT uses the encoder only [47, 15]. GPT is trained to predict the next word in a sequence based on the previous words. Its main goal is to generate text, which is why it facilitates the decoder part of the transformer, which is built for that purpose [47]. In contrast, BERT is trained on the Masked Language Model (MLM) and Next Sentence Prediction (NSP). In MLM, random words in a sentence are masked, and the model predicts these words based on the bidirectional context provided by the unmasked words. In NSP, the model predicts whether one sentence logically follows another. Its main objective is to understand the context and meaning of the text through advanced language comprehension. For this purpose, it employs the encoder component of the transformer, which is specifically designed for this task. It is because of this context aware word embeddings that it can be applied to many down stream NLP tasks [15].

### 2.2.7 Large Language Models

Large Language Models (LLMs) are technically based on the transformer. As pointed out, in contrast to traditional approaches, the transformer makes it possible to scale the model to a large amount of data and parameters. This idea is what gives the LLMs the name. The key differentiation from previous generations is the possibility of training the model using the huge amount of data available on the Internet or in books without a need for labeling. This leads to the current trend to create larger and more complex models. However, training these models on enormous datasets requires specialized hardware and is very expensive. This has led to a situation where only a few companies with sufficient resources can afford to train these models. These companies then offer the trained models as foundational models that can be used directly or fine-tuned for specific purposes [2, pp. 83–103].

Trained on huge amounts of data, the LLMs show significant improvements [2, pp. 83–134]:

- They understand language and its context better than smaller models.
- They produce language that comes close to a human-like writing style.
- They show some level of commonsense reasoning.
- They can transfer their learning to various tasks by giving instructions or examples.

- Some believe they could lay the ground for Artificial General Intelligence (AGI), although this is highly controversial and questionable.

Despite the improvements, LLMs also face some challenges [2, pp. 83–134]:

- LLMs can create false information that was not present in the training data. This is known as “hallucinations”. These fabricated facts can be written convincingly, making it easy to believe them even if they are incorrect.
- LLMs may not always accurately follow instructions because their current reasoning and understanding capabilities are limited.
- It is often possible to trick safety instructions against creating fake news, hate speech, or other violations.
- LLMs can contain bias based on the training data.

### 2.2.8 ChatGPT

ChatGPT is a popular LLM capable of answering questions, following instructions, and engaging in a textual conversation. ChatGPT is based on the GPT architecture and trained in three phases [67, 69]:

*Pre-training on Large Datasets:* The foundational step in ChatGPT’s training involves pre-training on a diverse and extensive collection of text data sourced from the Internet. This initial phase uses unsupervised learning, exposing the model to various topics, writing styles, and information. This stage aims to equip the model with a broad understanding of language, including syntax, semantics, and general world knowledge. During pre-training, ChatGPT learns to predict the next word in a sentence given the preceding words, a process that helps grasp context, coherence, and the intricacies of language. This extensive pre-training is what gives ChatGPT its base capability to generate text across a wide range of subjects and formats [67, 69].

*Fine-tuning through Supervised Learning:* ChatGPT undergoes a fine-tuning process using supervised learning after the pre-training phase. In this stage, the model is trained on curated datasets that contain structured pairs of prompts and responses. These datasets are designed to mimic conversational exchanges, including questions

and answers, dialogues, and other forms of interactive communication. Supervised fine-tuning helps ChatGPT to improve its understanding of conversational context and how to generate relevant, accurate, and coherent responses. This phase significantly enhances the model's ability to engage in dialogues, answer queries, and perform specific language tasks as requested by the user [67, 69].

*Refinement with Reinforcement Learning from Human Feedback (RLHF)*: The final stage in ChatGPT's training involves refinement through Reinforcement Learning from Human Feedback (RLHF). This advanced training method incorporates human judgment into the learning loop. AI trainers, who are humans, interact with the model by providing prompts and then rank the model's responses according to relevance, coherence, and appropriateness. These rankings are used as feedback to train the model further, employing reinforcement learning techniques. RLHF allows ChatGPT to fine-tune its responses based on human preferences, aligning its outputs with what is considered valuable or desirable in human communication. This stage is critical for addressing nuances and complexities in conversations that are not fully captured through supervised learning alone, such as subtleties of tone and style and the avoidance of generating harmful or biased content [67, 69].

The combination of pre-training on large datasets, fine-tuning through supervised learning, and refinement with RLHF is what enables ChatGPT to achieve its capabilities in generating human-like text. This comprehensive training methodology enhances the model's linguistic abilities, adaptability to various conversational contexts, and alignment with human values and preferences [67, 69].

### 2.2.9 Multimodality

Multimodality in LLMs refers to concepts allowing LLMs to handle further modalities beyond text. The most common modality besides text is images, but video, voice, or other forms are also possible.

Incorporating multimodality in LLMs is a very recent development, and the technology forefront is moving fast. OpenAI published their multimodal model GPT-4V(ision) right around the start of this thesis [68]. Google published their multimodal model Gemini shortly before finishing this thesis, which is too late for this thesis to investigate it further [26].

GPT-4V(ision) is built and trained similar to ChatGTP. The difference is that GPT-4V is also trained on a vast amount of images surrounded by text [68]. This enables various capabilities, including but not limited to the following [65]:

- It can handle interleaved sequences of images and texts as input.
- It can use information from both the texts and the images to follow instructions.
- It can understand pointers and references to the image.
- It understands spatial relationships in the images and can localize objects.
- It can understand tables, charts, and whole documents.

To summarize, Generative AI and LLMs show various improvements in Natural Language Processing and reasoning tasks compared to traditional approaches. Their key innovation lies in the transformer architecture with the attention mechanisms, which makes the model scalable to a huge amount of training data. Multimodal models transfer these capabilities to other modalities like images. For example, it enables document analysis with LLMs, which is the aim of this thesis.

### 3 Related work

This thesis aims to use multimodal LLMs to generate business process models from documents with text and images. To the best of the author's knowledge, no paper currently tries to achieve that. However, various related approaches exist in BPM and AI research areas.

Regarding positional papers in the space of BPM and AI, Vidgof et al. outline the opportunities and challenges of LLMs for the BPM domain [62]. Similarly, Kampik et al. set up a vision for applying LLMs to the BPM domain [33]. Van der Aa et al. outline the opportunities and challenges of applying NLP in BPM [29]. Dumas et al. set up a vision of AI-augmented Business Process Management Systems and possible research questions to achieve it [17]. Beheshti et al. propose opportunities for training an LLM on business process data to achieve various BPM tasks [4]. Similarly, Rizk suggests a process-specific Foundational Model and what opportunities it could offer [50]. In contrast to these positional papers, the thesis validates a specific use case in depth.

Current research exists in the context of generating process models from text with LLMs or NLP. Klievtsova et al. proposed application scenarios for modeling business processes with LLMs based on textual conversations [35]. Bilal et al. found 11 NLP and 8 BPMN tools that could generate models from textual requirements utilizing NLP [40]. Friedrich et al. and Sholiq et al. proposed NLP techniques to generate process models from natural language [21, 56]. Bellan et al. use LLMs for extracting business process entities and relations from texts [5, 6]. Grohs et al. create imperative and declarative process models from textual descriptions [28]. Unlike these papers, the thesis emphasizes generating business models through multimodal documents incorporating texts and images, not just text alone.

In the context of using images to generate process models, Schaefer et al. proposed a NN based on CNNs to create BPMN models from sketches [53]. Kang et al.



transform BPMN business process images into Petri nets with traditional deep learning techniques [34]. Antinori et al. recreate business process models from BPMN images using traditional object recognition and Optical Character Recognition (OCR) methods [3]. Gantayat et al. create BPMN models from images using traditional CNNs [23]. In contrast, the thesis explores a multimodal context where images and text converge within documents. Additionally, it leverages multimodal LLMs, diverging from the conventional methods highlighted in the preceding papers. This approach offers greater versatility since an LLM can perform multiple tasks using fundamental reasoning skills without being exclusively trained on a specific type of image or text.

Furthermore, LLMs can be leveraged for process mining, as Berti et al. demonstrated [8]. Moreover, Berti et al. proposed abstractions, scenarios, and prompt definitions for LLM-based process mining [9]. Jessen et al. examined different LLM prompts in the process mining domain [32]. The distinction between process mining and document analysis, as addressed by the thesis, lies in the primary sources utilized: process mining relies on event logs, whereas document analysis focuses on documents, which are often multimodal.

Regarding process modeling, Sola et al. recommended suitable labels for a new activity during modeling using LLMs [57]. Farkas examined how to recommend the next element during modeling using LLMs [63]. The thesis focuses not on modeling support but on document analysis instead.

Other fields where LLMs are applied to the BPM domain are Business Process Automation and Predictive Process Monitoring. Mandvikar et al. introduced a framework to automate processes with LLMs [39]. Desmond et al. examined authoring business automation using natural language [14]. Cabrera et al. and Brennig et al. used NLP to utilize natural language texts for Predictive Process Monitoring [12, 10]. Fahland et al. used LLMs for explaining business process models [19]. Generative AI can also be used in the domain of business process improvements, as van Dun et al. demonstrated [72]. In contrast to these papers, the thesis concentrates on a different aspect of the BPM lifecycle, specifically the discovery phase.

In summary, existing work relevant to the thesis is found in positional papers, text-to-BPMN, and image-to-BPMN conversions. However, no existing method utilizes

multimodal Large Language Models (LLMs) for documents that combine images and text, as the thesis intends to achieve. Additionally, there is research in other application fields outside the document analysis domain, such as process mining, process modeling, process automation, predictive process monitoring, and business process improvement.

## 4 Contribution Overview

As stated in Chapter 1.2, the goal of this thesis is to evaluate the feasibility and suitability of generative AI methods for generating process models from multimodal documents with images and text. The thesis should:

- Provide benchmarking possibilities for the fast-changing environment.
- Aim to measure the current Gen AI capabilities.

To achieve that, the thesis provides three contributions:

1. It introduces a dataset with multimodal process documentation and ground truths to establish a common ground (Chapter 5).
2. Then it applies current generative AI methods to generate the process models from the process documentation (Chapter 6).
3. Finally, it introduces an evaluation framework to objectively measure the generation methods' performance against the ground truth (Chapter 7).

Figure 4.1 visualizes the interplay of the contributions. With the three contributions, the thesis objectively measures the current Gen AI capabilities for multimodal BPM document analysis. Furthermore, contributions one and three provide an objective benchmarking environment for future developments.

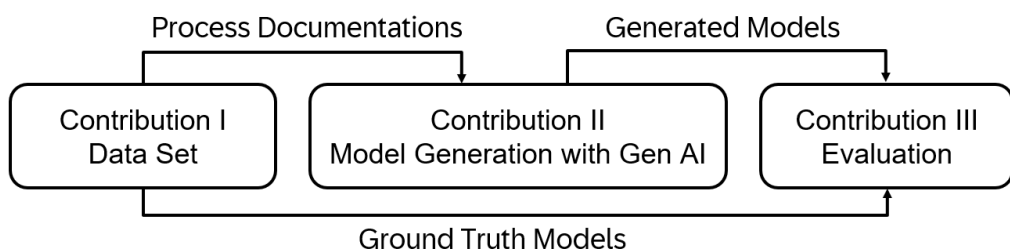


Figure 4.1: Interplay of the three contributions

## 5 Contribution I: Data Set

This chapter introduces a ground truth data set to establish a common foundation for objectively measuring the performance of LLM systems. First, the requirements of the data set are defined. Then, existing data sets are evaluated based on the requirements. A new dataset is created as these do not fully fulfill the requirements. Finally, the created dataset is evaluated against the requirements.

### 5.1 Requirements

The thesis aims to generate structured process models from multimodal images and text documents. Therefore, the data set must contain multimodal documents containing pages of images and text describing process models. Furthermore, it needs to contain ground truths of the process models in a structured format.

The multimodal documents should be representative with regard to the models described. As described in Chapter 2.1.2, BPMN 2.0 is the most used modeling language. Therefore, if process models are visually presented, they should be pictured in BPMN 2.0. Furthermore, it should include the most used elements of BPMN 2.0 as described in Chapter 2.1.4. In addition, it should contain realistic models with a diversity of different domains. The documents should also be available as images without losing information because some LLMs only take images and text via the API but not a whole document in PDF format.

LLMs should be able to generate the ground truths' structured format easily. LLMs commonly offer the capability *function calling* to force JSON output format [46, 27]. Therefore, the format of the ground truth should be in JSON. JSON is a common format that allows easy transformations in other formats if needed. In addition, the ground truth should be in a very simple format and contain only the most important

information about a process. That makes it easily understandable for LLMs and reduces the cost as fewer tokens are used. Furthermore, the ground truth format should be able to capture the most used BPMN elements and be extendable to further datasets.

The data set should contain enough data points to measure performance accurately. The thesis sets a goal of at least 100 models to achieve an appropriate level of representativeness. As no training of LLMs is planned with the data set, more models are not necessarily needed.

To summarize the data set requires the following:

- RQ1 - Multimodal documents with multiple pages of texts and images describing business processes.
- RQ2 - An image representation of the documents.
- RQ3 - Ground Truth in a simple JSON format that can capture the most used BPMN 2.0 elements.
- RQ4 - Representative BPMN 2.0 models.
- RQ5 - Minimum of 100 data points.

## 5.2 Existing data sets

There are multiple datasets of business process models. The SAP Signavio Academic Models (SAP-SAM) dataset is a collection of hundreds of thousands of business models, mainly in BPMN notation. It was created by curating a subset from roughly a decade of created models at [academic.signavio.com](https://academic.signavio.com), a platform where researchers, teachers, and students model business processes [58]. Camunda offers a similar but smaller and less diverse dataset [24]. The main challenge regarding the requirements is that these datasets do not contain multimodal documents describing the business process.

The Business Process Model and Textual Description (MaD) dataset contains 30,000 pairs of Business Process Models and their textual descriptions. One challenge of this dataset is that it only uses a simplified version of BPMN (no type of activities,

limited type of events, gateways, no lanes or message flows) [37]. The PET dataset contains 45 annotated text-model pairs [7]. Leopold et al. provide additional 53 model-text pairs [36]. Klievtsova et al. currently provide 24 textual descriptions with multiple ground truth models per description. The advantage of the multiple ground truth models is that they capture the idea of multiple equivalent models of a business process [35, 43]. All these datasets only provide textual descriptions and not multiple page-long multimodal documents, which conflicts with the requirements.

Hand-drawn BPMN models with ground truths are available in the hdBPMN - hand-drawn BPMN dataset [53]. The BPMN-Redrawer Dataset offers images of BPMNs with ground truths [3]. However, neither offers multimodal documents describing the processes.

In summary, multiple datasets of business process models, model-text pairs, and model-image pairs exist, but no dataset includes multimodal documents.

### 5.3 Creation of the data set

As stated in Chapter 5.2, currently, no suitable dataset exists. Therefore, this thesis creates a new dataset to meet the requirements. To create the dataset, first, a foundational dataset is chosen. Then, the foundational dataset is cleaned by filtering suitable and representative models to ensure high data quality. Afterward, the cleaned dataset is used to generate multimodal process documentation with SAP Signavio Process Manager, a commercial process management tool [51]. Furthermore, the cleaned process models are parsed into a simplified JSON format to meet the requirements of the ground truth. Figure 5.1 illustrates the process.

The thesis uses the SAP-SAM as a foundational dataset because it offers many processes, and some of them have parts of textual descriptions included, which is important for multimodal documents. The huge amount of processes and different element types gives enough space to clean the dataset and provide a representative distribution in the cleaned dataset. Additionally, it is already available in a format that the Signavio Process Manager can use.

