Recommending Activities for Business Process Models

Inauguraldissertation
zur Erlangung des akademischen Grades
eines Doktors der Naturwissenschaften
der Universität Mannheim

vorgelegt von

Diana Sarah Sola
aus Mannheim

Mannheim, 2023

Dekan:	Professor Dr. Claus Hertling, Universität Mannheim
Referent:	Professor Dr. Heiner Stuckenschmidt, Universität Mannheim
Koreferent:	Professor Dr. Henrik Leopold, Kühne Logistics University

Tag der mündlichen Prüfung: 12.12.2023

Abstract

Business process models are essential artifacts in organizations. They serve as a basis for effective business process management and facilitate communication among various process stakeholders. However, creating these models is a complex task. It requires a combination of domain knowledge and modeling expertise, which is difficult to find in an individual. Moreover, the distributed nature of these modeling settings poses challenges in maintaining consistency and clarity in the resulting models, especially in large-scale modeling initiatives involving thousands of models. These issues have severe consequences, potentially leading to the use of inaccurate, incomplete, or inconsistent models for downstream analyses and managerial decisions. In light of these challenges, this doctoral thesis focuses on recommending activities for business process models to support modelers and enhance their modeling experience. Specifically, it provides the following six key contributions. First, we publish and analyze the largest publicly available collection of business process models. The dataset comprises over one million models in 16 modeling and 41 natural languages, reflecting a high degree of diversity. Second, we propose an explainable activity-recommendation approach based on rules. It uses problem-specific templates for rule learning, which makes it very effective. Third, we explore different approaches to use knowledge graph completion methods for activity recommendation. Our investigations provide valuable insights into the applicability of these methods in a context that differs from traditional benchmarks. Fourth, we develop two approaches for leveraging natural language semantics contained in process models to improve recommendations for models under development that differ significantly from those used for training. One approach extends the rule-based approach with semantic components, while the other is based on a transformer-based language model. Fifth, our transformer-based approach is the first activity-recommendation approach that is able to overcome vocabulary limitations of the models used for training. This enables our approach to better adapt to new recommendation contexts that were not encountered during training. Last, we enhance the framework for evaluating activity-recommendation approaches in terms of evaluated recommendation scenarios, simulation procedures, and metrics.

Zusammenfassung

Geschäftsprozessmodelle sind essenzielle Artefakte in Organisationen. Sie dienen als Grundlage für effektives Geschäftsprozessmanagement und erleichtern die Kommunikation zwischen verschiedenen Stakeholdern von Prozessen. Die Erstellung dieser Modelle ist jedoch eine komplexe Aufgabe. Sie erfordert eine Kombination aus Domänenwissen und Modellierungsexpertise, welche bei einer Einzelperson schwer zu finden ist. Zudem stellt die verteilte Modellierungsumgebung Herausforderungen bei der Umsetzung von Konsistenz und Klarheit in den resultierenden Modellen dar, insbesondere bei groß angelegten Modellierungsinitiativen mit Tausenden von Modellen. Diese Probleme haben schwerwiegende Folgen, die zur Verwendung ungenauer, unvollständiger oder inkonsistenter Modelle für nachgelagerte Analysen und Managemententscheidungen führen können. Angesichts dieser Herausforderungen beschäftigt sich diese Doktorarbeit mit der Empfehlung von Aktivitäten für Geschäftsprozessmodelle, um Modellierer zu unterstützen und ihre Modellierungserfahrung zu verbessern. Konkret liefert sie die folgenden sechs Beiträge. Erstens veröffentlichen und analysieren wir die größte öffentlich verfügbare Sammlung von Geschäftsprozessmodellen. Der Datensatz umfasst über eine Million Modelle in 16 Modellierungs- und 41 natürlichen Sprachen und spiegelt einen hohen Grad an Vielfalt wider. Zweitens stellen wir einen erklärbaren, regelbasierten Ansatz zur Aktivitätsempfehlung vor. Dieser Ansatz verwendet problemspezifische Vorlagen zum Regellernen, was ihn sehr effektiv macht. Drittens untersuchen wir verschiedene Ansätze zur Verwendung von Methoden zur Vervollständigung von Wissensgraphen für die Aktivitätsempfehlung. Unsere Untersuchungen liefern wertvolle Erkenntnisse über die Anwendbarkeit dieser Methoden in einem Kontext, der sich von traditionellen Benchmarks unterscheidet. Viertens entwickeln wir zwei Ansätze, um die in Prozessmodellen enthaltene Semantik natürlicher Sprache zur Verbesserung von Empfehlungen für solche Modelle zu nutzen, die sich wesentlich von denen unterscheiden, die für das Training verwendet wurden. Einer dieser Ansätze erweitert den regelbasierten Ansatz um semantische Komponenten, während der andere ein Transformer-basiertes Sprachmodell verwendet. Fünftens ist unser Transformer-basierter Ansatz der erste Ansatz zur

Aktivitätsempfehlung, der die Vokabularbeschränkungen der für das Training verwendeten Modelle überwinden kann. Dadurch kann sich unser Ansatz besser auf neue Empfehlungskontexte, die während des Trainings nicht aufgetreten sind, anpassen. Schließlich verbessern wir den Rahmen zur Evaluierung von Ansätzen zur Aktivitätsempfehlung in Bezug auf evaluierte Empfehlungsszenarien, Simulationsverfahren und Metriken.

Acknowledgements

On these pages, I would like to express my sincere gratitude to all the individuals who made my PhD journey an enjoyable experience.

First, I would like to thank my supervisor Heiner Stuckenschmidt. His guidance, support and belief in my abilities were crucial in the creation of this thesis. At SAP, where I was employed as a PhD student, Andreas Gerber played a pivotal role in making this thesis possible. I want to thank Andreas for creating an environment that allowed me to focus on my research and for always being open to discussing new ideas.

Throughout my time as a PhD student, I had the privilege of working and collaborating with numerous exceptional individuals. I would especially like to express my gratitude to Christian Meilicke and Han van der Aa, who were indispensable to my main research projects and my growth as a researcher. Christian helped me navigate my PhD journey and was always available to discuss results, teaching me a great deal about conducting meaningful research. Han helped shape my research ideas, and his exceptional writing skills greatly enhanced my own. It was a great pleasure working with both of them.

Despite our PhD topics having limited overlap, I am grateful that I had such an amazing colleague in Bernhard Schäfer. Over the years, he played various roles, from office buddy to mentor, career advisor, and friend. I would like to thank Bernhard for always being available to discuss topics related to data science and other aspects of life, and for sharing both wins and struggles.

Furthermore, I want to thank Henrik Leopold, Kiran Busch and Alexander Rochlitzer for our joint research project. Our collaboration was truly enjoyable. Additionally, I would like to thank Han and Henrik for the organization of the first process analytics research seminar. After a depressing period of limited social contact due to the coronavirus pandemic, this seminar was a fruitful and enjoyable event where I had the opportunity to make valuable contacts. Moreover, I would like to thank Timotheus Kampik, Christian Warmuth and Jana-Rebecca Rehse for the collaboration in the context of the SAP-SAM dataset. Working with all of them was a very enjoyable experience. I also want to acknowledge the guidance of Peyman Badakhshan and Diana Veit in the innovation project at SAP, which my research was a part of. It was a great pleasure working with both. Further, I want to express my gratitude to Aaron Kurz, Peter Pfeiffer, and Adrian Rebmann for fruitful discussions and memorable times we shared at conferences. I am also grateful to all the members of the Data and Web Science Group at the University of Mannheim. Their collective knowledge in artificial intelligence, knowledge graphs, natural language processing, and computer vision was a source of constant learning and inspiration.

During my time as a PhD student, I was fortunate to receive support and encouragement from friends and family. I would especially like to thank Xiyuan Liu, Joana Sadik and Stephan Gruber for always believing in me more than I did. I am also grateful to my family for their belief in me and their encouragement of my enthusiasm for learning. In particular, I want to thank my sister Annabell Sola for her support throughout my school and university years, for always cheering me up, and for making me a proud big sister.

Last, I want to express my gratitude to Kiran for being an endless source of love and support. During particularly challenging times, Kiran responded with even greater empathy and loving attention. I am truly grateful for his presence in my life and his unwavering support.

Contents

Acronyms

BPM Business Process Management.

BPMAI Business Process Management Academic Initiative.

BPMN Business Process Model and Notation.