In the cleaning step, the SAP-SAM set is examined and filtered to be representative and meet some technical requirements. The experiments carried out as part of

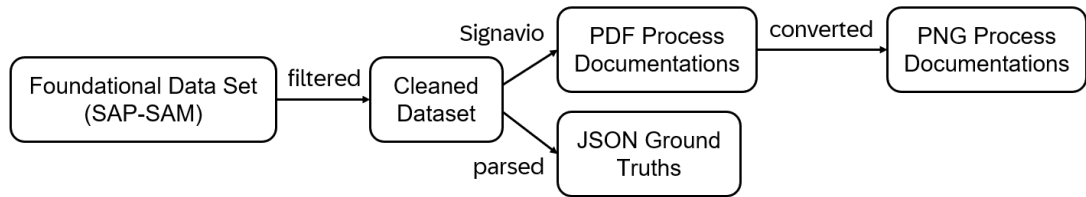


Figure 5.1: The creation process of the dataset builds up on the SAP-SAM dataset. Then, a cleaned dataset is created by applying filters. Then, Process Documentations in PDF format are created via the Signavio Process Manager and converted into images. Furthermore, the JSON Ground Truths is created.

this work have shown that the following filters increase the quality of the data set significantly:

- BPMN 2.0 models only: As stated in Chapter 2.1.2, most models are currently modeled in BPMN 2.0.
- English and German models only: The author can only verify the quality of models in these languages fluently.
- Models with description properties only: This increases the text in the process documentation, which is created later.
- Minimum of four activities per model: Models with a lower number of activities were often low-quality.
- Models with minimum one lane: Models with no lanes often showed low quality, and lanes frequently occur in BPMN models as shown in Chapter 2.1.4.
- Models where the longest label is a minimum of five letters: This ensures no empty models or dummy models are used, for example, models that contain the activities *A1*, *A2*, and *A3* only.
- Models with different names: Often, models with the same name are duplicates. It increases the number of representatives across different domains.
- Models with valid syntax checked by Signavio Process Manager Syntax Checker: Non-compliant models often indicate low quality.
- Models that passed manual checks: A few models were removed manually because of their bad quality. For example, some had an extensive usage of

dummy names like John Doe, and others had a lot of disconnected elements that were not linked with flows.

Furthermore, models with expanded subprocesses are excluded because the format of the ground truth, which is introduced later, does not support message and control flow within expanded subprocesses. This is acceptable as expanded subprocesses are only present in 3 percent of business process models [13, pp. 87–88]. In contrast, collapsed subprocesses are included. In addition, models with less than 100 elements are included only because they are problematic for the resolution of the image representation of the documents. Moreover, models with stacked lanes (lanes in lanes) were removed because they can not be represented by the ground truth JSON. Both exclusions are acceptable because they are rare in the dataset and represent edge cases.

The cleaned dataset is then uploaded to [academic.signavio.com](https://academic.signavio.com). Signavio Process Manager offers functionality to generate the required multimodal process documentation for the uploaded models. As this functionality is unavailable as an API, a Selenium script is created and used to generate the process documentation automatically [55]. In addition, the process documentation is converted into images without losing any information, where each image represents one page of the documents.

Furthermore, the complicated original structure of the SAP-SAM dataset is parsed into a simplified JSON structure with only the essential information. The exact format is described in Section 5.4.

### 5.4 Characteristics of the data set

After applying the filters, the cleaned dataset consists of 123 cleaned process models. For each model, the following four parts exist: The original SAP-SAM data format, the multimodal process documentation, the image representation of the documentation, and the ground truth in a simple JSON format (Figure 5.2). The original SAP-SAM data format is complex and irrelevant to this thesis.

The multimodal process documentation is a PDF with 4-18 pages (see the frequency of pages in Figure 5.3), which describes a process. The first pages contain a cover page, a table of contents, and some meta-information. Then, the process is



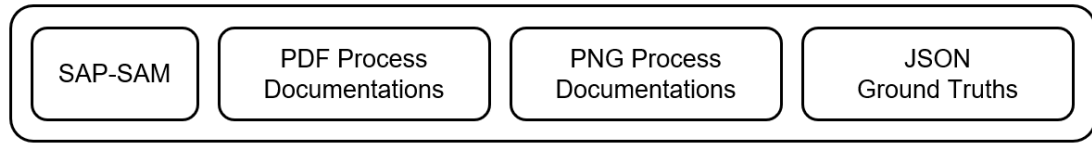


Figure 5.2: The created data set consists of the original SAP-SAM format, the Process Documentations in PDF and PNG format, and the Ground Truths in JSON.

illustrated as a BPMN diagram as an image. Afterward, it follows a short description of the process, if available. Finally, each element is described briefly. The information in the element description is often equivalent to the pictured model. Only sometimes is additional information for the elements provided as a short description. The image representations are exact copies of the process documentation in a PNG format where every page is one image.

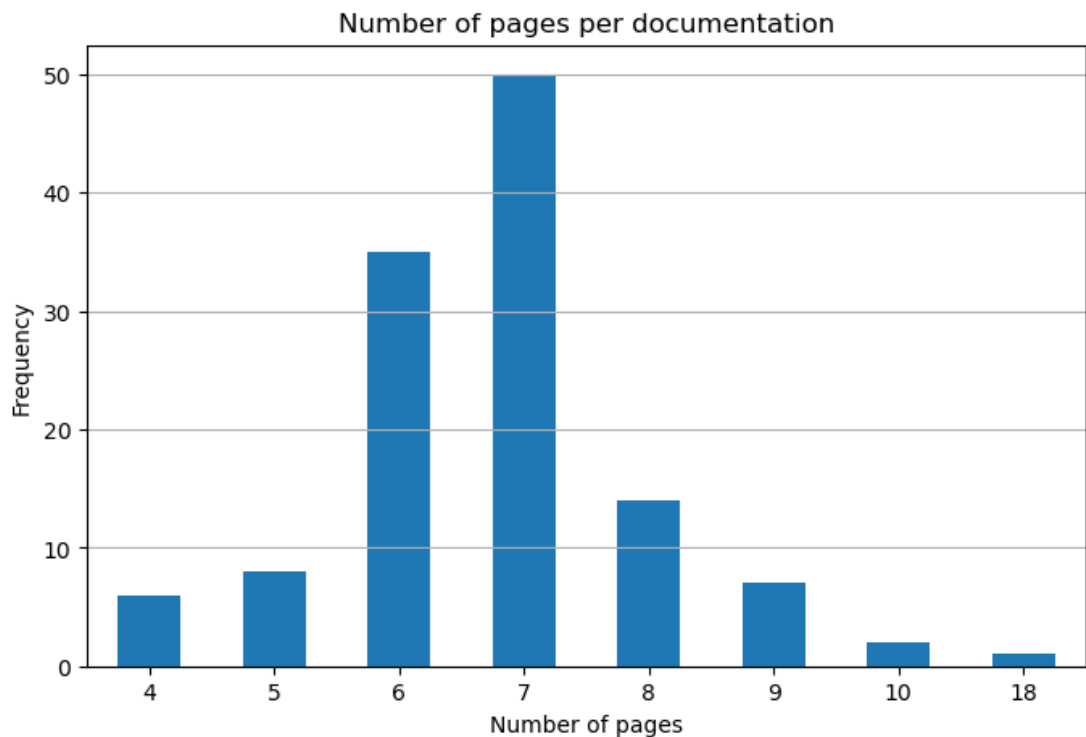


Figure 5.3: Frequency of the number of pages of the process documentation within the created dataset.

The JSON ground truth schema effectively encapsulates the key BPMN 2.0 elements

(see Figure 5.4). The schema's business process consists of referencable objects with IDs. A referenceable object can either be a task, an event, a gateway, a pool, a lane, a message flow, or a sequence flow. Tasks, events, and gateways have a name and a type, while the name is optional for gateways. A pool has a name and can have multiple lanes. A lane has a name and can contain multiple referenceable objects. A message flow has an optional label and exactly one target and one source reference to referencable objects. The same goes for sequence flows; the only difference is that the label attribute is called condition. Listing 5.1 shows an example instance of the JSON format. The complete JSON schema definition of the ground truth is located in the appendix in Listing A.1.

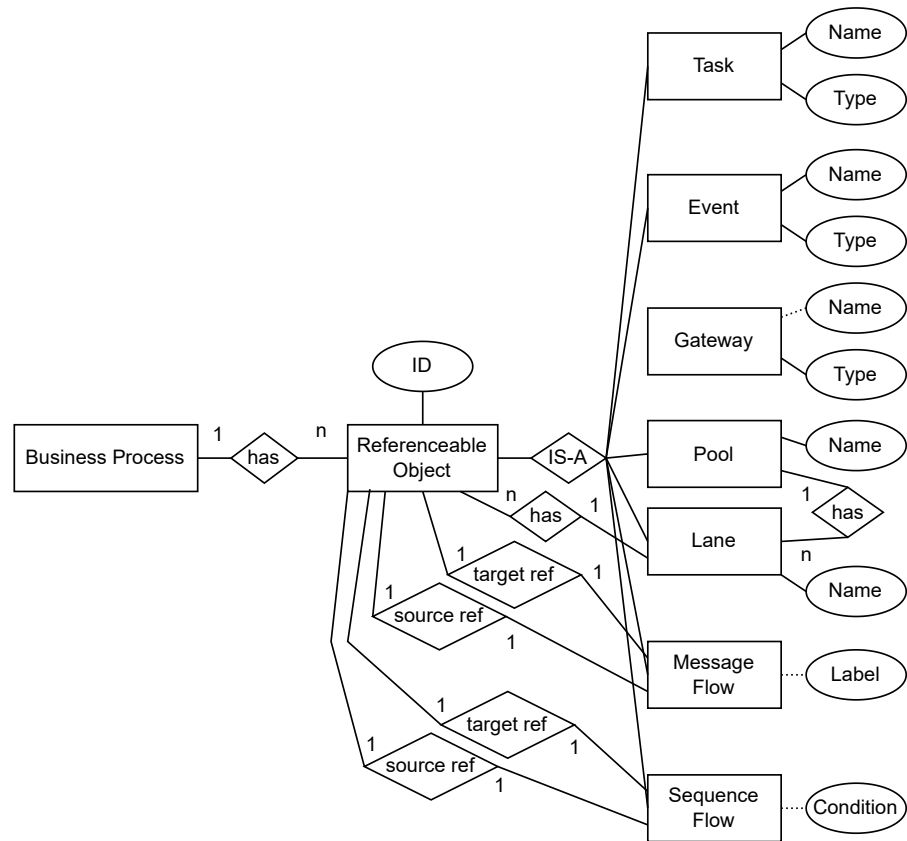


Figure 5.4: ER model of the ground truth schema.

```

1 {
2   "tasks": [
3     {
4       "id": "0",
5       "name": "Produce the car",
6       "type": "UserTask"
    }
  ]
}

```

```

7      },
8      {
9          "id": "1",
10         "name": "Test the car",
11         "type": "ServiceTask"
12     },
13     {
14         "id": "2",
15         "name": "Fix the issue",
16         "type": "UserTask"
17     }
18 ],
19 "events": [
20     {
21         "id": "3",
22         "name": "Order arrived",
23         "type": "StartMessageEvent"
24     },
25     {
26         "id": "4",
27         "name": "Car produced",
28         "type": "EndMessageEvent"
29     }
30 ],
31 "gateways": [
32     {
33         "id": "5",
34         "name": "Detected errors?",
35         "type": "Exclusive"
36     },
37     {
38         "id": "6",
39         "type": "Exclusive"
40     }
41 ],
42 "pools": [
43     {
44         "id": "7",
45         "name": "Car manufacture",
46         "lanes": [
47             {
48                 "id": "8",
49                 "name": "Production",
50                 "elemRefs": [
51                     "0",
52                     "2",
53                     "3",
54                     "6"
55                 ]
56             },
57             {
58                 "id": "9",
59                 "name": "Quality Control",
60                 "elemRefs": [
61                     "1",
62                     "4",

```

```

63             "5"
64         ]
65     }
66 ]
67
68 },
69 {
70     "id": "10",
71     "name": "Customer",
72     "lanes": []
73 }
74 ],
75 "sequenceFlows": [
76     {
77         "id": "11",
78         "sourceRef": "3",
79         "targetRef": "0"
80     },
81     {
82         "id": "12",
83         "sourceRef": "0",
84         "targetRef": "6"
85     },
86     {
87         "id": "13",
88         "sourceRef": "6",
89         "targetRef": "1"
90     },
91     {
92         "id": "14",
93         "sourceRef": "1",
94         "targetRef": "5"
95     },
96     {
97         "id": "15",
98         "sourceRef": "5",
99         "targetRef": "2",
100        "condition": "no"
101    },
102    {
103        "id": "16",
104        "sourceRef": "5",
105        "targetRef": "4",
106        "condition": "yes"
107    },
108    {
109        "id": "17",
110        "sourceRef": "2",
111        "targetRef": "6"
112    }
113 ],
114 "messageFlows": [
115     {
116         "id": "18",
117         "sourceRef": "10",

```

```

119         "targetRef": "3"
120     },
121     {
122         "id": "19",
123         "sourceRef": "4",
124         "targetRef": "10"
125     }
126 ]
127 }
```

Listing 5.1: JSON Ground Truth Example

The JSON format covers the most common elements with frequency rates from 100 - 25 %, which are described in Chapter 2.1.4. To cover all elements that are used in more than 3% of models, the schema can be adapted by adding a new list for associations, a new list for object elements with the types *data object*, *text annotation*, *message*, and *group*, as well as a flow type for default flow.

The number of elements per ground truth in the dataset ranges from 13 to 88, with 39 being the mean. The mean indicates that the models are of good quality and compliant with the process modeling guideline of using not more than 50 elements per model [41]. The frequency distribution is pictured in Figure 5.5.

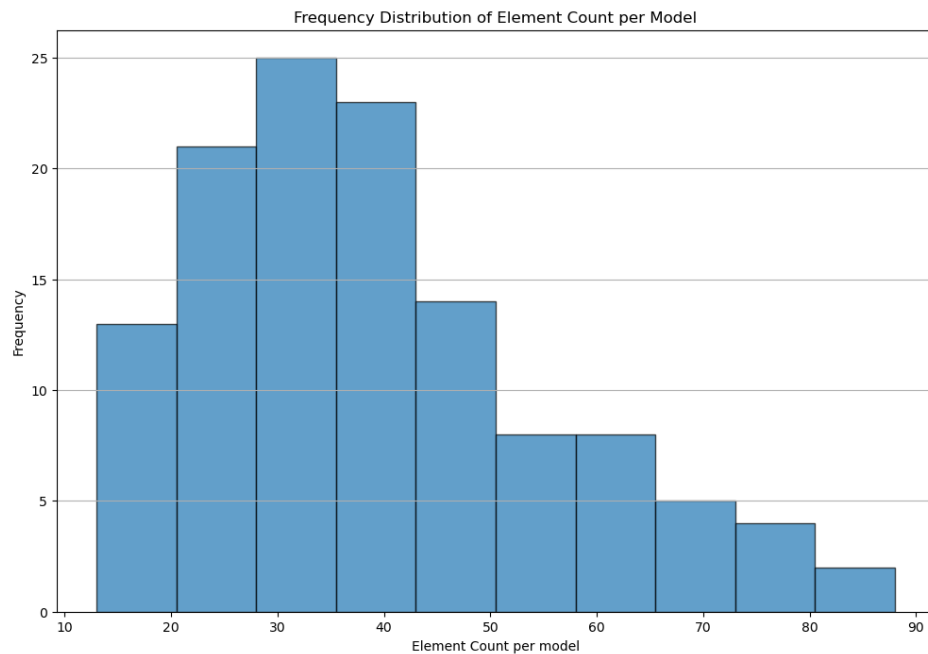


Figure 5.5: Element frequency distribution across all models.

Tasks and Sequence Flows are the predominant elements utilized, as illustrated in Figure 5.6. Among these are 10 distinct types of tasks, with abstract and manual tasks emerging as the most frequently employed, as depicted in Figure 5.7. Additionally, the dataset encompasses 17 varied event types, among which the End None Event and Start None Event stand out as the most commonly utilized, as shown in Figure 5.8. Furthermore, the analysis identifies 4 unique types of gateways, with exclusive and parallel gateways being the preferred choices, as evidenced in Figure 5.9. This pattern aligns closely with the typical distribution observed in other models, as discussed in Chapter 2.1.4.

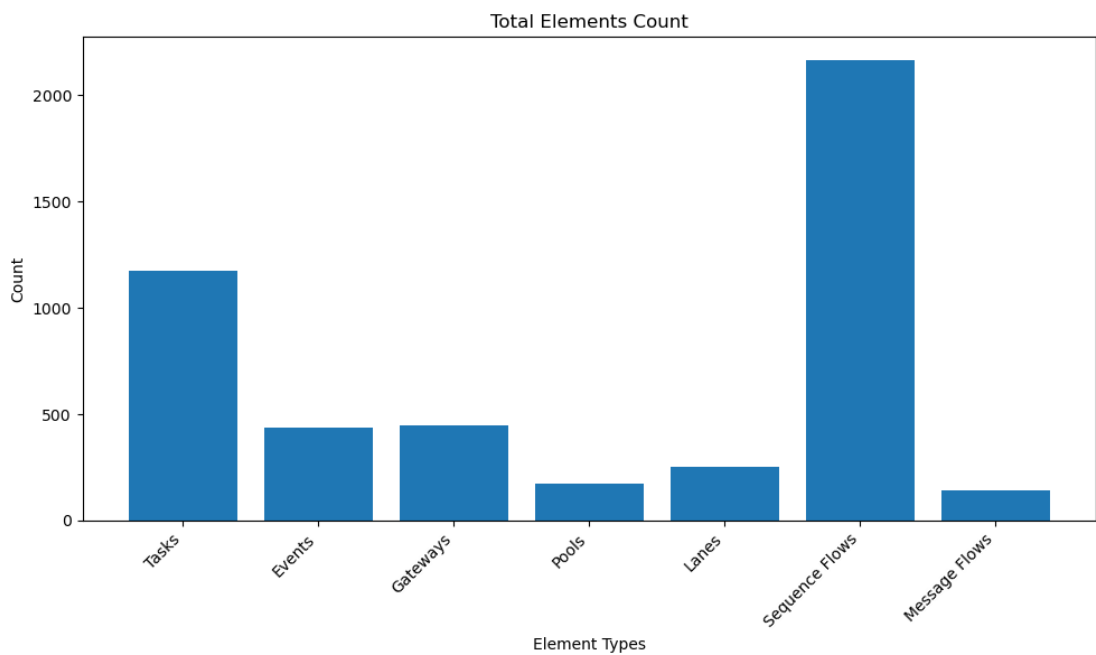


Figure 5.6: Total number of elements over all the 123 models in the dataset. The most frequently used elements are Sequence Flows and Tasks.

## 5 Contribution I: Data Set

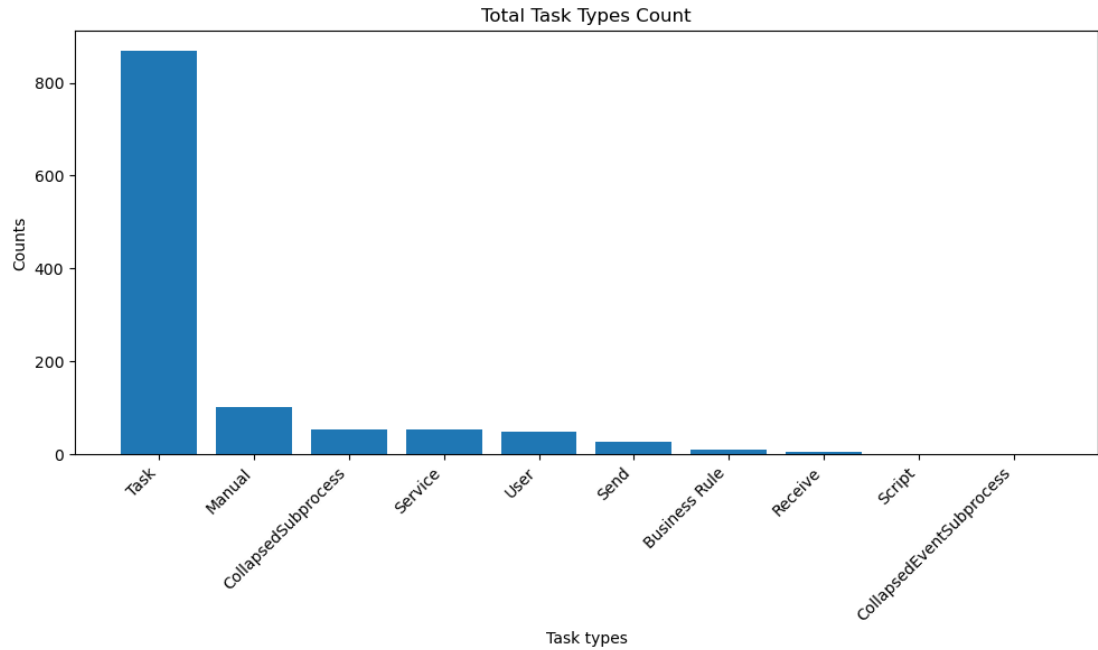


Figure 5.7: Total number of task types over all the 123 models in the dataset. The most frequently used task types are the abstract task and manual task.

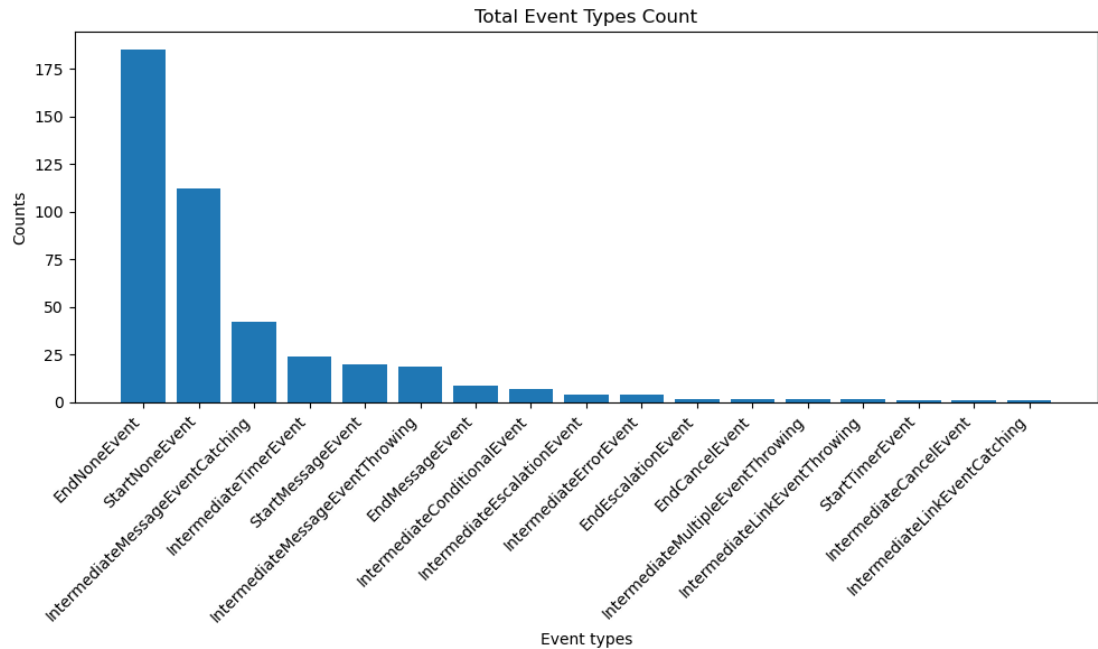


Figure 5.8: Total number of event types over all the 123 models in the dataset. The most frequently used event types are End None and Start None.

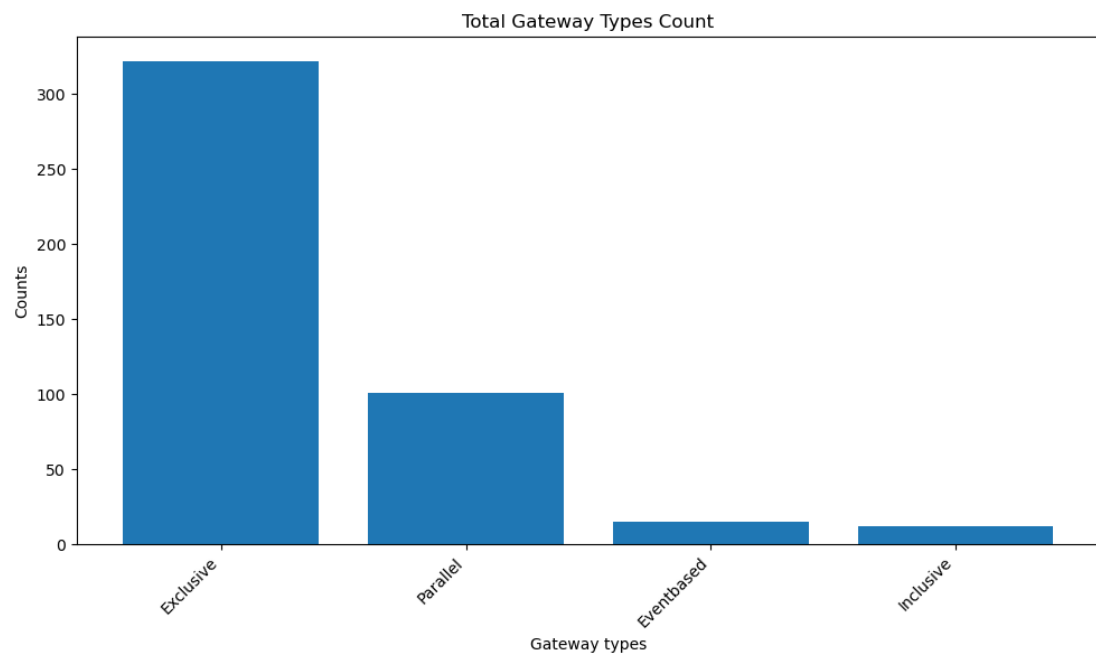


Figure 5.9: Total number of gateway types over all the 123 models in the dataset. The most frequently used gateway types are the exclusive and parallel gateways.



## 5.5 Evaluation of the data set

This section evaluates the dataset regarding the requirements from Chapter 5.1. The requirements are the following:

- *RQ1 - Multimodal documents with multiple pages of texts and images describing business processes:* The created process documentations fulfill that aspect. The documentation's distribution of text and images is questionable. While text and images are present, the diagram in the documentation contains all the necessary information to meet the ground truth. The dataset could be further improved if some information is only available in the images and others are only available in the text to improve the multimodal portion.
- *RQ2 - An image representation of the documents:* Met with the created images per documentation page.
- *RQ3 - Ground Truth in a simple JSON format that can capture the most used BPMN 2.0 elements:* Met with the created JSON schema. It supports the most common elements (frequency rate 100 - 25 %) and can be easily extended to support all elements used in more than 3 % of usual models.
- *RQ4 - Representative BPMN 2.0 models:* This requirement is fulfilled because, as shown, the element distribution is similar to the usual distribution.
- *RQ5 - Minimum of 100 data points:* Met with 123 data points.

To summarize, the requirements of the dataset are mostly fulfilled. A questionable part is the distribution of text and images because the image alone already contains all the information necessary to create the ground truth. This is acceptable because the LLM still reads the whole document. Furthermore, the text contains some descriptions which are not present in the diagram. The ground truths could be extended by that information in future versions to meet the multimodal requirement better.

## 6 Contribution II: Model Generation with Gen AI

In this chapter, the Generative AI-guided generation of process models is performed. First, current multimodal models and methods are described and chosen. Afterward, the conceptual design of the generation process is presented. Finally, the concept is executed.

### 6.1 Current multimodal models

As described in Chapter 1.2, the thesis is written at the forefront of technology. High-performing multimodal LLMs are rare, although promising models are currently published, and more models will probably be available in the future. The thesis is created from October 2023 until April 2024. In November 2023, OpenAI published its first multimodal LLM, GTP-4V (Vision), which the thesis includes in the evaluations [70]. Google published the Gemini Series starting in December 2023 with limited preview access. In February 2024, they introduced the multimodal model Gemini 1.5 Pro [31]. The model seems very promising but was published too late to be considered for further evaluation in the thesis. Grok, the LLM from xAI founded by Elon Musk and built on X (Twitter), has announced multimodal capabilities in the future [73]. Further, a lot of smaller and often open-source LLMs are available [22].

In December 2023, Fu et al. benchmarked current multimodal LLMs [22]. They evaluated the multimodal LLMs on different tasks. The following are relevant to our use case:

- Existence: Recognition if items exist in the image.
- Count: Counting the number of objects.

- Position: Understanding the positions of the objects.
- OCR: Recognition of characters and texts in the image.
- Commonsense Reasoning (short Comm.): Answering questions on the images that need additional commonsense reasoning.
- Code Reasoning (short Code.): For example, guessing the output based on the image of the code.

Table 6.1 shows the scores of the top six performing LLMs. Overall, GPT-4V performed best, outperforming the other LLMs, especially in the categories OCR and Code Reasoning. These categories are highly relevant for the use case (OCR for recognizing the labels of the entities and Code Reasoning for creating the JSON output). After GPT-4V, follow WeMM, XComposer-VL, InfMLLM, SPHINX, and LLaVA.

<b>Models</b>	<b>Total</b>	<b>Existence</b>	<b>Count</b>	<b>Position</b>	<b>OCR</b>	<b>Comm.</b>	<b>Code.</b>
GPT-4V	942	190	160	95	185	142	170
WeMM	867	195	140	127	148	140	118
XComposer-VL	824	190	158	127	125	139	85
InfMLLM	802	190	152	143	133	132	53
SPHINX	776	195	160	153	88	130	50
LLaVA	774	185	155	133	125	128	48

Table 6.1: Comparison of top six multimodal models in different performance categories. The maximal score per category is 200 (for total 1200) [22].

As stated in Chapter 1.2 this contribution aims to achieve the best current performance while accepting the risk of closed systems. Therefore, the thesis uses GPT-4V for the generation process. For future considerations, Gemini seems worth looking into as well.

## 6.2 Current approaches

Multiple approaches exist to apply LLMs to a task. This section gives an overview of the most interesting candidates. One very important aspect when applying LLMs is prompt engineering. Prompt engineering refers to crafting the input prompt for an LLM, guiding it to perform specific tasks [2]. The following are important prompt engineering strategies that often increase performance significantly [2]:

- Clear and precise: Explaining clearly what the LLM should do by keeping it short and precise.
- Examples: Giving LLMs examples of the task.
- Constraints: Specifying constraints in the prompt to restrict the output. For example, restring the output to JSON.
- Priming: Introducing a context, for example, telling the LLM *You are a BPMN expert*.
- Iterative refinement: Predicting prompt performance is hard. Refining prompts iteratively based on outputs helps.
- Question decomposition: Breaking down complex questions into simpler sub-questions.

The thesis applies the strategies of clear, precise, constrained, and primed prompts developed with further iterative refinements. Moreover, the thesis evaluates the improvements of giving zero, one, or multiple examples. The literature refers to the following methodology [11]:

1. Zero-Shot Prompting: Giving the LLM no example at all to perform the task.
2. One-Shot Prompting: Providing the LLM one example to perform the task.
3. Few-Shot Prompting: Providing the LLM multiple examples to perform the task.

Furthermore, the thesis explores question decomposition using the chain-of-thought technique, linking smaller queries (thoughts) to address larger ones through repeated LLM prompts, enhancing answers but raising costs. Due to these costs exceeding the thesis budget, the concept will be outlined but not applied to the dataset.

Additionally, LLMs frequently support function calling, enabling them to produce JSON that complies with the given function definition. This feature for JSON output creation is not yet available in GPT-4V but is accessible in its procedures. It seems interesting for future considerations.

Another LLM technique is fine-tuning. Fine-tuning re-trains part of the LLM with task-specific data. The effort needed exceeds the budget of this thesis. This seems to be a promising approach to increase performance for future work.

To summarize, the thesis applies fundamental prompt engineering techniques and evaluates zero-shot, one-shot, and few-shot approaches in more detail. Furthermore, chain-of-thought is considered conceptually. Function calling and fine-tuning offer potential but remain inaccessible or beyond the resource scope for this thesis.

## 6.3 Conceptual design

This chapter outlines the conceptual design of generating business process models from the process documents. As stated in Chapter 6.2, the thesis utilizes different prompt engineering strategies. It examines four approaches in more detail: zero-shot, one-shot, few-shot, and chain-of-thought-prompting.