CMMN Case Management Model and Notation.

DMN Decision Model and Notation.

EPC Event-driven Process Chain.

ILP Inductive Logic Programming.

KGE Knowledge Graph Embedding.

MRR Mean Reciprocal Rank.

NLP Natural Language Processing.

SAP-SAI SAP Signavio Academic Initiative.

SAP-SAM SAP Signavio Academic Models.

UML Unified Modeling Language.

USE Universal Sentence Encoder.

List of Figures

List of Tables

Chapter 1

Introduction

This chapter provides an introduction to this doctoral thesis. In Section 1.1, we discuss the motivation behind our research on recommending activities for business process models. Subsequently, in Section 1.2, we highlight the main contributions of this thesis. In Section 1.3, we then provide an overview of the publications that have been created in this context. Finally, in Section 1.4, we offer an outline of the remaining chapters of this thesis.

1.1 Motivation

To motivate the research topic addressed in this thesis, we first provide some background on Business Process Management (BPM), highlighting the importance of business process models within this discipline. After discussing common challenges associated with business process modeling, i.e., the creation of business process models, we then introduce activity recommendation as an established approach to support process modelers. Finally, we present problem areas in this field that need to be addressed to further enhance the process modeling experience through activity recommendation.

Background. As business environments become increasingly complex and dynamic, organizations face the challenge of continuously improving and aligning their operations to remain competitive [173]. In this regard, BPM plays a crucial role in modern organizations, encompassing all management activities related to how these organizations operate [107]. Business process managament is centered around the concept of business processes, which are collections of activities performed by an organization to deliver value to customers [55]. The successful execution of these processes depends on a complex interplay of organizational and

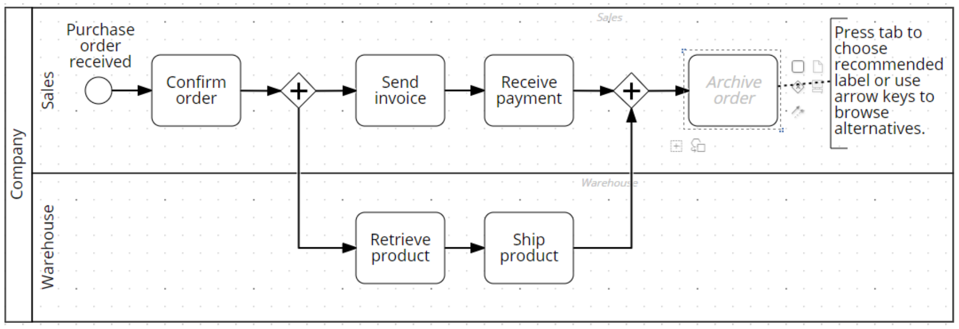


Figure 1.1: An example use case of activity recommendation

technical factors, including resources, roles, responsibilities, and supporting information systems [183].

To effectively manage business processes, organizations rely on business process models, which provide graphical representations of these processes. These models serve not only to document, analyze, improve and implement the processes [37], but also to facilitate understanding and communication among stakeholders [70], thereby enhancing collaboration and decision-making in organizations [15]. However, capturing the intricacies of business processes in models is a challenging task. Business process modeling requires both domain expertise and modeling skills, a combination that is rarely found in one individual [58]. This complicates the modeling process and necessitates considerable time and resource investments in meetings and iterative modeling sessions to bring together both modeling and domain experts [44]. Moreover, the distributed nature of these modeling settings and the potential for miscommunication and ambiguity among the involved individuals pose a risk to the consistency and clarity of the resulting process models [126, 132]. These risks are amplified in large-scale modeling projects that span the entire organizational scope.

Overall, these challenges can result in incorrect, incomplete, or inconsistent models, which can negatively impact subsequent analysis tasks and managerial decisions [1,9]. To mitigate these risks, various forms of business process modeling support have been established, one of which is activity recommendation.

Problem statement. This doctoral thesis addresses the activity-recommendation problem. Activity recommendation aims to suggest suitable labels for activities that are newly inserted by a modeler in a business process model under development. A repository of available business process models typically serves as a basis for this task. To illustrate this, consider the process model under development shown in Figure 1.1, where a user has just inserted an unlabeled activity on the model's right-hand side. The activity-recommendation task is to determine a suitable activity for this position, i.e., to find an appropriate label for it.

The process model in the figure depicts a process that starts when a purchase order has been received, after which various activities are performed to handle the order. It contains diamond shapes marked with a **+**, which represent parallel gateways used to visualize the concurrent execution of activities within the warehouse and the sales department. After the model synchronizes the two branches of concurrent activities, a new activity has been inserted, for which the activity-recommendation task is to suggest one or more suitable labels. As shown, a recommended label is *Archive order*. This label is fitting, because the preceding nodes indicate that the order has been fulfilled, after which it is natural to archive the order.

Our research is concerned with such activity recommendations. Specifically, it is driven by the need to improve existing work on recommending activities for business process models in four central areas: explainability, scalability, generalization, and evaluation. In the following, we delve into each of these areas, discussing the current state and identifying improvement opportunities.

Explainability. Research on recommender systems has increasingly acknowledged the importance of making the recommendation process more transparent to users [104]. However, this aspect has largely been neglected in the field of activity recommendation, where the focus has primarily been on increasing accuracy. Activity-recommendation approaches typically use black-box machine learning techniques that make it difficult to understand how and why a certain activity recommendation was made. In contrast, explainable approaches provide insights into the rationale behind a recommendation, making it easier to understand how the approach arrived at its decision [59]. By providing this transparency, explainable approaches can benefit both users and developers of the recommendation approach. For users, explanations can help them make more informed and faster decisions [163], leading to increased satisfaction [148]. For developers, transparency can help them better understand how the approach arrived at a particular recommendation, making it easier to debug and improve the system [175]. Therefore, an activity-recommendation approach that is not only accurate but also explainable has the potential to provide an improved experience for both users and developers.

Scalability. In today's business landscape, organizations maintain large repositories of business process models, often containing several thousand process models [34]. In fact, managing large process model repositories is a key application of BPM [35]. As the number of organizations conducting enterprise-wide and global process modeling continues to grow [134], scalability has become a crucial factor not only in process modeling itself, but also in activity recommendation. We consider an activity-recommendation approach scalable if it maintains its functionality and efficiency even as its operational context expands in size. For instance, it should continue to work effectively when applied to repositories of process models that are significantly larger in scale. However, most activity-recommendation approaches have been evaluated on small datasets only, or are simply not feasible for large repositories of process models. These issues can be contributed to limited availability of data for developing and evaluating activity-recommendation approaches, driven by legal concerns and the fear of exposing sensitive information about internal operations of organizations [161]. The publication of large-scale model collections, commonly seen in related research fields, has been largely hindered by this inherent dilemma. The extensive scale of process modeling initiatives in organizations thus introduces two essential needs related to activity recommendation. First, there is a need to develop and evaluate activity-recommendation approaches in such a way that they can effectively meet the requirements of organizations with large repositories of business process models. Second, the availability of large process model datasets is crucial in facilitating such efforts.

Generalization. In situations where a business process model under development deviates from the process models of a repository used for training, it is crucial that an activity-recommendation approach is able to generalize from the learned patterns to the new recommendation scenario. For example, if the approach has learned that the activity *check invoice* is typically followed by *send invoice*, it should be able to infer that a similar sequence applies to bills, i.e., *check bill* is followed by *send bill*. However, existing activity-recommendation approaches face limitations in their ability to generalize across activity labels, which has two main reasons. First, even though the significance of natural language semantics in process models has been acknowledged in BPM research [165], e.g., for process model abstraction [90] or anomaly detection in event logs [166], existing activity-recommendation approaches largely neglect them. This is disadvantageous, since the analysis of the natural language semantics of activity labels can reveal more general patterns, which enable the generation of more relevant recommendations in a wide range of modeling scenarios. Second, current activity-recommendation approaches provide recommendations only in the form of labels contained in the repository at hand, leading to poor recommendations for process models that significantly differ from those in the repository. An approach that is able to generate recommendations beyond the vocabulary of the repository could expand the coverage of modeling scenarios, where useful recommendations are generated. Thus, by leveraging natural language semantics and overcoming the vocabulary limitations of the given repository, an activity-recommendation approach has the potential to better generalize across activity labels, improving the quality and relevance of recommendations in diverse scenarios.