The meta prompt is pictured in Listing 6.1 and follows different prompting strategies. First, it uses priming: “You are a BPMN expert.” Then, it clearly and precisely describes the task the LLM should perform: extracting the process information from the passed document. Then, the prompt constrains the output to only generate the JSON according to the passed schema. The scheme corresponds to the defined ground truth schema in Chapter 5.4 and is located in the appendix in Listing A.1.

```
1 meta_prompt = """### Instruction ###
2   You are a BPMN expert. Your task is to extract process
      information out of the passed documents which are
      parsed as a list of images where each image
      represents one page of the document. Make sure that
      you include the sequence and message flow. Use
      numbers for the ids starting from zero. Generate
      json according to the following schema for
      extracting the process information. Only output the
      generated json.
3
4   ### Schema ###
5   """ + str(bpmn_schema)
```

Listing 6.1: Meta prompt

The meta prompt was refined based on iterative experiments highlighting adding the sequence and message flow; otherwise, they were often empty. The instruction for counting the IDs starting from zero was added to increase consistency.

For the zero-shot approach, the LLM is prompted with the meta-prompt and a list of images representing the pages of the process documentation (Listing 6.2). Interestingly, it performed better with the user message first, followed by the system message.

```
1 def zero_shot(LLM, process_documentation_images):
2     return LLM.ask([
3         {
4             "role": "user",
5             "content": process_documentation_images
6         },
7         {
8             "role": "system",
9             "content": [meta_prompt]
10        }
11    ])
```

Listing 6.2: Zero-shot approach

For the one-shot approach, one pair of process documentation image and JSON output is given as an example. Further instructions are provided to clearly separate the example from the other parts. In this case, the order of the system and user message is switched back because it performs better and is in the logical order. The full prompt can be seen in Listing 6.3.

```
1 def one_shot(LLM, example_image, example_json,
2               process_documentation_images):
3     instruction_example = """### Example ###
4     Use the following image json pair as an example
5     """
6     instruction_user = """### User Input ###
7     Now create the json based on these images
8     """
9     return LLM.ask([
```

```

9      {
10         "role": "system",
11         "content": [
12             meta_prompt,
13             instruction_example,
14             example_image,
15             example_json
16         ]
17     },
18     {
19         "role": "user",
20         "content": [instruction_user] +
21                     process_documentation_images
22     }
23 ])
```

Listing 6.3: One-shot approach

The few-shot approach follows the same logic but with three example pairs and is illustrated in Listing 6.4.

```

1 def few_shot(LLM, example_images, example_jsons,
2               process_documentation_images):
3     instruction_example_1 = """### Examples ###
4     Use the following image json pairs as an example
5     ### Example 1 ###
6     """
7     instruction_example_2 = "### Example 2 ###\n"
8     instruction_example_3 = "### Example 3 ###\n"
9     instruction_user = """### User Input ###
10    Now create the json based on these images
11    """
12    return LLM.ask([
13        {
14            "role": "system",
15            "content": [
```

```

15         meta_prompt ,
16         instruction_example1 ,
17         example_images [0] ,
18         example_json [0] ,
19         instruction_example2 ,
20         example_images [1] ,
21         example_json [1] ,
22         instruction_example3 ,
23         example_images [2] ,
24         example_json [2] ,
25     ]
26 },
27 {
28     "role": "user",
29     "content": [instruction_user] +
30                 process_documentation_images
31 }
32 ] )

```

Listing 6.4: Few-shot approach

Figure 6.1 illustrates the chain-of-thought approach. It splits the generation process into smaller parts and chains them together to achieve the goal. First, the LLM is prompted with the process documentation to generate the tasks, events, and gateways separately, resulting in three independent prompts. The results of the three prompts are assembled to an intermediate model  $M_1$ . Then, the LLM is prompted to generate the pools inclusive lanes and referenced elements based on  $M_1$  and the process documentation. This results in the second intermediate model  $M_2$ . In the last step, the LLM is instructed to generate the final complete model  $M_3$  with sequence and message flow from  $M_2$  and the process documentation.



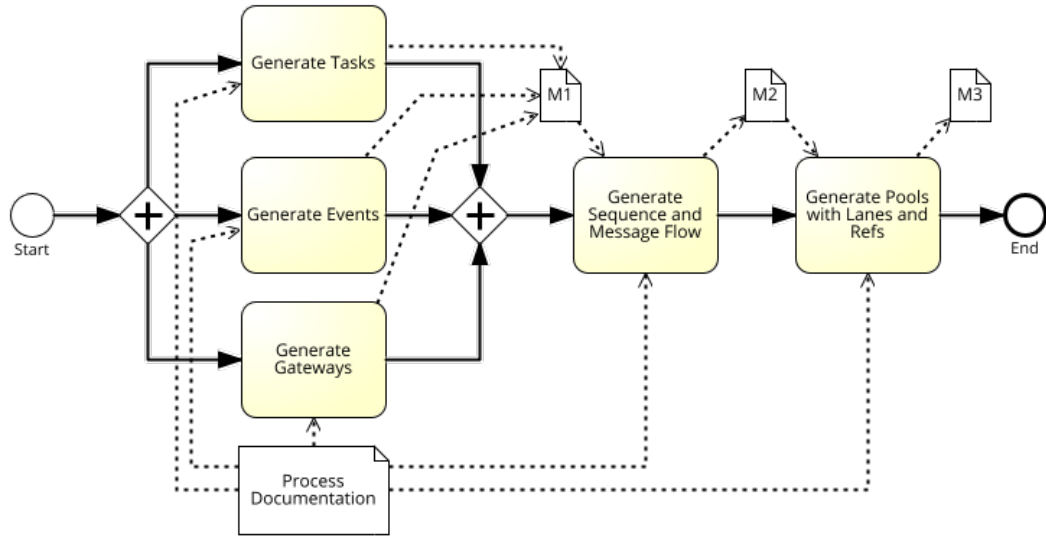


Figure 6.1: In the chain-of-thought approach, five prompts are chained together. That results first in the intermediate models  $M_1$ ,  $M_2$ , and then a final complete model  $M_3$ .

## 6.4 Execution

This section describes the execution of the zero-, one-, and few-shot approaches. The chain-of-thought approach is excluded because the first experiments were successful and showed some improvements but were too costly to be performed on a representative dataset.

To execute the approaches, the 123 models from the introduced dataset (see Chapter 5.4) are taken. The first three models are used as examples of the one- and few-shot approach. Each approach is applied to the remaining 120 models. The LLM used is GPT-4V, as explained in Chapter 6.1. A single generation took approximately 45 to 60 seconds. Due to the high-resolution images, the input tokens varied from around 17,000 to 20,000 tokens for the few-shot approach. The produced answer consumed around 700 to 1500 tokens, resulting in a price of around \$0.20 per generation.

The generation process ran stable, resulting in no breaking errors. The generated answer could be parsed as valid JSON without any problems. Some schema violations occurred when validating the generated JSON against the JSON schema introduced in Listing A.1. From the 360 generated models (120 models times three

approaches), 41 violations occurred. 22 violations resulted from elements placed in the wrong list and violating the enum definition of the element type. For example, events were often misplaced in the task list and gateways in the event list. 18 violations are due to a misspelling of element types, violating the enum definition. For instance, the terms *EndEvent* and *StartEvent* were frequently used in place of the anticipated *EndNoneEvent* and *StartNoneEvent*. A missing required name of an event caused one violation.

## 7 Contribution III: Evaluation

In this chapter, the generated models are evaluated against the ground truth from the data set to measure the generation performance. First, the requirements for the evaluation framework are defined. Existing evaluation methods are examined based on the requirements. Then, a new evaluation framework is introduced and self-evaluated against the requirements. Finally, the framework is applied and the results are presented.

### 7.1 Requirements

This section defines the requirements of the evaluation framework. The evaluation framework should measure the similarity of two models in the BPMN JSON schema (Chapter 5.4), namely the generated model and the ground truth model.

The framework should accurately measure the degree of similarity between the two BPMN models, ensuring a direct proportion and uniform assessment throughout the comparison. This guarantees that the analysis is equitable and consistent, enabling reliable comparisons. Furthermore, it should result in a normalized score between 0 (no similarity) and 1 (identical). It should be easily understandable and intuitive, facilitating quick comprehension and application. Moreover, the framework should allow intuitively summarizing the metric for an overall score and breaking it into more fine-grained scores to gather more insights. In addition, the evaluation process should be designed for efficient computation, ensuring that it can be completed within just a few minutes on a standard personal computer.

Furthermore, the framework should allow syntactic derivations if the meaning of the words is still the same. For example, the two activities “Produce a car” and “Produce

the car” should not be treated as unequal but as almost equal. This thesis refers to the term semantic-awareness for this concept.

Moreover, the metric should include measuring the frequency of the items. For example, if one model has 6 gateways, but the other has only 4, they should not be treated the same.

Additionally, the metric should also consider the control flow of the models, meaning that not only the elements themselves should be included but their connection with each other and how they relate.

To summarize, the evaluation framework should meet the following requirements:

- *RQ1 - Accurate*: Accurately determine the similarity between two BPMN models, ensuring a direct proportion and uniform assessment across the comparison.
- *RQ2 - Normalized*: Result in a score between 0 and 1.
- *RQ3 - Intuitive*: Be easily understandable and intuitive.
- *RQ4 - Granular Insight*: Allow summarizing the metric for an overall score but also breaking it down into a more fine-grained score in an intuitive way.
- *RQ5 - Efficient*: Be able to compute it efficiently on an average computer.
- *RQ6 - Semantic-aware*: Be semantic-aware and allow syntactically derivations.
- *RQ7 - Frequency-aware*: Include measuring the frequency of the items.
- *RQ8 - Control-flow-aware*: Consider the control flow of the models.

## 7.2 Existing evaluation methods

Schoknecht et al. [54, pp. 16–21] examined the most common similarity measures for business process models. They identified four main dimensions: Natural Language, Graph Structure, Behaviour, and Human Estimation.

The Natural Language Dimension focuses on the textual similarity of the models, for example, the label of the elements. The literature distinguishes between syntactical and semantical analyses. Syntactical approaches compare the characters without

understanding the meaning of the words. The most common examples are the String Edit Distance, Word Edit Distance, or Vector Space representations. Semantical approaches, in contrast, compare the texts with understanding their meaning [54, pp. 16–21]. BERT sentence-transformer is a common method for the semantic approach [49]. To meet the requirement *Semantic-aware*, the thesis focuses on BERT as a replicate of the Natural Language Dimension.

The Graph Structure Dimension interprets the model as a graph and examines the connections between the elements, namely the control flow. Minor et al. [42] calculate the similarity by counting the number of common nodes and edges and relating them to the overall number of nodes and edges. The edges represent the control flow of the process model [54, pp. 16–21]. They take an extension of the Sørensen–Dice coefficient for calculating the similarity of sets of edges and nodes [59, 16]. An alternative for calculating the similarity of sets is the Jaccard coefficient [30]. After conducting experiments, the thesis chooses the Sørensen–Dice coefficient over the Jaccard coefficient because it meets the requirement *Intuitive* for the use case better.

Another related approach divides the nodes into different roles [54, pp. 16–21]. For example, Yan et al. [64] uses the roles start, stop, split, join, regular to distinguish nodes. Two nodes are now considered equivalent if the label and the role are similar. The thesis will adapt this approach to match the elements in the used model schema.

A different method for the Graph Structure Dimension is the graph and tree edit distance. The main idea is to count the operations needed to transform one model into the others (for example, removing and adding edges or nodes) [54, pp. 16–21]. However, calculating the graph edit distance is NP-hard and also hard to approximate (APX-hard) [38, 66]. The models from the dataset took multiple hours, contradicting the requirement *Efficient*.

The Behaviour Dimension examines the similarity of the actual behavior of the business process models. One possibility is to simulate all possible execution orders of the elements, also called traces or event logs. Then, the similarity can be calculated based on the longest common subsequent of the traces. Another method is to use the traces to calculate dependencies between the activities and compare these dependencies. Furthermore, causal footprints of the models (for example, preceding or followed by causalities per element pair) can be compared to calculate similarity [54, pp. 16–21]. All these approaches need a behavioral simulation of the

models similar to a process engine. The thesis examined two tools with underlying process engines: pm4py and the SAP Signavio Process Transformation Suite [20, 52]. However, both rely on painting coordinates or Petri-net representations to import the models into the tools, which the generated models do not directly support. Therefore, this thesis excludes the behavior dimension because it seems technically not feasible with the resource constraints of this thesis.

The Human Estimation Dimension uses manual human feedback to estimate the similarity [54, pp. 16–21]. As this process takes time, it does not meet the requirement to be *Efficient*.

## 7.3 Introducing the evaluation framework

In the last chapter, several methods for evaluating the similarity of process models have been described. No single method fulfills the requirement alone. Therefore, this section introduces its own evaluation framework and combines the described methods. First, the model is broken into elements, representing nodes and edges in the Graph Structure Dimension. Then  $dice_{SFA}$ , an extended version of the dice coefficient is introduced, which uses BERT and represents the Natural Language Dimension.

### 7.3.1 Element based breakdown

The first step of the evaluation is to break down the model into elements according to the introduced schema.

Let:

- $TN$  be a multiset of task names.
- $TT$  be a multiset of task types.
- $EN$  be a multiset of event names.
- $ET$  be a multiset of event types.
- $GN$  be a multiset of gateway names.

- $GT$  be a multiset of gateway types.
- $LN$  be a multiset of lane names. A lane name is defined as a tuple  $(p, l)$  where  $p$  is the pool label and  $l$  is the label of the lane.
- $LR$  be a multiset of lanes with reference. A lane with reference is a triple  $(p, l, r)$  where  $p$  is the pool label,  $l$  is the label of the lane, and  $r \in TN \cup EN \cup GT$ .  $GT$  is chosen over  $GN$  because the names of the gateways are optional, while the type is mandatory.
- $SF$  be a multiset of sequence flows. A sequence flow is defined as a triple  $(s, c, t)$  where  $s, t \in TN \cup EN \cup GT \cup LN$  and  $c$  provides a name for the sequence flow.
- $MF$  be a multiset of message flows. A message flow is defined as a triple  $(s, l, t)$  where  $s, t \in TN \cup EN \cup GT \cup LN$  and  $l$  provides a name for the message flow.

The defined multisets can describe a complete process model in the JSON schema format from Chapter 5.4. This thesis refers to them as a joint set of elements:

$$M = \{TN, TT, EN, ET, GN, GT, LN, LR, SF, MF\} \quad (7.1)$$

For a more simple notion, the elements are referred to as  $E_i$ :

$$M = \{E_i : 1 \leq i \leq 10\} \quad (7.2)$$

Consider the following example model in a JSON format:

---

```

1 {
2   "tasks": [
3     {
4       "id": "0",
5       "name": "Produce the car",
6       "type": "UserTask"
7     },
8     {
9       "id": "1",
10      "name": "Test the car",
11      "type": "ServiceTask"

```

```
12     },
13     {
14         "id": "2",
15         "name": "Fix the issue",
16         "type": "UserTask"
17     }
18 ],
19 "events": [
20     {
21         "id": "3",
22         "name": "Order arrived",
23         "type": "StartMessageEvent"
24     },
25     {
26         "id": "4",
27         "name": "Car produced",
28         "type": "EndMessageEvent"
29     }
30 ],
31 "gateways": [
32     {
33         "id": "5",
34         "name": "Detected errors?",
35         "type": "Exclusive"
36     },
37     {
38         "id": "6",
39         "type": "Exclusive"
40     }
41 ],
42 "pools": [
43     {
44         "id": "7",
45         "name": "Car manufacture",
46         "lanes": [
47             {
48                 "id": "8",
49                 "name": "Production",
50                 "elemRefs": [
51                     "0",
52                     "2",
53                     "3",
54                     "6"
55                 ]
56             },
57             {
58                 "id": "9",
59                 "name": "Quality Control",
60                 "elemRefs": [
61                     "1",
62                     "4",
63                     "5"
64                 ]
65             }
66         ]
67     }
```



```

68     },
69     {
70         "id": "10",
71         "name": "Customer",
72         "lanes": []
73     }
74 ],
75 "sequenceFlows": [
76     {
77         "id": "11",
78         "sourceRef": "3",
79         "targetRef": "0"
80     },
81     {
82         "id": "12",
83         "sourceRef": "0",
84         "targetRef": "6"
85     },
86     {
87         "id": "13",
88         "sourceRef": "6",
89         "targetRef": "1"
90     },
91     {
92         "id": "14",
93         "sourceRef": "1",
94         "targetRef": "5"
95     },
96     {
97         "id": "15",
98         "sourceRef": "5",
99         "targetRef": "2",
100        "condition": "no"
101    },
102    {
103        "id": "16",
104        "sourceRef": "5",
105        "targetRef": "4",
106        "condition": "yes"
107    },
108    {
109        "id": "17",
110        "sourceRef": "2",
111        "targetRef": "6"
112    }
113 ],
114 "messageFlows": [
115     {
116         "id": "18",
117         "sourceRef": "10",
118         "targetRef": "3"
119     },
120     {
121         "id": "19",
122         "sourceRef": "4",

```

```

124         "targetRef": "10"
125     }
126 ]
127 }

```

Listing 7.1: JSON example model

That results in the following element sets:

$$EN = ["Order arrived", "Car produced"] \quad (7.3)$$

$$ET = ["StartMessageEvent", "EndMessageEvent"] \quad (7.4)$$

$$TN = ["Produce the car", "Test the car", "Fix the issue"] \quad (7.5)$$

$$TT = ["ServiceTask", "UserTask", "UserTask"] \quad (7.6)$$

$$GN = ["Detected errors ?"] \quad (7.7)$$

$$GT = ["Exclusive", "Exclusive"] \quad (7.8)$$

$$SF = \begin{bmatrix} \text{"Order arrived - Produce the car",} \\ \text{"Produce the car - Exclusive",} \\ \text{"Exclusive - Test the car",} \\ \text{"Test the car - Exclusive",} \\ \text{"Exclusive - no - Fix the issue",} \\ \text{"Exclusive - yes - Car produced",} \\ \text{"Fix the issue - Exclusive"} \end{bmatrix} \quad (7.9)$$

$$MF = ["Customer - Order arrived", "Car produced - Customer"] \quad (7.10)$$

$$LN = \begin{bmatrix} \text{"Car manufacture - Production",} \\ \text{"Car manufacture - Quality Control",} \\ \text{"Customer"} \end{bmatrix} \quad (7.11)$$

$$LR = \begin{bmatrix} \text{"Car manufacture - Production - Produce the car",} \\ \text{"Car manufacture - Production - Fix the issue",} \\ \text{"Car manufacture - Production - Order arrived",} \\ \text{"Car manufacture - Production - Exclusive",} \\ \text{"Car manufacture - Quality Control - Test the car",} \\ \text{"Car manufacture - Quality Control - Car produced",} \\ \text{"Car manufacture - Quality Control - Exclusive"} \end{bmatrix} \quad (7.12)$$

### 7.3.2 Similarity score calculation

In the second step, the similarity scores of the two models are calculated based on the breakdown introduced in Section 7.3.1. The main idea is to calculate the similarity score for each element separately. Let:

- $M_1 = \{E_{i,1} : 1 \leq i \leq 10\}$  be the ground truth model.
- $M_2 = \{E_{i,2} : 1 \leq i \leq 10\}$  be the generated model.

Then, the similarity scores are calculated per element with the  $dice_{SFA}$  metric, an adjustment of the Sørensen–Dice coefficient:

$$sim(E_{i,1}, E_{i,2}) = dice_{SFA}(E_{i,1}, E_{i,2}) \quad (7.13)$$

The original Sørensen–Dice coefficient is defined as [59, 16]:

$$dice(E_{i,1}, E_{i,2}) = \frac{2|E_{i,1} \cap E_{i,2}|}{|E_{i,1}| + |E_{i,2}|} \quad (7.14)$$

To calculate an overall score, the weighted score over all the elements is computed:

$$sim(M_1, M_2) = \frac{\sum_{i=1}^{10} w_i \cdot dice_{SFA}(E_{i,1}, E_{i,2})}{\sum_{i=1}^{10} w_i} \quad \text{with } w_i = |E_{i,1}| + |E_{i,2}| \quad (7.15)$$

Thereby, weights are equal to the cardinal weights of the two sets. This can be adapted if an element is considered more important than others. In this thesis, all elements are treated as equally important. The same procedure is used to calculate aggregated scores for:

- the tasks based on the elements  $TN$  and  $TT$ .
- the events based on the elements  $EN$  and  $ET$ .
- the gateways based on the elements  $GN$  and  $GT$ .
- the flows based on the elements  $SF$  and  $MF$ .
- the lanes based on the elements  $LN$  and  $LR$ .

The  $dice_{SFA}$  is an adjustment of the Sørensen–Dice coefficient (see Algorithm 1) that is semantic- and frequency-aware.  $dice_{SFA}$  takes two lists as an input and applies the two functions *make\_similar\_items\_equal* (for semantic-awareness) and *index\_list* (for frequency-awareness) before calculating the original dice Sørensen–Dice coefficient with the function *dice*.

---

**Algorithm 1** Semantic-and Frequency-Aware Sørensen–Dice coefficient

---

**Require:**  $list1, list2$  (two lists of items)

**Ensure:** Semantic-and frequency-aware Sørensen–Dice coefficient of the two lists

```

1: function DICE_SFA( $list1, list2$ )
2:   ( $list1, list2$ )  $\leftarrow$  MAKE_SIMILAR_ITEMS_EQUAL( $list1, list2$ )
3:   ( $list1, list2$ )  $\leftarrow$  (INDEX_LIST( $list1$ ), INDEX_LIST( $list2$ ))
4:   return DICE( $list1, list2$ )
5: end function

```

---

The function *make\_similar\_items\_equal* (Algorithm 2) semantically matches items from two lists, even when syntactically dissimilar, enhancing the metric’s *semantic-awareness*. It employs a BERT sentence transformer for item vectorization and considers items similar if their cosine similarity exceeds a 0.7 threshold - determined experimentally, though adjustable for different contexts. Items can be matched only once, prioritizing matches with the highest similarity first.

---

**Algorithm 2** Make Similar Items Equal

---

**Require:**  $list1, list2$  (two lists of items), *similarity\_func* (computes similarity of two items), *threshold* (minimum similarity score for items to be considered equal)

**Ensure:** Modified  $list1, list2$  with similar items replaced by equal items

```

1: function MAKE_SIMILAR_ITEMS_EQUAL( $list1, list2, similarity\_func, threshold$ )
2:   Compute comparison_matrix for  $list1 \times list2$  using similarity_func
3:   Sort comparison_matrix by scores in descending order
4:   Initialize matched indices lists: matched_list1, matched_list2
5:   for each pair ( $i1, i2$ ) and score in sorted comparison_matrix do
6:     if  $i1$  in matched_list1 or  $i2$  in matched_list2 then Skip to next pair
7:     else if score  $\geq$  threshold then
8:       Set  $list1[i1]$  to value of  $list2[i2]$  to make them equal
9:       Add  $i1$  to matched_list1 and  $i2$  to matched_list2
10:    end if
11:  end for
12:  return  $list1, list2$ 
13: end function

```

---

The *index\_list* function appends indices to each item, as detailed in Algorithm 3. This adaptation is necessary because the Sørensen–Dice coefficient operates solely on sets and disregards the frequency of items. The indexing process effectively converts multisets to sets while preserving all entries, making it *frequency-aware*.

---

**Algorithm 3** Index List

---

**Require:** *list* (a list of items)

**Ensure:** *indexed\_list* (each item in the list made unique by appending an index)

```

1: function INDEX_LIST(list)
2:   Initialize count as empty dictionary, indexed_list as empty list
3:   for each item in list do
4:     index  $\leftarrow$  count.get(item, 0)
5:     indexed_list.append(item + str(index))
6:     count[item]  $\leftarrow$  index + 1
7:   end for
8:   return indexed_list
9: end function

```

---

The function *dice* calculates the original Sørensen–Dice coefficient for the two lists, resulting in a score between 0 and 1 (see Algorithm 4).

---

**Algorithm 4** Sørensen–Dice Coefficient Calculation

---

**Require:** *list1*, *list2* (two lists of items)

**Ensure:** Sørensen–Dice coefficient of the lists in a range between 0 and 1

```

1: function DICE(list1, list2)
2:   intersection  $\leftarrow$  len(set(list1)  $\cap$  set(list2))
3:   union  $\leftarrow$  len(set(list1)) + len(set(list2))
4:   if union = 0 then return 1
5:   else
6:     return 2 · intersection / union
7:   end if
8: end function

```

---

Consider the following example for tasks:

$$TN_1 = [\text{"Produce the car"}, \text{"Test the car"}, \text{"Fix the issue"}, \text{"Ship the car"}] \quad (7.16)$$

$$TN_2 = [\text{"Produce a car"}, \text{"Wash the car"}, \text{"Fix the error"}, \text{"Order the car"}] \quad (7.17)$$

$$TT_1 = [\text{"ServiceTask"}, \text{"UserTask"}, \text{"UserTask"}, \text{"UserTask"}] \quad (7.18)$$

$$TT_2 = [\text{"ServiceTask"}, \text{"ServiceTask"}, \text{"ServiceTask"}, \text{"ServiceTask"}] \quad (7.19)$$

That results in these two calculations:

$$\text{sim}(TN_1, TN_2) = \text{dice}_{SFA}(TN_1, TN_2) \quad (7.20)$$

$$\text{sim}(TT_1, TT_2) = \text{dice}_{SFA}(TT_1, TT_2) \quad (7.21)$$

For the task names, the *make\_similar\_items\_equal* function finds two semantic matches even if the syntax of the words is different. "Produce the car" is similar to "Produce a car" with a similarity score of 0.98, above the threshold. "Fix the issue" and "Fix the error" are the second match with 0.88 similarity above the threshold. For the task types, there is one exact match for the Service Task from  $TT_1$  and one of the Service Tasks from  $TT_2$ . Applying the function results in:

$$TN_1 = [\text{"Produce a car"}, \text{"Test the car"}, \text{"Fix the error"}, \text{"Ship the car"}] \quad (7.22)$$

$$TN_2 = [\text{"Produce a car"}, \text{"Wash the car"}, \text{"Fix the error"}, \text{"Order the car"}] \quad (7.23)$$

$$TT_1 = [\text{"ServiceTask"}, \text{"UserTask"}, \text{"UserTask"}, \text{"UserTask"}] \quad (7.24)$$

$$TT_2 = [\text{"ServiceTask"}, \text{"ServiceTask"}, \text{"ServiceTask"}, \text{"ServiceTask"}] \quad (7.25)$$

Next, the *indexed\_list* function transforms the multisets into sets with indexes. This is important for the task types in the example because the instances of *UserTasks* and *ServiceTasks* are treated as multiple entries in the final set. The rest of the items get only a 0 appended, which does not change anything:

$$TN_1 = [\text{"Produce a car0"}, \text{"Test the car0"}, \text{"Fix the error0"}, \text{"Ship the car0"}] \quad (7.26)$$

$$TN_2 = \left[ \begin{array}{l} \text{"Produce a car0"}, \text{"Wash the car0"}, \text{"Fix the error0"}, \\ \text{"Order the car0"} \end{array} \right] \quad (7.27)$$

$$TT_1 = [\text{"ServiceTask0"}, \text{"UserTask0"}, \text{"UserTask1"}, \text{"UserTask2"}] \quad (7.28)$$

$$TT_2 = [\text{"ServiceTask0"}, \text{"ServiceTask1"}, \text{"ServiceTask2"}, \text{"ServiceTask3"}] \quad (7.29)$$

Finally, the Sørensen–Dice coefficient is computed:

$$\text{Score}_{TN} = 2 \cdot (2/8) = 0.5 \quad (7.30)$$

$$\text{Score}_{TT} = 2 \cdot (1/8) = 0.25 \quad (7.31)$$

The aggregated score for tasks is calculated based on the weighted sum of  $TN$  and  $TT$ :

$$w_{TN} = (|TN_1| + |TN_2|) = 4 + 4 = 8 \quad (7.32)$$

$$w_{TT} = (|TT_1| + |TT_2|) = 4 + 4 = 8 \quad (7.33)$$

$$Score_{Tasks} = \frac{w_{TN} \cdot Score_{TN} + w_{TT} \cdot Score_{TT}}{w_{TN} + w_{TT}} \quad (7.34)$$

$$= \frac{8 \cdot 0.5 + 8 \cdot 0.25}{8 + 8} = 0.375 \quad (7.35)$$

The same procedure is followed to calculate the scores for EN, ET, an aggregated score of EN and ET, GN, GT, an aggregated score of GN and GT, LN, LR, an aggregated score of LN and LR, SF, MF, an aggregated score of SF and MF and the overall score which aggregates all elements, namely EN, ET, GN, GT, LN, LR, SF and MF. Figure 7.1 illustrates the different calculated scores and how they are connected to each other.

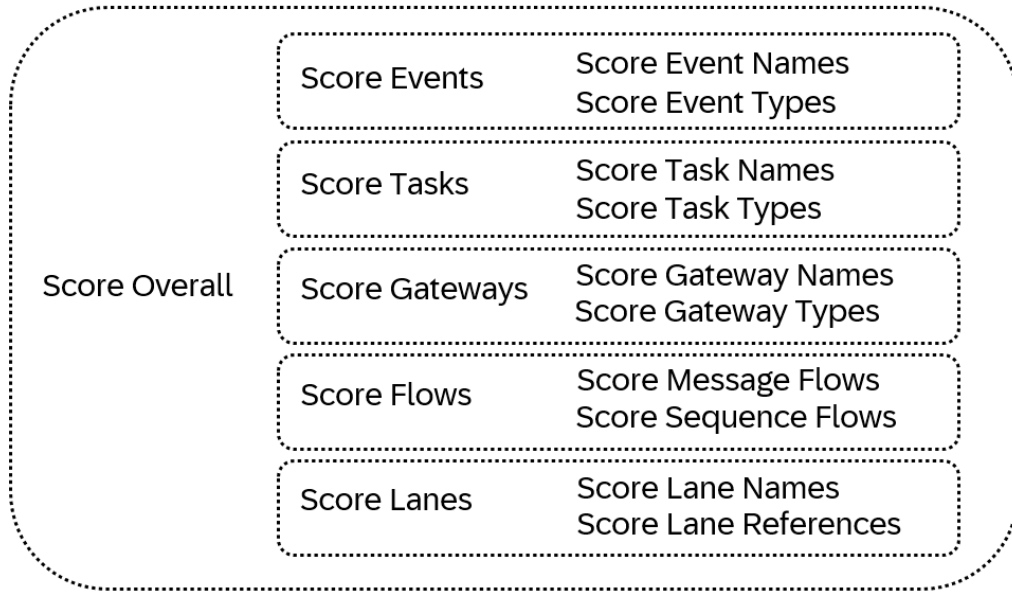


Figure 7.1: Overview of the different element scores and how they are aggregated.

### 7.3.3 Evaluation of the evaluation framework

This section evaluates the introduced evaluation framework to the requirements of section 7.1:

- *RQ1 - Accurate*: The methodology assigns a score of 0 for completely dissimilar BPMN models and 1 for identical models, with intermediate values for partial similarities. While proving absolute accuracy and a perfectly direct proportionality mathematically is challenging, empirical evidence from tests supports the framework's high precision and tendency towards a linear relationship in measuring similarity. However, integrating BERT for similarity assessment with a fixed threshold of 0.7 introduces a limitation, potentially deviating from a strictly linear model. Additionally, the issue of set overlap may further contribute to deviations from linearity. Despite these hurdles, the empirical results have consistently demonstrated intuitive and precise outcomes, aligning well with the requirement.
- *RQ2 - Normalized*: The requirement is satisfied due to the framework generating scores ranging from 0 to 1, as previously explained.
- *RQ3 - Intuitive*: The requirement is fulfilled as it has been consistently deemed intuitive across numerous tests, as outlined above.
- *RQ4 - Granular Insight*: The framework fulfills this requirement with fine-grained element scores, aggregated scores, and an overall score.
- *RQ5 - Efficient*: The metric can be computed on average for one model pair within seconds and therefore meets the requirement.
- *RQ6 - Semantic-aware*: The requirement is met using BERT similarity matching.
- *RQ7 - Frequency-aware*: Indexing the list fulfills the requirement.
- *RQ8 - Control-flow-aware*: The requirement is satisfied by evaluating message and sequence flows.