Evaluation. The evaluation of activity-recommendation approaches is a non-trivial task, as is the case with recommender systems in general [146]. Typically, these approaches are evaluated through offline experiments, where completed models contained in existing datasets are used to simulate varying states of process models under development, and the performance is measured with metrics such as hit rate and mean reciprocal rank. Offline experiments offer several advantages. They do not require interaction with real users, thereby allowing for the comparison of multiple candidate approaches at a low cost [51]. Also, they enable an effective evaluation of a large number of recommendation cases. To make the conclusions drawn from offline experiments applicable beyond the specific experimental settings, it is essential that the experiments are diverse and closely aligned with practical situations. However, current activity-recommendation research has several shortcomings in this regard that need to be addressed.

First, as discussed before, activity-recommendation approaches are often evaluated on small datasets, which may not fully represent real-world conditions. To enhance the relevance of the evaluations, it would be beneficial to use larger, more diverse datasets. Second, there is a research gap regarding the performance of these approaches in situations, where the repository contains few or even no process models that are similar to the process model under development. In particular, the potential scenario in which the process model under development only contains unseen labels is highly relevant and should be investigated for a comprehensive understanding of the recommendation performance. Third, the existing procedures for simulating a recommendation case provide either few or a lot of context for the recommendation [72, 73], while others are not even clearly defined [48, 177]. To enable more comprehensive evaluations that consider varying context lengths, it is necessary to establish an additional procedure that provides an intermediate level of context. Last, the two commonly used metrics—hit rate and mean reciprocal rank—are very strict. For instance, they would consider a recommendation of *create delivery* as a failure if the ground truth is *create shipment*, despite the similarity between these two activities. Given that an activity can be described in multiple ways, such a recommendation could still be very useful to the user. Therefore, there is a need for well-defined metrics that take such nuances into account.

The evaluation of activity-recommendation approaches could thus overall be improved by making it more comprehensive and practice-oriented. This could be achieved by using larger datasets, investigating special but important recommendation cases, refining the procedures used to simulate recommendation cases, and defining more nuanced evaluation metrics.

1.2 Contributions

This doctoral thesis focuses on activity recommendation in business process modeling. The contributions of this thesis lead to improvements in the areas of explainability, scalability, generalization, and evaluation, discussed in Section 1.1. To begin with, through the publication and analysis of the largest publicly available collection of business process models to date, we enable advancements in the areas of scalability and evaluations. Furthermore, we propose several activity-recommendation approaches that outperform existing approaches not only in established metrics but also in terms of explainability, scalability or generalization. Additionally, we have conducted numerous experiments and developed enhanced evaluation frameworks that contribute to the improvement of evaluation practices in activity recommendation. We highlight the contributions of this thesis as follows.

1. *Publication and analysis of the largest publicly available collection of business process models*: In 2019, the Business Process Management Academic Initiative published a dataset containing almost 30,000 process models in different modeling and natural languages [184]. This dataset gained significant attention and led to the development of various methods for BPM tasks, including activity recommendation. However, filtering the dataset based on specific criteria, such as using only process models of a particular modeling and natural language, or excluding different revisions of process models, significantly reduces its size. For example, the dataset contains 9,956 BPMN 2.0 models in English, to which further filters and reductions may be applied to ensure a certain model quality. Such reductions in size present challenges when training machine learning models with a large number of parameters or testing the scalability of an approach. Additionally, they limit the diversity of evaluation cases. To address these issues, we have published the SAP-SAM dataset, which includes over one million process models, with a notable subset of 157,762 BPMN 2.0 models in English. In Chapter 4, we provide insights into the characteristics of SAP-SAM. The dataset comprises models in 16 different modeling and 41 natural languages, reflecting a high degree of diversity and large dataset sizes, even after filtering.

2. *Development of an explainable activity-recommendation approach based on rules*: As discussed in Section 1.1., an activity-recommendation approach that is not only accurate but also explainable has significant advantages. By providing explanations for the generated recommendations, users can make informed decisions and choose from the presented alternatives more quickly. This ultimately enhances the overall modeling experience. Therefore, we introduce in Chapter 5 an activity-recommendation approach that is based on rules and is as such inherently explainable. Examining the rules that underpin a recom-

mendation not only helps users, but also assists researchers in debugging and refining such an approach. Our approach draws inspiration from rule-based methods used for knowledge graph completion, such as AMIE [46] or Any-BURL [106]. However, we use our domain-specific knowledge to develop an approach specifically tailored to meet the unique requirements of activity recommendation. Moreover, our experiments show that our rule-based approach is applicable on large-scale datasets and outperforms existing approaches.

3. *Exploration of different approaches to use knowledge graph completion methods for activity recommendation*: In our research, we propose an activity-recommendation approach that is inspired by rule-based methods used in knowledge graph completion. Similarly, there is another approach that employs a knowledge graph embedding model [177]. However, this approach lacks the ability to consider more than one preceding activity as a context for an activity recommendation. This highlights a research gap regarding the application of methods, originally designed for knowledge graph completion, to activity recommendation. To address this gap, we suggest in Chapter 6 to transform the given process model repository and the process model under development into a large knowledge graph. Then, we formulate the activity-recommendation task as a completion task within this graph. We explore various approaches to establish such a knowledge graph, while also experimenting with a diverse range of both embedding- and rule-based knowledge graph completion methods. Through our enhanced evaluations, we not only contribute to the understanding of the applicability of knowledge graph completion methods to activity recommendation, but also provide valuable insights into the effectiveness of these methods in a context that differs from traditional benchmarks.

4. *Leveraging natural language semantics contained in business process models*: Natural language semantics in business process models play a crucial role in understanding the context and meaning of included activities. This semantic information is valuable for activity recommendation, as it helps to generalize the activity patterns found in the model repository used for training. This in turn is particularly useful for generating recommendations for process models under development that differ significantly from those in the training repository. However, as we discuss in Chapter 3, the use of natural language semantics is hardly considered in current activity-recommendation research. In fact, no existing approach has successfully incorporated natural language semantics when dealing with large model repositories. In Chapter 7, we address this issue by extending our rule-based approach to additionally consider action and business-object patterns in the use of activities as well as their semantic similarity. In the subsequent Chapter 8, we tackle the challenge through a novel

activity-recommendation approach that employs a transformer-based language model and is as such notably scalable. This approach additionally opens new possibilities in the context of activity recommendation, which we describe in the following contribution.

5. *Overcoming vocabulary limitations of the given repository with a transformer-based approach*: An activity-recommendation approach that is able to generate label recommendations beyond the vocabulary of the given model repository has the potential to adapt more effectively to new recommendation scenarios that it may not have encountered during training. By generalizing from the examples it was trained on, such an approach can thus handle a broader range of recommendation scenarios. However, existing approaches are limited to providing recommendations in the form of labels contained in the given repository of business process models. In Chapter 8, we address this constraint by leveraging the power of transfer-learning techniques with a transformer-based language model. This enables our approach to recommend activities that were not present in the repository, thereby facilitating adaptation to new recommendation scenarios and ultimately increasing the relevance of the generated recommendations.

6. *Enhanced evaluation through the introduction of new recommendation scenarios, simulation procedures, and metrics*: As outlined in Section 1.1., current evaluations of activity-recommendation approaches fall short in terms of comprehensiveness and practical relevance. For example, the recommendation cases for evaluation are often very similar to the training examples. In our work, we aim to alleviate these issues in several respects. First, we evaluate our approaches on the large-scale SAP-SAM dataset that we publish as part of this thesis. Second, in the course of this work, we assess the approaches in different application scenarios, which vary in terms of the availability of process models in the repository that are similar to the process model under development. Third, we introduce a novel procedure for simulating recommendation cases for offline experiments, which provides a level of recommendation context that strikes a balance between existing procedures. Finally, we establish new sets of metrics that are able to take semantic similarities between the ground truth and the recommended labels into account.

1.3 Publications

The research conducted as part of this doctoral thesis has led to the publication of one journal article, two conference papers, and two workshop papers. Among these, one workshop paper focuses on the SAP-SAM dataset, while the remaining papers revolve around activity recommendation. The following list provides a chronological overview of the publications:

- D. Sola, C. Meilicke, H. van der Aa, and H. Stuckenschmidt, “A rule-based recommendation approach for business process modeling”, *Advanced Information Systems Engineering (CAISE)*, 2021, pp. 328-343.
- D. Sola, C. Meilicke, H. van der Aa, and H. Stuckenschmidt, “On the use of knowledge graph completion methods for activity recommendation in business process modeling”, *Business Process Management Workshops (AI4BPM)*, 2021, pp. 5-17.
- D. Sola, H. van der Aa, C. Meilicke, and H. Stuckenschmidt, “Exploiting label semantics for rule-based activity recommendation in business process modeling”, *Information Systems*, 2022, Article 102049.
- D. Sola, C. Warmuth, B. Schäfer, P. Badakhshan, J. R. Rehse, and T. Kampik, “SAP Signavio Academic Models: A large process model dataset”, *Process Mining Workshops (PQMI)*, 2022, pp. 453-465.
- D. Sola, H. van der Aa, C. Meilicke, and H. Stuckenschmidt, “Activity recommendation for business process modeling with pre-trained language models”, *European Semantic Web Conference (ESWC)*, 2023, pp. 316-334.

In the outline of this thesis in Section 1.4, we will mention which parts of this thesis have been published in one of the listed papers.

Moreover, the author of this doctoral thesis has contributed to a research project concerned with prompt engineering in business process management, which is beyond the scope of this thesis. This project has led to the following publication of a working conference paper:

- K. Busch, A. Rochlitzer, D. Sola, and H. Leopold, “Just tell me: Prompt engineering in business process management”, *International Conference on Business Process Modeling, Development and Support (BPMDS)*, 2023, pp. 3-11.

1.4 Outline

The remainder of this thesis is organized into eight more chapters, which we outline as follows:

- *Chapter 2: Machine Learning Foundations.* This chapter lays the groundwork for our activity-recommendation approaches by delving into essential machine learning foundations. Our research draws inspiration from methods used in the fields of knowledge graphs and natural language processing. Therefore, we first discuss rule-based and embedding-based methods for knowledge graph completion, followed by an introduction into transformer-based language models.

- *Chapter 3: Business Process Modeling.* This chapter provides an introduction into business process modeling within the broader context of BPM. After providing a general overview of BPM, we focus on business process models. Then, we highlight the importance of having adequate support for business process modeling and review related research. We conclude this chapter by summarizing the research gaps that we aim to address.

- *Chapter 4: The SAP-SAM Dataset.* This chapter presents the SAP-SAM dataset, which we have published as part of this work. We begin by describing the origins of the models contained in the dataset. Then, we explore properties of SAP-SAM as well as possible applications, and examine its limitations. Finally, we discuss related datasets. The research presented in this chapter is based on the paper titled *SAP Signavio Academic Models: A large process model dataset* by Diana Sola, Christian Warmuth, Bernhard Schäfer, Peyman Badakhshan, Jana-Rebecca Rehse, and Timotheus Kampik [154]. The author of this thesis was a main contributor to every section of this chapter, except for the origins and applications sections, which are included to ensure a comprehensive overview of the dataset. Additionally, the author made substantial contributions to the publication of the dataset and related resources.

- *Chapter 5: Rule-based Activity Recommendation.* This chapter introduces our rule-based approach for activity recommendation. First, we explain the behavioral abstraction of process models that serves as a basis for our approach. Subsequently, we explain the two main phases of the approach: rule learning and rule application. Through an extensive experimental evaluation, we finally demonstrate that our approach outperforms a variety of other methods. The research discussed in this chapter is based on the paper titled *A rule-based recommendation approach for business process modeling* by Diana Sola, Christian Meilicke, Han van der Aa, and Heiner Stuckenschmidt [151], where the author of this thesis was the main contributor.

- *Chapter 6: Activity Recommendation as Knowledge Graph Completion.* This chapter focuses on activity recommendation as a completion task in a knowledge graph. To this end, we first explore different approaches to constructing a knowledge graph based on given business process models. Then, we assess the performance of different rule- and embedding-based methods from the knowledge graph domain for the specific completion task. We also compare their effectiveness to approaches specifically designed for activity recommendation, including our rule-based approach. The research presented in this chapter is based on the paper titled *On the use of knowledge graph completion methods for activity recommendation in business process modeling* by Diana Sola, Christian Meilicke, Han van der Aa, and Heiner Stuckenschmidt [150], where the author of this thesis was the main contributor.

- *Chapter 7: Rule-based Activity Recommendation with Natural Language Semantics.* This chapter presents an extended version of our rule-based approach that incorporates natural language semantics. We first motivate the extensions in both rule learning and rule application, before we explain them in detail. In the subsequent evaluation, we demonstrate that the presented extensions serve as meaningful additions that can improve the quality of the provided recommendations. The research discussed in this chapter is based on the paper titled *Exploiting label semantics for rule-based activity recommendation in business process modeling* by Diana Sola, Han van der Aa, Christian Meilicke, and Heiner Stuckenschmidt [152], where the author of this thesis was the main contributor.

- *Chapter 8: Transformer-based Activity Recommendation.* This chapter introduces our transformer-based activity-recommendation approach. We start by explaining how we pose the activity-recommendation problem as a set of sequence-to sequence tasks in order to employ a pre-trained language model. Through an extensive experimental study, we then demonstrate that our approach is superior in terms of semantic accuracy. Moreover, we show that it is capable of handling and generating activity labels that go beyond the vocabulary of the model repository. The research presented in this chapter is based on the paper titled *Activity recommendation for business process modeling with pre-trained language models* by Diana Sola, Han van der Aa, Christian Meilicke, and Heiner Stuckenschmidt [153], where the author of this thesis was the main contributor.

- *Chapter 9: Conclusion.* This chapter concludes this thesis by summarizing the key results of the presented research. In addition, we outline promising future research directions that have the potential to further enhance our work.

Chapter 2

Machine Learning Foundations

In this section, we cover the machine learning foundations that our activity-recommendation approaches and some related works are based on. Our research draws inspiration from methods employed in knowledge graph completion and natural language processing. Therefore, in Section 2.1, we first introduce the concept of knowledge graphs and discuss both rule- and embedding-based methods for knowledge graph completion. Following this, in Section 2.2, we provide an introduction into natural language processing, with a particular focus on transformer-based language models.

2.1 Knowledge Graph Completion

The concept of *knowledge graphs* has been discussed for over 50 years [62]. However, it was not until the introduction of Google's Knowledge Graph in 2012 [147] that this concept gained significant prominence. Well-known examples of publicly accessible large-scale knowledge graphs include DBpedia [87], Freebase [10], Wikidata [174], and YAGO [61].

Knowledge graphs are a form of data representation, which leverages graphs to represent relationships between various entities in a structured and intuitive way [62]. These entities could be anything that can be an object of thought [185], such as the K-means clustering algorithm, the number pi, or the Eiffel Tower. To define and reason about the semantics of the terms used to describe these entities and their relationships, standard knowledge representation formalisms, such as ontologies [13] or rules [68], can be used. Moreover, there are numerous representations that support the application of machine learning techniques over knowledge graphs [180]. Thus, the decision to use a knowledge graph enables a range of techniques for integrating knowledge and extracting value from data [62].

$$\begin{tikzcd}
\text{Max} \arrow[r, shift left, "worksWith"] & \text{Jane} \arrow[l, shift left] \arrow[rr, shift left, "worksWith"] \arrow[dr, "reportsTo"'] & & \text{Joe} \arrow[ll, shift left] \arrow[dl, "reportsTo"] \\
& & \text{Erika} &
\end{tikzcd}$$

Figure 2.1: An example knowledge graph

In a knowledge graph, entities are represented as nodes, while the relationships between these entities are modeled as edges connecting the nodes. Thus, a knowledge graph is essentially a multi-relational graph composed of nodes (entities) and different types of edges (relations) [180]. A directed edge from one node to another, labeled with a specific relation, forms a triple *(s,r,o)*, where *s* is the subject, *r* is the relation, and *o* is the object. This can be interpreted as the fact that the subject *s* is in relation *r* to the object *o*.

To illustrate this, consider the example knowledge graph depicted in Figure 2.1. This knowledge graph, derived from a corporate context, captures the professional relationships between *Max*, *Jane*, *Joe*, and *Erika*. For example, the knowledge graph contains a directed edge from *Joe* to *Erika*, labeled *reportsTo*. This suggests a hierarchical relationship, where Joe is in a subordinate position to Erika, thus explaining why Joe reports to her.