As shown, the requirements are mostly fulfilled. The challenge lies in the accuracy because although experiments have shown accurate results, it can not be mathematically proven that the score is always fully accurate.



## 7.4 Results

The evaluation framework has been applied to the 360 generated models and their ground truths (120 per approach). The mean is calculated for each approach to get an average score for the zero-shot, one-shot, and few-shot approaches.

Figure 7.2 visually represents the result of the conducted evaluation, while Table 7.1 shows the exact results. The overall score for one-shot is 87 percent, for few-shot 86 percent while for zero-shot only 81 percent. For all approaches, the score for task names, task types, and event types is higher than the overall score, while the event names are slightly below it. There is a big drop in similarity for gateway names (60 - 28 percent only), while gateway types are similar to the overall score. The sequence flow is slightly below the overall score (75 - 77 percent), and the message flow drops marginally more (68 - 72 percent). The lanes are around the overall score again.

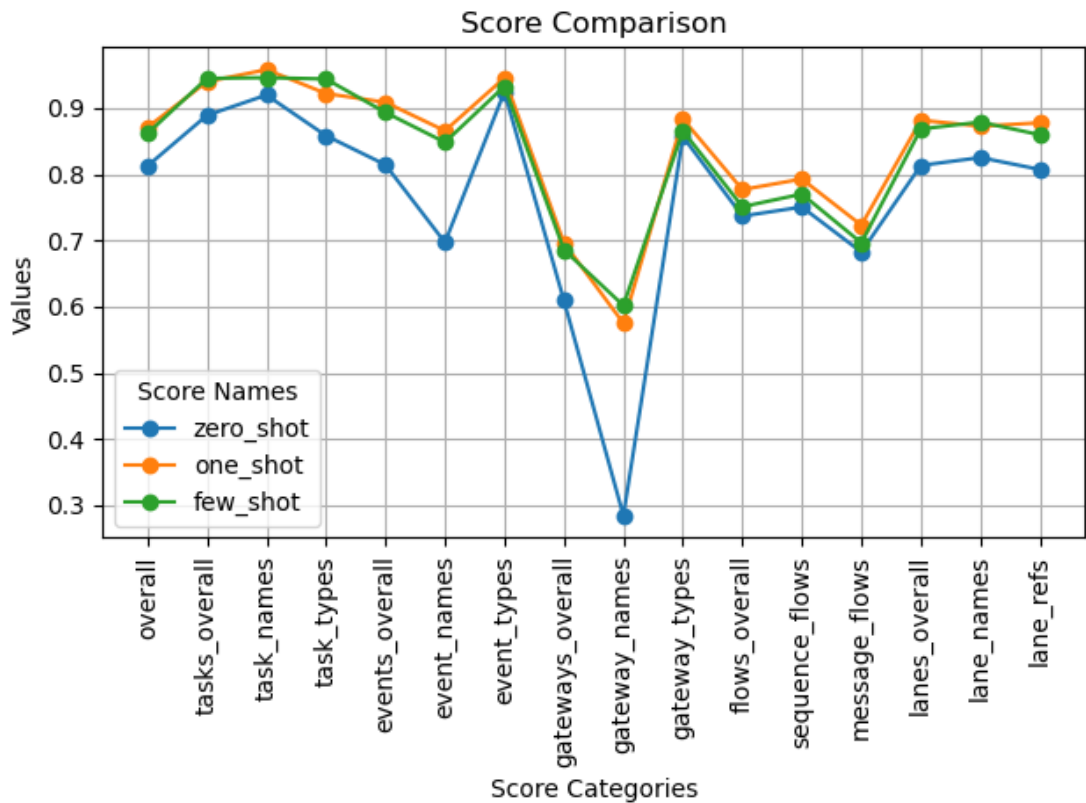


Figure 7.2: Average scores for zero-shot, one-shot and few-shot

Score Name	Zero Shot	One Shot	Few Shot
Overall	0.812654	0.871079	0.861744
Tasks Overall	0.889146	0.939833	0.944912
Task Names	0.919780	0.958201	0.946046
Task Types	0.858431	0.922100	0.944000
Events Overall	0.814431	0.908772	0.893474
Event Names	0.697021	0.865807	0.848409
Event Types	0.924640	0.945658	0.932955
Gateways Overall	0.609449	0.696482	0.686032
Gateway Names	0.284444	0.575058	0.601764
Gateway Types	0.856959	0.883706	0.865670
Flows Overall	0.736907	0.776942	0.751045
Sequence Flows	0.750651	0.792892	0.770219
Message Flows	0.682600	0.723168	0.696521
Lanes Overall	0.813442	0.881780	0.868453
Lane Names	0.825298	0.872698	0.878962
Lane Refs	0.807339	0.877842	0.859720

Table 7.1: Comparison of average zero-shot, one-shot, and few-shot scores.

When comparing zero, one, and few-shot, the results show that the performance of one-shot and few-shot is very similar and that they both outperformed zero-shot significantly. One-shot and few-shot have an average difference below one percent, while their difference to few-shot is around 7 - 8 percent (see Table 7.2). The difference increases when the scores get worse, for example, for the gateway or event names.

The standard deviation is pictured in Figure 7.3. It shows a high correlation between the score and the deviation. If the score is low, for example, for gateway names and message flows, the deviation is high. This can also be seen in the standard deviation for the lower score of zero-shot compared to one and few-shot.

Score Name	Zero - One	Zero - Few	One - Few
Overall	-0.058424	-0.049089	0.009335
Tasks Overall	-0.050686	-0.055765	-0.005079
Task Names	-0.038422	-0.026267	0.012155
Task Types	-0.063669	-0.085570	-0.021900
Events Overall	-0.094340	-0.079043	0.015298
Event Names	-0.168786	-0.151388	0.017398
Event Types	-0.021018	-0.008315	0.012704
Gateways Overall	-0.087034	-0.076583	0.010451
Gateway Names	-0.290613	-0.317320	-0.026706
Gateway Types	-0.026747	-0.008711	0.018036
Flows Overall	-0.040035	-0.014138	0.025897
Sequence Flows	-0.042242	-0.019568	0.022673
Message Flows	-0.040569	-0.013921	0.026647
Lanes Overall	-0.068338	-0.055011	0.013327
Lane Names	-0.047401	-0.053664	-0.006263
Lane Refs	-0.070503	-0.052381	0.018122
Mean	-0.075552	-0.066671	0.008881

Table 7.2: Pairwise differences per approach. The *Zero - One* row is calculated by taking the score from the zero-shot method and subtracting the score from the one-shot method. *Zero - Few* and *One - Few* works, respectively.

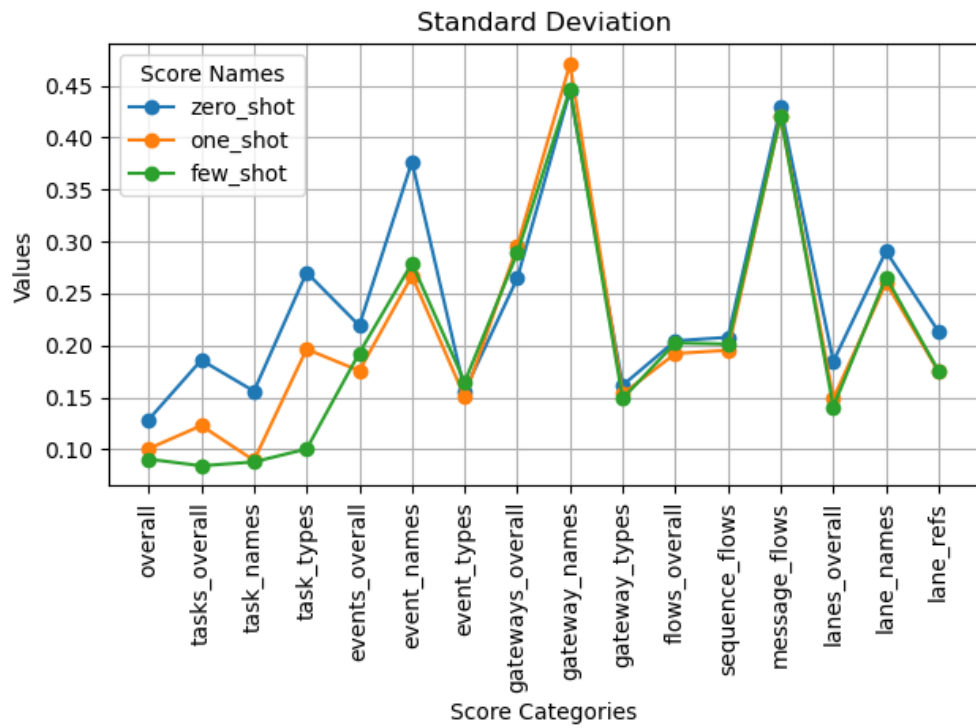


Figure 7.3: Standard deviations for zero-shot, one-shot and few-shot scores

## 8 Discussion

### 8.1 Derived implications

The results, with an overall similarity score of 87 percent for the one-shot approach, show that creating process models from multimodal documents using generative AI techniques is feasible. It can be derived that the document analysis to create as-is models can be partly automated with humans in the loop. It shows that a multimodal LLM approach is suitable and could be used instead of processing text and images separately. That would bring multiple advantages. With the multimodal approach, only one model is needed, which can also be used for other tasks, reducing development and maintenance efforts. Furthermore, the multimodal LLM does not need effortful training but can be used off the shelf. It allows more flexibility, for example it is not tailored to one specific type of image as the more traditional approaches are.

Furthermore, in the future, models and approaches will probably improve and make the use case even more feasible. The capability *function calling* of LLMs can improve the results when it becomes available. The use of newer models like Gemini 1.5 could improve performance as well. In addition, optimizing the prompt further could bring improvements. Overall, there seems to be potential for multimodal LLMs to be applied for automating document analysis in the future. The thesis provides a stable benchmarking environment for these future developments.

Moreover, it can be derived that LLMs work well with JSON schemas that contain element references. In addition, the results indicate that one-shot prompting is sufficient in the use case, and more examples do not necessarily lead to better results.

From the element-wise scores, it can be derived that LLMs excel in extracting tasks and events while having difficulties with flows, particularly message flows.

That is reasonable, as the concept of flows is more difficult than tasks and events. Furthermore, message flows are less common and more advanced than sequence flows, explaining the difference in the score. In addition, LLMs seem to have difficulties with the gateway names. This can be explained by the fact that gateway names are optional and often placed differently in the diagram. Figure 8.1 and 8.2 show examples where a gateway name is placed above and another is placed on the right.

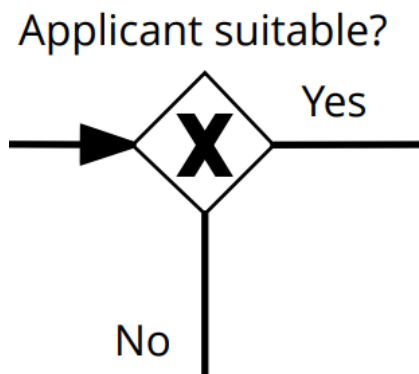


Figure 8.1: Gateway with name on the top

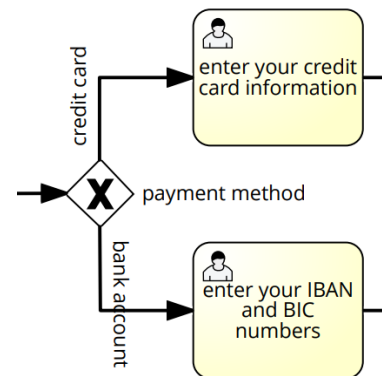


Figure 8.2: Gateway with name on the right

## 8.2 Limitations of the thesis

The thesis shows several limitations. While the dataset consists of multimodal documents, the text and image distribution is imperfect. The diagram in the documentation contains all the necessary information to meet the ground truth. It would be better to have some information only available in the images and others only available in the texts to evaluate the multimodal capabilities better. The LLM still reads the whole document containing multiple pages of text with additional descriptions, making the limitation acceptable.

Furthermore, the thesis limited the generation process to using one model only. That makes it difficult to predict if other LLMs would behave similarly, for instance, if they have similar difficulties regarding gateway types and flows. Moreover, the thesis is limited by the current technology, which moves very fast and could be already outdated when reading this thesis. That GPT-4V was published after the start of the thesis, and Gemini 1.5 near the end of the thesis supports that assumption. The

thesis created a stable benchmarking environment to overcome this limitation and allow future measurements of improved technology.

In addition, the thesis was resource-constrained, making it impossible to investigate the approaches of chain-of-thought prompting and fine-tuning. The used prompts can probably also be improved with more effort.

Last but not least, the evaluation part can not be mathematically proven to be always fully accurate. Integrating BERT for similarity assessment with a fixed threshold of 0.7 introduces a limitation, potentially deviating from a strictly linear model. Additionally, the issue of set overlap may further contribute to deviations from linearity. That results in some elements being weighted more than others, although the thesis could not argue which elements are more important and should gain more weight. Despite these hurdles, the empirical results have consistently demonstrated intuitive and precise outcomes, ensuring a direct proportion and uniform assessment across the comparison.

# 9 Conclusion

## 9.1 Summary

The thesis presented the feasibility and suitability of generative AI methods in the context of generating process models from multimodal documents containing images and text. Furthermore, a benchmarking system was introduced to offer a framework for future evaluations in the fast-changing environment.

For this purpose, first, the theoretical underpinnings of the Business Process Management and Generative AI domains were established. Next, the requirements of a suitable dataset were defined to provide a common objective ground. Existing datasets were evaluated against the requirements, and a new dataset with multimodal process documentation was created to meet the requirements.

Afterward, current multimodal models and approaches were reviewed. GPT-4V was chosen as the best-performing model. Zero-, one-, few-shot and chain of thought prompting were chosen as the most promising approaches and conceptually examined. Then, process models were generated from the process documentation of the dataset with the zero-, one-, and few-shot approaches.

After that, the requirements of an evaluation framework were defined for measuring the carried-out generation performance and to enable objective future evaluations. Existing evaluation methods were examined, and a new evaluation framework was introduced to meet the requirements. The framework measures the similarity of two process models by breaking the models down into elements and calculating the  $dice_{SFA}$  metric.  $dice_{SFA}$  is a modified Sørensen–Dice coefficient incorporating semantic and frequency awareness. The framework was then applied to the generated models and the ground truths models from the dataset to evaluate the generation performance.



The results showed, with an 87 percent overall similarity score of the one-shot method, that the approach is feasible and suitable.

### 9.2 Outlook

Future work could benchmark the usage of text-only and image-only models against the dataset to compare the performance to traditional approaches. Furthermore, the distribution of text and images across datasets and model usage should be investigated. To achieve that, the ground truth could be extended by further information only included in the text, or the process documentation could exclude some elements in the diagram. Irrespective of this, the ground truth could be extended to capture more BPMN elements already conceptually outlined in the thesis.

In addition, further datasets could be created that contain a more diverse range of documentation, for example, other types of diagrams, pictures, and textual descriptions.

Moreover, new models and approaches can be benchmarked with the dataset created in this thesis. Gemini 1.5 Pro is already available; more models are soon to come. When resources permit, exploring chain-of-thought prompting, function calling, and fine-tuning seems promising.