Despite their usefulness in various applications, such as web search [147] or recommendations [53], knowledge graphs are often incomplete as knowledge may exist implicitly or may be entirely missing [145]. Therefore, knowledge graph completion is concerned with predicting such missing information in a knowledge graph using the existing facts. According to Chen et al. [23], knowledge graph completion methods can be divided into two categories: traditional knowledge graph completion methods and methods based on knowledge representation learning. In Section 2.1.1, we delve into a specific type of traditional knowledge graph completion methods, namely rule-based methods. Then, in Section 2.1.2, we discuss methods based on knowledge representation learning, which embed entities and their relationships into a continuous low-dimensional vector space.

2.1.1 Rules

Given the symbolic nature of knowledge, employing *rules* for knowledge graph completion is an intuitive choice [75]. In this context, rules are typically expressed using the syntax of first-order logic [157]. An example for such a rule is given by

$$\textit{worksWith(Y,X)} \leftarrow \textit{worksWith(X,Y)}. \tag{2.1}$$

This rule captures a pattern in the knowledge graph depicted in Figure 2.1, namely that if a person *X* works with a person *Y*, then usually *Y* also works with *X*.

Within such rules, the smallest component is referred to as an *atom*. Specifically, an atom is a formula $r(X, Y)$, where r denotes a binary relation, and both X and Y can be variables or constants. In the context of knowledge graphs, constants correspond to specific entities within the graph. The rule (2.1), for instance, is composed of the atoms *worksWith(X,Y)* and *worksWith(Y,X)*. Other examples of atoms include *worksWith(Max,Y)* and *worksWith(Max,Jane)*. These atoms differ in the number of constants they contain. We call an atom *instantiated* if it includes at least one constant, such as *worksWith(Max,Y)*. If an atom is entirely composed of constants, like *worksWith(Max,Jane)*, then we consider it as *grounded*. In terms of such logical formulas, a specific fact (s, r, o) in a knowledge graph can thus be expressed as grounded atom $r(s, o)$.

In the context of knowledge graphs, we are typically interested in patterns that can be described using *Horn rules*. A Horn rule has the form *head←body*. Here, the head corresponds to an atom, while the body consists of a set of atoms. Coming back to rule (2.1), we can see that it conforms to the format of a Horn rule, with the body consisting of a single atom. Moreover, rule (2.1) only contains variables, describing a general pattern. However, rules can also describe specific instances of a pattern. In this context, we use the term *rule grounding* to refer to an application of a rule to a specific case. Specifically, a rule grounding is a variant of the rule, where all included atoms are grounded. An example of a rule grounding, in the context of the knowledge graph depicted in Figure 2.1, is given by

$$\textit{worksWith(Max,Jane)} \leftarrow \textit{worksWith(Jane,Max)}.$$

With the concept of rule groundings at hand, we can define the *predictions* of a rule *head←body* in a knowledge graph $\mathcal{K}$ as the set of head atoms of all rule groundings, where the body atoms appear as facts in $\mathcal{K}$. The predictions of a set of rules can then be derived as the union of the predictions of each rule [157]. To illustrate this, consider the knowledge graph in Figure 2.1 and the rule

$$\textit{reportsTo(X,Z)} \leftarrow \textit{worksWith(X,Y), reportsTo(Y,Z)}. \tag{2.2}$$

This rule expresses that if person *X* works with person *Y*, and *Y* reports to person *Z*, then X also reports to *Z*. The predictions of this rule in the knowledge graph in Figure 2.1 are *reportsTo(Jane,Erika)*, *reportsTo(Joe,Erika)*, and *reportsTo(Max,Erika)*. As the last fact is missing in the knowledge graph, this rule provides a useful prediction that contributes to the completion of the knowledge graph.

Before we can use rules to predict missing facts in a knowledge graph, we first need to learn them. The concept of rule learning is rooted in *Inductive Logic Programming (ILP)*. Prominent examples of ILP methods include FOIL [128] and WARMR [57]. In general, ILP methods use a background theory and a set of positive and negative examples to learn a set of rules, with the rules being designed to cover all positive examples and none of the negatives [29].

However, traditional ILP methods, such as FOIL and WARMR, are generally unsuitable for modern knowledge graphs, which has two primary reasons [85]. First, these methods struggle to handle large amounts of facts. Second, they do not take into account the *open world assumption* that current knowledge graphs adopt. The open world assumption suggests that any fact not present in the knowledge graph could potentially be true or false [113]. In particular, the absence of a fact does not necessarily mean it is incorrect.

Thus, while a knowledge graph under the open world assumption does provide a background theory and positive examples through existing facts, it does not provide negative examples. Without these negative examples, traditional ILP methods cannot be directly applied. For instance, FOIL operates under the assumption that the user can provide explicit counter-examples for the rules [128]. On the other hand, WARMR generates negative examples by assuming that facts not explicitly available are false, which is in line with the *closed world assumption*, not the open world assumption [57].

In response to the limitations of traditional ILP methods in handling modern knowledge graphs, several rule-based approaches specifically designed for large knowledge graphs have been proposed. These approaches can be broadly classified into two categories: *top-down* and *bottom-up* approaches [157]. Top-down approaches start with general rules and iteratively refine them. This refinement process continues until the rules become too specific, leading to a number of predicted positive examples that falls below a certain threshold. On the other hand, bottom-up approaches initially employ multiple example rules as a basis and progressively generalize them. This generalization process continues until the rules become too general, resulting in a number of predicted negative examples exceeding a certain threshold. In Chapter 5, we will propose a novel rule-based approach tailored for activity recommendation, which does not conform to the traditional categories. Our approach uses rule templates to ensure full control over the types of generated rules, learning only such rules that are relevant for activity recommendation.

Prominent examples for top-down and bottom-up approaches are AMIE [46] and AnyBURL [106], respectively. AMIE employs three different refinement operators, which take a set of rules as input and return a set of more specific rules. AnyBURL uses so-called bottom rules, which are grounded rules derived from sampled paths in a given knowledge graph. After sampling a path from the knowledge graph and constructing such a bottom rule from that path, a generalization lattice rooted in the bottom rule is built and all useful rules that appear in the lattice are stored. In Chapter 6, we will approach the activity-recommendation problem by framing it as a knowledge graph completion task in order to investigate the suitability of AnyBURL for addressing this particular task. In the context of knowledge graph completion on typical benchmarks, experimental evaluations have demonstrated that AnyBURL is competitive with knowledge graph completion methods based on representation learning [106], which we discuss in the following section.

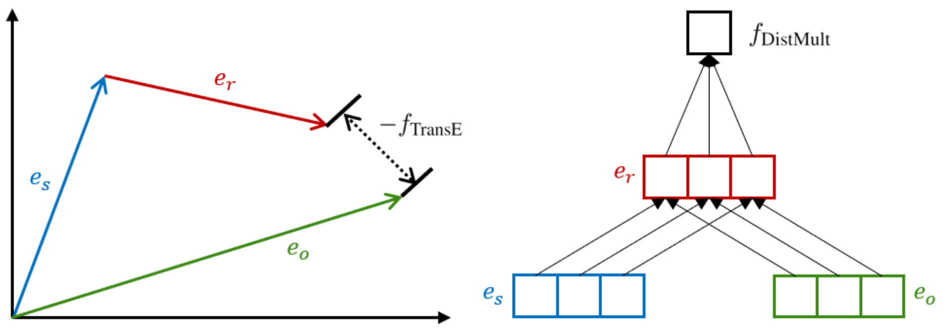
2.1.2 Knowledge Graph Embeddings

Knowledge representation learning models, also known as *Knowledge Graph Embedding (KGE) models*, are based on the idea of embedding entities and relationships of a knowledge graph into a continuous low-dimensional vector space while preserving the inherent structure of the knowledge graph [180]. In essence, a KGE model is an injective function that maps entities and relations to low-dimensional real value vectors, so-called *embeddings*, maintaining the intrinsic relationships between them [157]. For instance, in the context of the knowledge graph shown in Figure 2.1, we would like to embed the entity *Jane* in a way that its embedding is close to the embeddings of the entity *Max* or other co-workers.

KGE models simplify operations on knowledge graphs, which has two main reasons [157]. First, they reduce the dimensionality of object representations. This is particularly valuable when dealing with knowledge graphs that contain millions of entities. By employing a KGE model, these entities can be represented as embeddings of just a few hundred dimensions. Second, the structure of the embedding space allows for meaningful comparisons of objects that were previously incomparable in their original forms. For example, a distance between entities can be measured by calculating the Euclidean distance between their embeddings.