# A Appendix

---

```
1 bpmn_schema = {
2   "$schema": "https://json-schema.org/draft/2020-12/schema",
3   "title": "BPMN Schema",
4   "type": "object",
5   "properties": {
6     "tasks": {
7       "type": "array",
8       "items": {
9         "$ref": "#/$defs/task"
10      },
11    },
12    "events": {
13      "type": "array",
14      "items": {
15        "$ref": "#/$defs/event"
16      }
17    },
18    "gateways": {
19      "type": "array",
20      "items": {
21        "$ref": "#/$defs/gateway"
22      }
23    },
24    "pools": {
25      "type": "array",
26      "items": {
27        "$ref": "#/$defs/pool"
28      }
29    },
30    "sequenceFlows": {
31      "type": "array",
32      "items": {
33        "$ref": "#/$defs/sequenceFlow"
34      }
35    },
36    "messageFlows": {
37      "type": "array",
38      "items": {
39        "$ref": "#/$defs/messageFlow"
40      }
41    },
42  },
43  "required": ["tasks", "events", "gateways", "pools", "sequenceFlows", "messageFlows"],
```

---

```

44 "$defs": {
45   "task": {
46     "type": "object",
47     "properties": {
48       "id": { "type": "string" },
49       "name": { "type": "string" },
50       "type": { "enum": ["Task", "Manual", "CollapsedSubprocess", "
        Service", "User", "Send", "Business Rule", "Receive", "Script
        ", "CollapsedEventSubprocess"] }
51     },
52     "required": ["id", "name", "type"],
53     "description": "A unit of work, the job to be performed",
54   },
55   "event": {
56     "type": "object",
57     "properties": {
58       "id": { "type": "string" },
59       "name": { "type": "string" },
60       "type": { "enum": ["StartNoneEvent", "EndNoneEvent", "
        IntermediateMessageEventCatching", "IntermediateTimerEvent",
        "StartMessageEvent", "IntermediateMessageEventThrowing", "
        EndMessageEvent", "IntermediateConditionalEvent", "
        IntermediateEscalationEvent", "IntermediateErrorEvent", "
        EndEscalationEvent", "EndCancelEvent", "
        IntermediateMultipleEventThrowing", "
        IntermediateLinkEventThrowing", "StartTimerEvent", "
        IntermediateCancelEvent", "IntermediateLinkEventCatching"] }
61     },
62     "required": ["id", "name", "type"]
63   },
64   "gateway": {
65     "type": "object",
66     "properties": {
67       "id": { "type": "string" },
68       "name": { "type": "string" },
69       "type": { "enum": ["Exclusive", "Parallel", "Eventbased", "
        Inclusive"] }
70     },
71     "required": ["id", "type"]
72   },
73   "pool": {
74     "type": "object",
75     "properties": {
76       "id": { "type": "string" },
77       "name": { "type": "string" },
78       "lanes": {
79         "type": "array",
80         "items": {
81           "$ref": "#/$defs/lane"
82         }
83       }
84     },
85     "required": ["id", "name", "lanes"],
86     "description": "Organization or role who performs the tasks",
87   },
88   "lane": {

```

```
89     "type": "object",
90     "properties": {
91       "id": { "type": "string" },
92       "name": { "type": "string" },
93       "elemRefs": {
94         "type": "array",
95         "items": { "type": "string" }
96       }
97     },
98   },
99   "sequenceFlow": {
100     "type": "object",
101     "properties": {
102       "id": { "type": "string" },
103       "sourceRef": { "type": "string" },
104       "targetRef": { "type": "string" },
105       "condition": { "type": "string" },
106     },
107     "required": ["id", "sourceRef", "targetRef"],
108     "description": "Defines the execution order of activities.
109                     Activities, events and gateways within a pool must be connected
110                     with sequence flow. They can not stand alone. Pictured as an
111                     solid arrow",
112   },
113   "messageFlow": {
114     "type": "object",
115     "properties": {
116       "id": { "type": "string" },
117       "sourceRef": { "type": "string" },
118       "targetRef": { "type": "string" },
119       "label": { "type": "string" }
120     },
121     "required": ["id", "sourceRef", "targetRef"],
122     "description": "Messages flow between different pools. Pictured as
123                     a an dotted arrow",
124   }
125 }
```

---

Listing A.1: JSON Ground Truth Schema

The created dataset and source code are publicly available at <https://github.com/SAP-samples/multimodal-generative-ai-for-bpm>.

# Bibliography

- [1] Thomas Allweyer. *BPMN 2.0: Introduction to the Standard for Business Process Modeling*. 2nd. BOD – Books on Demand GmbH, 2016. ISBN: 978-3-7412-1978-8.
- [2] Thimira Amaratunga. *Understanding Large Language Models*. Berkeley, CA: Apress, 2023. ISBN: 979-8-8688-0016-0. DOI: 10.1007/979-8-8688-0017-7.
- [3] Alessandro Antinori et al. “BPMN-Redrawer: From Images to BPMN Models”. In: *Proceedings of the Best Dissertation Award, Doctoral Consortium, and Demonstration & Resources Track at BPM 2022 co-located with 20th International Conference on Business Process Management (BPM 2022), Münster, Germany, September 11th to 16th, 2022*. Ed. by Christian Janiesch et al. Vol. 3216. CEUR Workshop Proceedings. CEUR-WS.org, 2022, pp. 107–111. URL: [https://ceur-ws.org/Vol-3216/paper\\\_246.pdf](https://ceur-ws.org/Vol-3216/paper\_246.pdf).
- [4] Amin Beheshti et al. “ProcessGPT: Transforming Business Process Management with Generative Artificial Intelligence”. In: *IEEE International Conference on Web Services, ICWS 2023, Chicago, IL, USA, July 2-8, 2023*. Ed. by Claudio A. Ardagna et al. IEEE, 2023, pp. 731–739. DOI: 10.1109/ICWS60048.2023.00099. URL: <https://doi.org/10.1109/ICWS60048.2023.00099>.
- [5] Patrizio Bellan, Mauro Dragoni, and Chiara Ghidini. “Extracting Business Process Entities and Relations from Text Using Pre-trained Language Models and In-Context Learning”. In: *Enterprise Design, Operations, and Computing: 26th International Conference, EDOC 2022, Bozen-Bolzano, Italy, October 3-7, 2022, Proceedings*. Ed. by João Paulo A. Almeida et al. Vol. 13585. Lecture Notes in Computer Science. Cham: Springer, 2022, pp. 182–199. ISBN: 978-3-031-17603-6. DOI: 10.1007/978-3-031-17604-3\_11.

- [6] Patrizio Bellan, Mauro Dragoni, and Chiara Ghidini. “Leveraging pre-trained language models for conversational information seeking from text”. In: *CoRR* abs/2204.03542 (2022). DOI: 10.48550/ARXIV.2204.03542. arXiv: 2204.03542. URL: <https://doi.org/10.48550/arXiv.2204.03542>.
- [7] Patrizio Bellan et al. “PET: An Annotated Dataset for Process Extraction from Natural Language Text Tasks”. In: *BUSINESS PROCESS MANAGEMENT WORKSHOPS*. [S.l.]: SPRINGER INTERNATIONAL PU, 2023, pp. 315–321. ISBN: 978-3-031-25383-6. DOI: 10.1007/978-3-031-25383-6\_23. URL: [https://link.springer.com/chapter/10.1007/978-3-031-25383-6\\_23](https://link.springer.com/chapter/10.1007/978-3-031-25383-6_23).
- [8] Alessandro Berti and Mahnaz Sadat Qafari. *Leveraging Large Language Models (LLMs) for Process Mining (Technical Report)*. July 24, 2023. URL: <http://arxiv.org/pdf/2307.12701.pdf>.
- [9] Alessandro Berti, Daniel Schuster, and Wil M. P. van der Aalst. “Abstractions, Scenarios, and Prompt Definitions for Process Mining with LLMs: A Case Study”. In: *BUSINESS PROCESS MANAGEMENT WORKSHOPS*. Ed. by Jochen de Weerd and Luise Pufahl. Vol. 492. Lecture Notes in Business Information Processing. [S.l.]: Springer, 2024, pp. 427–439. ISBN: 978-3-031-50973-5. DOI: 10.1007/978-3-031-50974-2\_32.
- [10] Katharina Brenning et al. “Text-Aware Predictive Process Monitoring of Knowledge-Intensive Processes: Does Control Flow Matter?” In: *BUSINESS PROCESS MANAGEMENT WORKSHOPS*. Ed. by Jochen de Weerd and Luise Pufahl. Vol. 492. Lecture Notes in Business Information Processing. [S.l.]: Springer, 2024, pp. 440–452. ISBN: 978-3-031-50973-5. DOI: 10.1007/978-3-031-50974-2\_33.
- [11] Tom B. Brown et al. “Language Models are Few-Shot Learners”. In: *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual*. Ed. by Hugo Larochelle et al. 2020. URL: <https://proceedings.neurips.cc/paper/2020/hash/1457c0d6bfc4967418bfb8ac142f64a-Abstract.html>.
- [12] Lena Cabrera et al. “Text-Aware Predictive Process Monitoring with Contextualized Word Embeddings”. In: *BUSINESS PROCESS MANAGEMENT*

- WORKSHOPS*. Ed. by Cristina Cabanillas, Niels Frederik Garmann-Johnsen, and Agnes Koschmider. Vol. 460. Lecture Notes in Business Information Processing. CHAM: Springer, 2023, pp. 303–314. ISBN: 978-3-031-25382-9. DOI: 10.1007/978-3-031-25383-6\_22.
- [13] Ivan Compagnucci et al. “Trends on the Usage of BPMN 2.0 from Publicly Available Repositories”. In: *Perspectives in Business Informatics Research - 20th International Conference on Business Informatics Research, BIR 2021, Vienna, Austria, September 22-24, 2021, Proceedings*. Ed. by Robert Andrei Buchmann et al. Vol. 430. Lecture Notes in Business Information Processing. Springer, 2021, pp. 84–99. DOI: 10.1007/978-3-030-87205-2\_6. URL: [https://doi.org/10.1007/978-3-030-87205-2\\_6](https://doi.org/10.1007/978-3-030-87205-2_6).
- [14] Michael Desmond et al. *A No-Code Low-Code Paradigm for Authoring Business Automations Using Natural Language*. July 15, 2022. URL: <http://arxiv.org/pdf/2207.10648.pdf>.
- [15] Jacob Devlin et al. “BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding”. In: *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL-HLT 2019, Minneapolis, MN, USA, June 2-7, 2019, Volume 1 (Long and Short Papers)*. Ed. by Jill Burstein, Christy Doran, and Tamar Solorio. Association for Computational Linguistics, 2019, pp. 4171–4186. DOI: 10.18653/V1/N19-1423. URL: <https://doi.org/10.18653/v1/n19-1423>.
- [16] Lee R. Dice. “Measures of the Amount of Ecologic Association Between Species”. In: *Ecology* 26.3 (1945), pp. 297–302. ISSN: 00129658. DOI: 10.2307/1932409. URL: <http://www.jstor.org/stable/1932409>.
- [17] Marlon Dumas et al. “AI-augmented Business Process Management Systems: A Research Manifesto”. In: *ACM Transactions on Management Information Systems* 14.1 (2023), pp. 1–19. ISSN: 2158-656X. DOI: 10.1145/3576047.
- [18] Marlon Dumas et al. *Fundamentals of Business Process Management*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2018. ISBN: 978-3-662-56508-7. DOI: 10.1007/978-3-662-56509-4.
- [19] Dirk Fahland et al. *How well can large language models explain business processes?* Jan. 23, 2024. URL: <http://arxiv.org/pdf/2401.12846.pdf>.

- [20] Fraunhofer FIT. *pm4py API Reference — pm4py 2.7.9 documentation*. 2024-01-29. URL: <https://pm4py.fit.fraunhofer.de/static/assets/api/2.7.9/api.html#input-pm4py-read> (visited on 02/22/2024).
- [21] Fabian Friedrich, Jan Mendling, and Frank Puhlmann. “Process Model Generation from Natural Language Text”. In: *Advanced information systems engineering*. LNCS sublibrary. SL 3, Information systems and applications, incl. internet/web, and HCI. Heidelberg and New York: Springer, 2011, pp. 482–496. ISBN: 978-3-642-21640-4. DOI: 10.1007/978-3-642-21640-4\_36. URL: [https://link.springer.com/chapter/10.1007/978-3-642-21640-4\\_36](https://link.springer.com/chapter/10.1007/978-3-642-21640-4_36).
- [22] Chaoyou Fu et al. *MME: A Comprehensive Evaluation Benchmark for Multimodal Large Language Models*. June 23, 2023. URL: <https://arxiv.org/pdf/2306.13394.pdf> (visited on 07/11/2023).
- [23] Neelamadhav Gantayat et al. “Towards Creating Business Process Models from Images”. In: *Service-Oriented Computing*. Lecture Notes in Computer Science. Cham: Springer International Publishing, 2018, pp. 100–108. ISBN: 978-3-030-03596-9. DOI: 10.1007/978-3-030-03596-9\_7. URL: [https://link.springer.com/chapter/10.1007/978-3-030-03596-9\\_7](https://link.springer.com/chapter/10.1007/978-3-030-03596-9_7).
- [24] GitHub. *camunda/bpmn-for-research: A collection of BPMN diagrams that can be used for research*. 2024-02-20. URL: <https://github.com/camunda/bpmn-for-research> (visited on 02/20/2024).
- [25] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep learning*. Adaptive computation and machine learning. Cambridge, Massachusetts: The MIT Press, 2016. ISBN: 9780262337373.
- [26] Google DeepMind. *Gemini: A Family of Highly Capable Multimodal Models*. 2024. URL: [https://storage.googleapis.com/deepmind-media/gemini/gemini\\_1\\_report.pdf](https://storage.googleapis.com/deepmind-media/gemini/gemini_1_report.pdf) (visited on 02/19/2024).
- [27] Google. *Gemini API - Function calling*. URL: [https://ai.google.dev/docs/function\\_calling](https://ai.google.dev/docs/function_calling) (visited on 02/19/2024).
- [28] Michael Grohs et al. “Large Language Models Can Accomplish Business Process Management Tasks”. In: *BUSINESS PROCESS MANAGEMENT WORKSHOPS*. Ed. by Jochen de Weerd and Luise Pufahl. Vol. 492. Lecture



- Notes in Business Information Processing. [S.l.]: Springer, 2024, pp. 453–465. ISBN: 978-3-031-50973-5. DOI: 10.1007/978-3-031-50974-2\_34.
- [29] Han van der Aa et al. “Challenges and Opportunities of Applying Natural Language Processing in Business Process Management”. In: *Proceedings of the 27th International Conference on Computational Linguistics* (2018), pp. 2791–2801. URL: <https://aclanthology.org/C18-1236/>.
- [30] Paul Jaccard. “THE DISTRIBUTION OF THE FLORA IN THE ALPINE ZONE. 1”. In: *New Phytologist* 11.2 (1912), pp. 37–50. ISSN: 1469-8137. DOI: 10.1111/j.1469-8137.1912.tb05611.x.
- [31] Jaclyn Konzelmann and Wiktor Gworek – Google Labs. *Gemini 1.5 - Our next-generation model, now available for Private Preview in Google AI Studio - Google for Developers: February 15, 2024*. 2024-02-23. URL: <https://developers.googleblog.com/2024/02/gemini-15-available-for-private-preview-in-google-ai-studio.html> (visited on 02/23/2024).
- [32] Urszula Jessen, Michal Sroka, and Dirk Fahland. *Chit-Chat or Deep Talk: Prompt Engineering for Process Mining*. July 19, 2023. URL: <http://arxiv.org/pdf/2307.09909.pdf>.
- [33] Timotheus Kampik et al. *Large Process Models: Business Process Management in the Age of Generative AI*. 2023.
- [34] Qingyi Kang, Yong Yu, and Fei Dai. “Automatic Recognition of Business Process Images”. In: *2019 IEEE International Conference on Computer Science and Educational Informatization (CSEI)*. IEEE, 2019, pp. 104–113. ISBN: 978-1-7281-2308-0. DOI: 10.1109/CSEI47661.2019.8938949.
- [35] Nataliia Klievtsova et al. “Conversational Process Modelling: State of the Art, Applications, and Implications in Practice”. In: *Business Process Management Forum : BPM 2023 Forum, Utrecht, the Netherlands, September 11-15, 2023, Proceedings*. Ed. by Chiara Di Francescomarino. Springer, 2023, pp. 319–336. ISBN: 978-3-031-41623-1. DOI: 10.1007/978-3-031-41623-1\_19.
- [36] Henrik Leopold, Jan Mendling, and Artem Polyvyanyy. “Generating Natural Language Texts from Business Process Models”. In: *Advanced information systems engineering workshops*. Ed. by Jolita Ralyté. Lecture Notes in Computer Science. Heidelberg: Springer, 2012, pp. 64–79. ISBN: 978-3-642-31095-9.