In general, KGE models involve three steps [180]. The initial step includes specifying the form in which entities and relations are represented in continuous low-dimensional vector spaces. This is followed by the definition of a *scoring function* that measures the plausibility of facts *(s,r,o)*. Finally, the representations of entities and relations as embeddings are learned by solving an optimization problem that maximizes the total plausibility of facts contained in the knowledge graph.

The main difference between KGE models lies in their scoring functions [23]. According to Wang et al. [180], we can roughly classify the models into those that use distance-based scoring functions and those that leverage similarity-based scoring functions. In the following, we explore one representative model from each category, namely TransE [11] and DistMult [188], both of which we will employ in Chapter 6.



(a) TransE **(b)** DistMult

Figure 2.2: Illustrations of the KGE models TransE and DistMult, adapted from [75]

As illustrated in Figure 2.2a, TransE embeds both entities and relationships into the same vector space, representing relationships as translations from subject to object entities [11]. Specifically, if the fact *(s,r,o)* holds, then the sum of the embedding e_s of subject s and the embedding e_r of relation r should be close to the embedding e_o of object o. Consequently, the scoring function is defined as the negative distance between $e_s + e_r$ and e_o:

$$f_{\text{TransE}}(e_s, e_r, e_o) = -||e_s + e_r - e_o||,$$

where $||\cdot||$ can be the L_1 or L_2 norm. Note that the scoring function f_{TransE} uses the negative distance, as it measures the plausibility of a fact. Therefore, $f_{\text{TransE}}(e_s, e_r, e_o)$ should be large for a fact *(s,r,o)* in the knowledge graph.

Based on the idea of a translation-based scoring function, many variants of TransE have been proposed. For example, TransH [181] interprets a relation as a translating operation on a hyperplane, while TransR [96] introduces relation-specific projections of the entities.

Similar to TransE, DistMult also embeds entities and relationships into the same vector space. However, as illustrated in Figure 2.2b, DistMult uses multiplicative operations to model relations as pairwise interactions between the entity embeddings along the same dimensions [188]. Given the relation embedding $e_r \in \mathbb{R}^d, d \in \mathbb{N}$ and the entity embeddings $e_s, e_o \in \mathbb{R}^d$, the scoring function is thus given by

$$f_{\text{DistMult}}(e_s, e_r, e_o) = e_s^T \text{diag}(e_r) e_o = \sum_{i=1}^{d} (e_r)_i \cdot (e_s)_i \cdot (e_o)_i,$$

where $\text{diag}(e_r) \in \mathbb{R}^{d\times d}$ denotes the diagonal matrix constructed from the vector e_r. As such, DistMult is a simplification of the RESCAL model [117]. Unlike DistMult, RESCAL does not restrict to diagonal matrices. Instead, it uses full matrices to represent relations as pairwise interactions between the entity embeddings along all dimensions. In terms of scoring functions, this translates into $f_{\text{RESCAL}}(e_s, e_r, e_o) = e_s^T M_r e_o = \sum_{i=1}^{d} \sum_{j=1}^{d} (M_r)_{ij} \cdot (e_s)_i \cdot (e_o)_j$, where M_r denotes the relation matrix.

When it comes to making predictions in a knowledge graph, scoring functions are used for assigning potential triples a score and ranking the candidates accordingly. For instance, in entity prediction, conventional KGE models compute scores for all candidate entities and rank them based on these scores. However, the transformer-based KGT5 model [138] deviates from this approach, as it does not use a scoring function to make predictions. Transformer-based models originate from the field of natural language processing, as we explore in the following section.

2.2 Natural Language Processing

Natural Language Processing (NLP) is an interdisciplinary field that investigates the use of computational systems for processing natural (i.e., human) language [30]. According to Khurana et al. [77], NLP can be divided into two main components: natural language understanding and natural language generation. The former focuses on interpreting natural language and extracting meaning, with tasks like text classification [47], sentiment analysis [182], and named entity recognition [94]. The latter focuses on generating human-like text, covering tasks like text summarization [38], question answering [116] and machine translation [156].

A crucial aspect of NLP is *language representation learning* [2]. Similar to the concept of knowledge representation learning, which we previously discussed in Section 2.1.2, language representation learning is primarily concerned with deriving real-valued vector representations, i.e., embeddings, that capture the inherent semantics of a given text. These representations are essential for making text machine-processible, and are commonly used in *language models*. Language models essentially aim to predict a word given its context [112]. For example, a language model may predict the next word in a sentence based on the previous words.

Notable early methods that generate word embeddings include word2vec [111] and GloVe [124]. Both are unsupervised learning methods and create word embeddings that capture semantic meanings of words. In particular, these models generate word embeddings such that semantically similar words tend to have embeddings that are close to each other in the embedding space [111]. Moreover, the word embeddings can be meaningfully combined using simple algebraic operations, e.g., $e_{Paris} - e_{France} + e_{Italy} \approx e_{Rome}$ [111].

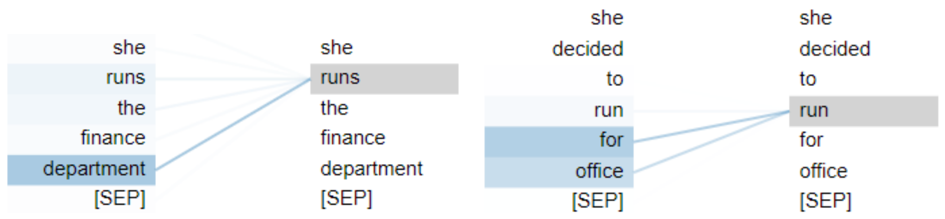


Figure 2.3: Visualization of the attention between the verb *run* and other words in two different sentences, i.e., contexts, using the visualization method by Jesse Vig [172]

However, these early methods generate context-independent word embeddings. In other words, they lack the ability to represent different meanings of words in different contexts [176]. Thus, they are unable to handle linguistic phenomena like polysemous words [158], i.e., words with multiple related meanings. For example, among other meanings, the verb *run* can imply being in charge (as in *she runs the finance department*), or it can mean standing as a candidate in an election (as in *she decided to run for office*).

To address this limitation, language models nowadays use contextualized word representations. Leveraging the sequential nature of textual data, recurrent neural architectures, such as long short-term memory, can be used to generate such contextualized representations [25, 159]. However, it is the *transformer* architecture that not only allows for contextualized word embeddings but also brings significant improvements in the performance and capabilities of language models [171].

2.2.1 Transformer

The transformer architecture completely relies on the *attention* mechanism, dispensing with recurrence [171]. This is a significant benefit as recurrent architectures impede efficient parallelization [171], and recurrent neural networks struggle with capturing long-range dependencies [60].

The attention mechanism enables a language model to selectively pay attention to relevant parts of a context when making predictions [5]. Figure 2.3 illustrates the attention mechanism in the previous example of the verb *run* and its different meanings. In the sentence *she runs the finance department*, a part of the attention mechanism focuses on the word *department*, as this word is crucial for understanding the contextual meaning of *run* in this sentence, which is to be in charge. Similarly, in the sentence *she decided to run for office*, the model pays attention to the words *for* and *office*, as they imply that *run* means standing as electoral candidate in this particular context.

The use of the transformer architecture in combination with *pre-training* on large volumes of unlabeled text using *self-supervised* learning allows for powerful language models. These models show remarkable capabilities across a diverse range of downstream tasks such as text summarization [99], machine translation [179], reasoning [80], and many more.

In self-supervised learning, the model is trained on labels generated from the data itself, as opposed to manually annotated labels. This learning approach has a major advantage over supervised learning: While supervised learning requires a large dataset labeled for a specific task, self-supervised learning enables leveraging the large amounts of unlabeled data that are available in the era of big data [98].

The algorithms for self-supervised learning differ mainly in the strategy for automatically deriving labels from the data [39], known as the pre-training objective. For instance, the well-known BERT model [32] uses a masked language modeling objective, which means that it is trained to predict words in a text that have been randomly masked, given the unmasked words as context. The T5 model [129], which we will use in Chapter 8, employs a variation of masked language modeling, where contiguous word spans rather than single words are masked, and the model is tasked with predicting the masked span.