- DOI: 10.1007/978-3-642-31095-9\_5. URL: [https://link.springer.com/chapter/10.1007/978-3-642-31095-9\\_5](https://link.springer.com/chapter/10.1007/978-3-642-31095-9_5).
- [37] Xiaoxuan Li et al. “MaD: A Dataset for Interview-based BPM in Business Process Management”. In: *2023 International Joint Conference on Neural Networks (IJCNN)*. IEEE, 2023, pp. 1–8. ISBN: 978-1-6654-8867-9.
- [38] Chih-Long Lin. “Hardness of approximating graph transformation problem”. In: *Algorithms and Computations: 5th International Symposium on Algorithms and Computation, Isaac '94, Beijing, China, August 1994*. Ed. by X.S Zhang and D.Z Du. Vol. 834. Lecture Notes in Computer Science. Berlin, Heidelberg: Springer Berlin Heidelberg, 1994, pp. 74–82. ISBN: 978-3-540-58325-7. DOI: 10.1007/3-540-58325-4\_168.
- [39] Shreekant Mandvikar and Alekhya Achanta. “Process Automation 2.0 with Generative AI Framework”. In: *International Journal of Science and Research (IJSR)* 12.10 (2023), pp. 1614–1619. ISSN: 2319-7064. DOI: 10.21275/SR231021200626. URL: [https://www.researchgate.net/profile/shreekant-mandvikar/publication/375000709\\_process\\_automation\\_20\\_with\\_generative\\_ai\\_framework](https://www.researchgate.net/profile/shreekant-mandvikar/publication/375000709_process_automation_20_with_generative_ai_framework).
- [40] Bilal Maqbool et al. “A Comprehensive Investigation of BPMN Models Generation from Textual Requirements—Techniques, Tools and Trends”. In: *Information Science and Applications 2018*. Ed. by Kuinam J. Kim and Nakhoon Baek. Lecture Notes in Electrical Engineering. Singapore: Springer, 2019, pp. 543–557. ISBN: 978-981-13-1056-0. DOI: 10.1007/978-981-13-1056-0\_54. URL: [https://link.springer.com/chapter/10.1007/978-981-13-1056-0\\_54](https://link.springer.com/chapter/10.1007/978-981-13-1056-0_54).
- [41] J. Mendling, H.A. Reijers, and Wil M. P. van der Aalst. “Seven process modeling guidelines (7PMG)”. In: *Information and Software Technology* 52.2 (2010), pp. 127–136. ISSN: 0950-5849. DOI: <https://doi.org/10.1016/j.infsof.2009.08.004>. URL: <https://www.sciencedirect.com/science/article/pii/S0950584909001268>.
- [42] Mirjam Minor, Alexander Tartakovski, and Ralph Bergmann. “Representation and Structure-Based Similarity Assessment for Agile Workflows”. In: *Case-Based Reasoning Research and Development*. Ed. by Rosina O. Weber and Michael M. Richter. Lecture Notes in Computer Science. [New York]: Springer-

- Verlag Berlin Heidelberg, 2007, pp. 224–238. ISBN: 978-3-540-74141-1. DOI: 10.1007/978-3-540-74141-1\_16. URL: [https://link.springer.com/chapter/10.1007/978-3-540-74141-1\\_16](https://link.springer.com/chapter/10.1007/978-3-540-74141-1_16).
- [43] Nataliia Kliedtsova. *Nataliia / conversational\_modeling · GitLab*. 2024-02-20. URL: [https://gitlab.com/Kliedtsova/conversational\\_modeling](https://gitlab.com/Kliedtsova/conversational_modeling) (visited on 02/20/2024).
- [44] Object Management Group® Standards Development Organization. *Business Process Model and Notation (BPMN), Version 2.0*. 2011. URL: <https://www.omg.org/spec/BPMN/2.0/PDF> (visited on 01/19/2024).
- [45] Pariwat Ongsulee. “Artificial intelligence, machine learning and deep learning”. In: *2017 15th International Conference on ICT and Knowledge Engineering*. DOI: 10.1109/ictke.2017.8259629.
- [46] *OpenAI Docs - Function calling*. 2024-02-19. URL: <https://platform.openai.com/docs/guides/function-calling> (visited on 02/19/2024).
- [47] Alec Radford et al. *Improving language understanding by generative pre-training*. 2018.
- [48] Hajo A. Reijers. “Business Process Management: The evolution of a discipline”. In: *Computers in Industry* 126 (2021), pp. 103–404. ISSN: 0166-3615. DOI: 10.1016/j.compind.2021.103404. URL: <https://www.sciencedirect.com/science/article/pii/S0166361521000117>.
- [49] Nils Reimers and Iryna Gurevych. “Sentence-BERT: Sentence Embeddings using Siamese BERT-Networks”. In: *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing, EMNLP-IJCNLP 2019, Hong Kong, China, November 3-7, 2019*. Ed. by Kentaro Inui et al. Association for Computational Linguistics, 2019, pp. 3980–3990. DOI: 10.18653/v1/D19-1410. URL: <https://doi.org/10.18653/v1/D19-1410>.
- [50] Yara Rizk et al. “A Case for Business Process-Specific Foundation Models”. In: *BUSINESS PROCESS MANAGEMENT WORKSHOPS*. Ed. by Jochen de Weerd and Luise Pufahl. Vol. 492. Lecture Notes in Business Information Processing. [S.l.]: Springer, 2024, pp. 44–56. ISBN: 978-3-031-50973-5. DOI: 10.1007/978-3-031-50974-2\_4.

- [51] SAP Signavio. *SAP Signavio Process Manager*. 2023-12-12. URL: <https://www.signavio.com/products/process-manager/> (visited on 02/21/2024).
- [52] SAP Signavio. *SAP Signavio Process Transformation Suite*. 2024-01-31. URL: <https://www.signavio.com/products/process-transformation-suite/> (visited on 02/22/2024).
- [53] Schäfer, Bernhard and van der Aa, Han and Leopold, Henrik and Stuckenschmidt, Heiner. "Sketch2Process: End-to-End BPMN Sketch Recognition Based on Neural Networks". In: *IEEE Transactions on Software Engineering* 49 (2023), pp. 2621–2641. DOI: 10.1109/TSE.2022.3228308. (Visited on 02/19/2024).
- [54] Andreas Schoknecht et al. "Similarity of Business Process Models—A State-of-the-Art Analysis". In: *ACM Computing Surveys* 50.4 (2018), pp. 1–33. ISSN: 0360-0300. DOI: 10.1145/3092694. URL: <https://dl.acm.org/doi/pdf/10.1145/3092694> (visited on 12/06/2023).
- [55] Selenium. *Selenium*. 2024-02-21. URL: <https://www.selenium.dev/> (visited on 02/21/2024).
- [56] Sholiq Sholiq, Riyanarto Sarno, and Endang Siti Astuti. "Generating BPMN diagram from textual requirements". In: *Journal of King Saud University - Computer and Information Sciences* 34.10 (2022), pp. 10079–10093. ISSN: 1319-1578. DOI: 10.1016/j.jksuci.2022.10.007. URL: <https://www.sciencedirect.com/science/article/pii/S1319157822003585>.
- [57] Diana Sola et al. "Activity Recommendation for Business Process Modeling with Pre-trained Language Models". In: *SEMANTIC WEB*. Ed. by Catia Pesquita et al. Vol. 13870. Lecture Notes in Computer Science. [S.l.]: SPRINGER INTERNATIONAL PU, 2023, pp. 316–334. ISBN: 978-3-031-33454-2. DOI: 10.1007/978-3-031-33455-9\_19.
- [58] Diana Sola et al. "SAP Signavio Academic Models: A Large Process Model Dataset". In: *Process Mining Workshops - ICPM 2022 International Workshops, Bozen-Bolzano, Italy, October 23-28, 2022, Revised Selected Papers*. Ed. by Marco Montali, Arik Senderovich, and Matthias Weidlich. Vol. 468. Lecture Notes in Business Information Processing. Springer, 2022, pp. 453–465. DOI: 10.1007/978-3-031-27815-0\_33. URL: [https://doi.org/10.1007/978-3-031-27815-0\\_33](https://doi.org/10.1007/978-3-031-27815-0_33).

- [59] Thorvald Sørensen. “A method of establishing groups of equal amplitude in plant sociology based on similarity of species and its application to analyses of the vegetation on Danish commons”. In: (1948). URL: [https://www.royalacademy.dk/Publications/High/295\\_S%C3%B8rensen,%20Thorvald.pdf](https://www.royalacademy.dk/Publications/High/295_S%C3%B8rensen,%20Thorvald.pdf) (visited on 02/06/2024).
- [60] Statista. *Global business process management market 2025 | Statista*. 2024-01-13. URL: <https://www.statista.com/statistics/664859/worldwide-business-process-management-market-size/> (visited on 01/13/2024).
- [61] Ashish Vaswani et al. “Attention is All you Need”. In: *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, December 4-9, 2017, Long Beach, CA, USA*. Ed. by Isabelle Guyon et al. 2017, pp. 5998–6008. URL: <https://proceedings.neurips.cc/paper/2017/hash/3f5ee243547dee91fbd053c1c4a845aa-Abstract.html>.
- [62] Maxim Vidgof, Stefan Bachhofner, and Jan Mendling. “Large Language Models for Business Process Management: Opportunities and Challenges”. In: *Business Process Management Forum - BPM 2023 Forum, Utrecht, The Netherlands, September 11-15, 2023, Proceedings*. Ed. by Chiara Di Francescomarino et al. Vol. 490. Lecture Notes in Business Information Processing. Springer, 2023, pp. 107–123. DOI: 10.1007/978-3-031-41623-1\_7. URL: [https://doi.org/10.1007/978-3-031-41623-1\\_7](https://doi.org/10.1007/978-3-031-41623-1_7).
- [63] Viktória Farkas. *Towards a Machine Learning-Based Approach for Recommending Next Elements in BPMN Models*. NLP4BPM Workshop at BPM23, 2023. URL: <https://drive.google.com/file/d/1W6LOFHu94sdAg0b8bfs sp55t75VuhbU/view> (visited on 02/22/2024).
- [64] Zhiqiang Yan, Remco Dijkman, and Paul Grefen. “Fast Business Process Similarity Search with Feature-Based Similarity Estimation”. In: *On the move to meaningful internet systems: OTM 2010*. Ed. by Robert Meersman. Lecture Notes in Computer Science. Berlin: Springer, 2010, pp. 60–77. ISBN: 978-3-642-16934-2. DOI: 10.1007/978-3-642-16934-2\_8. URL: [https://link.springer.com/chapter/10.1007/978-3-642-16934-2\\_8](https://link.springer.com/chapter/10.1007/978-3-642-16934-2_8).
- [65] Zhengyuan Yang et al. “The Dawn of LMMs: Preliminary Explorations with GPT-4V(ision)”. In: *CoRR* abs/2309.17421 (2023). DOI: 10.48550/ARXIV.

- 2309.17421. arXiv: 2309.17421. URL: <https://doi.org/10.48550/arXiv.2309.17421>.
- [66] Zhiping Zeng et al. “Comparing stars”. In: *Proceedings of the VLDB Endowment* 2.1 (2009), pp. 25–36. ISSN: 2150-8097. DOI: 10.14778/1687627.1687631.
- [67] openAI. *GPT-4 Technical Report*. Mar. 15, 2023. URL: <http://arxiv.org/pdf/2303.08774.pdf>.
- [68] openAI. “GPTV\_System\_Card”. In: (2023). URL: [https://cdn.openai.com/papers/GPTV\\_System\\_Card.pdf](https://cdn.openai.com/papers/GPTV_System_Card.pdf) (visited on 02/19/2024).
- [69] openAI. *Introducing ChatGPT*. 2024-02-17. URL: <https://openai.com/blog/chatgpt> (visited on 02/18/2024).
- [70] openAI. *November 6, 2023 - New models and developer products announced at DevDay: GPT-4 Turbo with 128K context and lower prices, the new Assistants API, GPT-4 Turbo with Vision, DALL-E 3 API, and more*. 2024-02-23. URL: <https://openai.com/blog/new-models-and-developer-products-announced-at-devday> (visited on 02/23/2024).
- [71] Eva A. M. van Dis et al. “ChatGPT: five priorities for research”. In: *Nature* 614.7947 (2023), pp. 224–226. DOI: 10.1038/d41586-023-00288-7. URL: <https://www.nature.com/articles/d41586-023-00288-7>.
- [72] Christopher van Dun et al. “ProcessGAN: Supporting the creation of business process improvement ideas through generative machine learning”. In: *Decision Support Systems* 165 (2023), p. 113880. ISSN: 0167-9236. DOI: 10.1016/j.dss.2022.113880. URL: <https://www.sciencedirect.com/science/article/pii/S0167923622001518>.
- [73] xAI Corp. *Announcing Grok*. 2024-02-23. URL: <https://x.ai/> (visited on 02/23/2024).

# Glossary

**AI** Artificial Intelligence.

**As-is process models** Capture the current state of the business processes.

**Attention Mechanism** Fundamental innovation of the transformer architecture that enables the weighting of different parts of input data differently.

**BPM** Business Process Management.

**BPM Lifecycle** Describes the different sub-fields and methods of BPM as a continuous cycle with different phases.

**BPMN** Business Process Model and Notation.

**Chain of Thought** Strategy of linking multiple intermediate prompts to solve complex problems step-by-step, aiming for a coherent solution.

**ChatGPT** A popular LLM developed by OpenAI.

**CNN** Convolutional Neural Network.

**DL** Deep Learning.

**Document analysis** Technique for creating as-is process process models by analyzing already available documentation about existing business processes.

**Few-Shot Prompting** Method where a model generalizes and performs tasks given a small set of examples, striking a balance between zero-shot and more extensive training.

**Fine-tuning** Process of adjusting a pre-trained model on a specific, often smaller, dataset to adapt or improve its performance on related tasks.

**Frequency-aware** Concept of considering the frequency of items.

**GAN** Generative Adversarial Network.

**Gen AI** Generative AI.

**GPT-4V** Multimodal LLM developed by OpenAI.

**Ground Truth** Accurate, real-world information or data used as a benchmark to validate the accuracy and reliability of algorithms, models, or systems.

**JSON** Lightweight data interchange format that is easy for humans to read and write and for machines to parse and generate.

**LLM** Large Language Model.

**LSTM** Long Short-Term Memory.

**ML** Machine Learning.

**Multimodal** Refers to the usage of multiple modalities, in this thesis mainly the two modalities image and text.

**One-shot Prompting** Technique where a model learns to perform tasks from just one example, showcasing its ability to generalize from minimal context.

**Process discovery** Part of the BPM Lifecycle that discovers and documents as-is process models.

**Process Mining** Technique for creating as-is process models by analyzing event logs recorded by an organization's information systems.

**Prompt Engineering** Refers to crafting the input prompt for an LLM, guiding it to perform specific tasks.

**Reinforcement Learning** Type of machine learning where an agent learns to make decisions by taking actions in an environment to maximize some notion of cumulative reward.

**RNN** Recurrent Neural Network.



**Semantic-aware** Concept of allowing syntactic derivations if the meaning of the words are still the same.

**Supervised Learning** Type of machine learning where an algorithm learns to make predictions or decisions from labeled training data.

**Sørensen–Dice coefficient** Statistical measure used to calculate the similarity between two samples, in this thesis two lists.

**To-be process models** Represent the ideal state of the business processes.

**Tokens** Refers in the context of LLMs to the basic units of text, such as words or parts of words, that the model processes and generates, with pricing often based on the number of tokens processed in a request.

**Transformer Architecture** Underlying model architecture of most modern LLMs that relies on self-attention mechanisms.

**Unsupervised Learning** Type of machine learning that finds patterns or intrinsic structures in input data without the need for labeled training data.

**VAE** Variational Autoencoder.

**Zero-shot Prompting** Approach where a model tackles tasks without any prior examples, relying solely on its general understanding and pre-trained knowledge.

Name: Marvin Völter

Matriculation number: 1032090

**Declaration**

I hereby declare that I wrote the master thesis independently and used no other aids than those cited.

Ulm, date 22.3.2024

Marvin Völter