Pre-training a language model using such self-supervised objectives enables it to develop general-purpose abilities and knowledge that can then be transferred to downstream tasks [129]. This concept of *transfer learning* is especially useful in natural language processing, where pre-training enables a model to capture polysemous disambiguation, lexical and syntactic structures, and factual knowledge [56].

Transfer learning is typically implemented in the *fine-tuning* paradigm, where a pre-trained model is further trained on a labeled dataset specific to the downstream task to turn it into a task-specific model [16]. However, given the remarkable task-agnostic performance of large language models [16, 142], *prompt-engineering* has gained in importance [97]. In the prompt-engineering paradigm, natural language task specifications, known as prompts, are provided to the language model during inference. These prompts establish the context for the downstream task without modifying the language model itself.

Independent of whether fine-tuning or prompt-engineering is employed, the transformer architecture has an *encoder-decoder* structure [171]. As such, it maps a textual input sequence $\mathbf{x} = (x_1, \dots, x_T)$ to a textual output sequence $\mathbf{y} = (y_1, \dots, y_{T'})$, where the output length T' is unknown a priori and may differ from the input length T [160]. For example, in the case of machine translation, the input sequence corresponds to a text in a source language, while the output sequence is the translated text in a target language.

To map the input sequence $\mathbf{x}$ to an output sequence $\mathbf{y}$, the encoder part of the transformer first generates a contextualized representation $\bar{\mathbf{x}}$ of $\mathbf{x}$. Then, the decoder generates the output sequence $\mathbf{y}$ one element at a time. This generation process is *auto-regressive*, which means that, in addition to the contextualized input sequence $\bar{\mathbf{x}}$, the previously generated elements are used in order to generate the next element in the output sequence. Formally, the decoder models the conditional probability distribution $P(\mathbf{y}|\bar{\mathbf{x}})$ of the output sequence $\mathbf{y}$ given the input representation $\bar{\mathbf{x}}$, which by Bayes' rule can be written as:

$$P(\mathbf{y}|\bar{\mathbf{x}}) = \prod_{t=1}^{T'} P(y_t|y_1, \ldots, y_{t-1}, \bar{\mathbf{x}}).$$

Most decoding methods aim to find the most likely output sequence, i.e., they want to find the output sequence $\hat{\mathbf{y}} = \arg\max_{\mathbf{y}} P(\mathbf{y}|\bar{\mathbf{x}})$ [71]. A simple decoding method is *greedy search*, which selects at each time step the most probable element, i.e., $\hat{y_t} = \arg\max_{y_t} P(y_t|y_1, \ldots, y_{t-1}, \bar{\mathbf{x}})$ for each $t \in \{1, \ldots, T'\}$. More sophisticated decoding methods include random sampling and beam search [50]. The *random sampling* method generates the output sequence by sampling the next element from the conditional probability distribution at each time step. *Beam search* with width *w*, which we will use in Chapter 8, uses the conditional probability distribution to track the *w* most likely output sequences at each time step.

Transformer-based encoder-decoder models, also known as transformer-based sequence-to-sequence models, can not only be used for typical NLP tasks such as machine translation. As we will explore in Chapter 8, they can also be employed for recommending activities for business process models in order to enhance business process modeling.

Chapter 3

Business Process Modeling

In this chapter, we discuss different aspects of business process modeling, a crucial sub-discipline of Business Process Management (BPM). The primary goal of BPM is to enhance the quality of products or services by refining an organization's business processes. In this context, business process models emerge as useful artifacts. However, creating these models, i.e., business process modeling, is a complex task that requires adequate support.

This chapter is organized as follows. Section 3.1 provides an overview of BPM in terms of a lifecycle. Subsequently, Section 3.2 introduces a formal definition of business process models and explores the notion of process model semantics. Section 3.3 then discusses common challenges encountered in business process modeling and provides an overview of related work in the field of business process modeling support, before Section 3.4 summarizes identified research gaps.

3.1 Business Process Management

Business process management plays an important role in organizations, which is reflected by the significant interest from both practical and academic perspectives [88]. While organizations are carrying out BPM initiatives to improve their operational performance, researchers are working on the development of methods and techniques to facilitate such initiatives [37]. In essence, BPM includes all management activities related to the way organizations conduct their operations [107]. The central concept driving these activities is the notion of *business processes*.

Business processes exist in organizations of all types and sizes, across different industries and sectors. In short, business processes structure the operations of an organization. However, the notion of business processes has evolved over the years, which led to multiple definitions emphasizing different aspects. At its core, a busi-

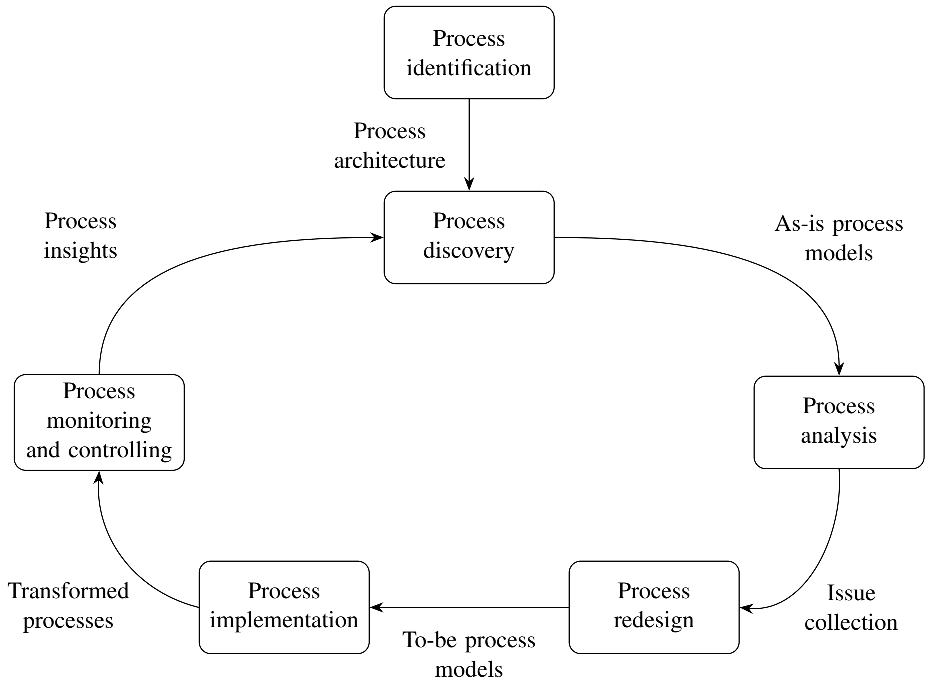


Figure 3.1: Business process management life cycle, adapted from [37]

ness process can be defined as a collection of activities that are performed by an organization to create an output, which delivers value to a customer [55]. This core notion of business processes already captures two important aspects, i.e., the focus on delivering value to a customer and the creation of output through a collection of activities. The customer can be either internal or external to the organization [37] and the activities can be further characterized as logically-related [27], where the order of the activities is determined by a set of pre-defined conditions [167]. Another important aspect is the organizational and technical environment in which a business process is carried out [183], as it affects the outcome of the business process. This includes, for example, the resources involved, the roles and responsibilities of actors, or the information systems supporting the process.

In today's dynamic business environments, *"every good process eventually becomes a bad process"* [54] if changes are not recognized or appropriately addressed. This is where BPM comes to play, as it enables organizations to flexibly react to constantly changing business environments [88]. The need for continuous adaption is the reason why BPM activities are typically structured in a life cycle, allowing for iterative refinement and ongoing improvement. However, the cyclical structure of BPM phases is idealistic and does not imply a strict temporal ordering in which they need to be executed. For example, BPM initiatives involving concurrent activities in multiple phases are common [183]. As the core of this thesis revolves around *business process models*, we adopt the comprehensive BPM life cycle proposed in [37] that allows understanding the significance of business process models in the life cycle and their usage in different contexts. Figure 3.1 provides a visual representation of the proposed life cycle, along with the outputs generated by each of its phases. The life cycle consists of six phases, which we outline as follows:

- **Process identification.** Given a business problem, relevant processes are identified and linked, before being represented in a process architecture.
- **Process discovery.** The current state of relevant processes is documented, yielding a collection of as-is business process models.
- **Process analysis.** The as-is processes are analyzed to obtain a structured collection of issues, outlining process weaknesses or potential improvements.
- **Process redesign.** The identified issues are addressed with redesigned processes, typically in the form of to-be business process models.
- **Process implementation.** The redesigned processes are implemented by realizing the required organizational and infrastructural changes to transform the as-is processes into the to-be processes.
- **Process monitoring and controlling.** Based on the data collected during process monitoring and the process insights derived, the transformed processes are continuously controlled.

Although our overview of the lifecycle phases and related activities is concise, it becomes evident that business process models are crucial artifacts in the context of the BPM lifecycle. Specifically, they can be used to represent as-is and to-be processes in the discovery and redesign phases, serving as useful means for the analysis and implementation phases. In essence, business process models capture information on business processes in a graphical manner. In the next section, we delve deeper into business process models, providing a formal definition and introducing the notion of process model semantics.

3.2 Business Process Models

To capture the operations of an organization in a business process model, many modeling languages are available, e.g., Petri Nets [125] or Business Process Model and Notation (BPMN) [118]. In this thesis, we aim to formalize business process models as generically as possible, enabling the development of approaches that are not constraint by any specific modeling language. Therefore, we adopt an abstract view, in which a business process model is defined as a directed attributed graph:

Definition 1 (Business process model) *Let $\mathcal{T}$ be a set of node types and $\mathcal{L}$ be the universe of node labels. A business process model is a tuple $M = (N, A, E, \tau, \lambda)$, where*

- *N is a set of nodes,*
- *$A \subseteq N$ is a set of activity nodes,*
- *$E \subseteq N \times N$ is a set of directed edges, such that all nodes of N are connected,*
- *$\tau : N \rightarrow \mathcal{T}$ is a function that maps a node to a type, and*
- *$\lambda : N \rightarrow \mathcal{L}$ is a function that maps a node to a label.*

The pre-set *of a node $n \in N$, i.e., the set of all nodes preceding n in the model, is given by $\bullet n = \{m \in N \mid (m, n) \in E\}$.*

This definition represents business process models as a set of connected nodes, where each node has a specific type and label. In its most basic form, a business process model contains activity and control-flow nodes. While activity nodes represent actions or indicate that something happens, control-flow nodes capture the execution flow of activities. Business process models may also include nodes that represent data objects, resources, roles, or responsibilities.

Compared to a similar view of business process models as directed attributed graph in [33], Definition 1 explicitly captures the set of activity nodes A, since these nodes are core to the activity-recommendation problem addressed in this thesis. Depending on the modeling language, this set may contain nodes of multiple types, i.e., there exists a subset of activity types, $\mathcal{T}_\mathcal{A} \subseteq \mathcal{T}$, such that $A = \{n \in N \mid \tau(n) \in \mathcal{T}_\mathcal{A}\}$. As an example, we consider the BPMN model in Figure 3.2. The model depicts a business process that starts when a purchase order is received, after which various activities are performed to handle the order. This involves a decision point, indicated by an *XOR-split gateway* (diamond shape with an X), where the order is either rejected, or confirmed and then further processed. For the BPMN modeling language, set A includes, among others, *tasks* (e.g., *Check stock availability*) and *events* (e.g., *Purchase order received*), whereas *gateways* (e.g., XOR splits) are not included in A.

Note that the edges in E can capture a partial order between nodes in N, to allow also for concurrency and alternative executions paths in a process, such as the two choices following the XOR-split gateway in Figure 3.2.

The business process model depicted in Figure 3.2 demonstrates the use of BPMN as modeling language to represent a purchase order process. In the context of BPM, a modeling language employs a notation, i.e., a collection of graphical symbols, to visually represent a business process. *Business process modeling* is the human activity of creating a business process model using such a language [107].

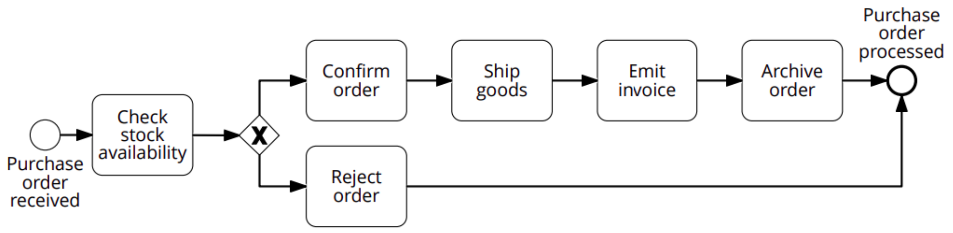


Figure 3.2: An example business process model in BPMN

In this thesis, we are interested in the semantics of business process models, which can best be understood by exploring the relationships between syntax, semantics and notation of modeling languages [41]. The syntax includes a set of constructs represented by the modeling notation, along with rules for combining these constructs [76]. The semantics, on the other hand, involve assigning meaning to these constructs [88]. The semantics of business process models can thus be derived from a combination of two aspects [162]:

1. The *formal semantics* of a modeling language: These dictate how a modeled process should be executed, capturing aspects such as the execution flow, choices, and concurrency.
2. The *natural language semantics* of individual model elements: These express the meaning of the individual parts of a process model through natural language labels.

As an illustration of this dual nature of process model semantics, consider the rightmost element in the business process model depicted in Figure 3.2, labeled *Purchase order processed*. First, this element represents an *end event* within BPMN, and as such, adheres to the formal semantics of the modeling language. This implies, for instance, that no outgoing sequence flow can occur from this point in the model. Second, this element has the label *Purchase order processed*, which, in terms of natural language semantics, indicates that a particular purchase order has been successfully processed.

Building on this notion of process model semantics, we can identify additional benefits that business process modeling offers to organizations. Through the lens of the BPM life cycle, we already recognized that process models can be used as a basis to effectively document, analyze, improve and implement business processes. Through the effective combination of formal and natural language semantics, business process modeling further facilitates understanding and communication of business processes across different stakeholders [70]. This, in turn, contributes to enhanced collaboration and decision-making in organizations.

However, to realize these benefits, several challenges have to be overcome. In the following section, we outline common challenges associated with business process modeling and discuss existing approaches to support and enhance the business process modeling experience.

3.3 Business Process Modeling Support

Business process modeling is an essential, yet complex task in organizations [101]. On one hand, it heavily relies on the knowledge and experiences of domain experts [58], who are usually process participants performing the activities of a business process on a regular basis [37]. On the other hand, it requires modeling expertise, which domain experts or casual modelers typically lack [43, 135].

A potential solution to this dilemma is bringing domain and modeling experts together in order to create business process models in collaboration [44]. However, this approach necessitates interviews, meetings or workshops, as well as multiple modeling iterations, leading to significant time and costs [140]. In addition, the distributed nature of such modeling settings makes it difficult to ensure consistency and clarity in the resulting models [126], due to ambiguities or miscommunications among the involved individuals [132]. These issues are amplified in large-scale modeling projects, where business process modeling is conducted on an organization-wide scope. Ultimately, these difficulties can result in scenarios where downstream analyses and managerial tasks are based on incorrect, incomplete, or inconsistent models [1, 9]. To prevent this, different forms of business process modeling support have been established.

One such approach involves providing developers of business process models with a set of modeling guidelines. By adhering to such guidelines, modelers can ensure the quality of their business process models. Available modeling guidelines come in different levels of abstraction, ranging from abstract frameworks, such as the *Guidelines of Modeling* [7], to directly applicable guidelines, such as the *Seven process modeling guidelines* [110], and even to more detailed, pragmatic guidelines as collected by Moreno-Montes et al. [114]. While these guidelines provide useful advice for process modelers, they are generic and do not take the current state and content of a given process model under development into account.

However, there are approaches that support process modelers beyond mere guideline-based modeling by providing recommendations during modeling time. These recommendations are based on the current modeling context and are presented in an autocompletion-like manner. Typically, such recommendation approaches use previously created business process models collected in a repository as a basis for the recommendation. In the following, we delve into two central categories of these approaches: those that suggest process model fragments and those that recommend activities. Moreover, we provide an overview of different approaches within each category.

3.3.1 Recommending Process Model Fragments

In the context of this thesis, *process model fragments* are business process models in the sense of Definition 1, i.e., they consist of one or more nodes with types and labels as well as directed edges connecting the nodes. Beyond the scope of our work, it is possible that process model fragments do not qualify as business process models. Other than in our definition, business process models are often considered as executable, which means that they must fulfill additional criteria such as correctness or consistency. In that regard, a process fragment differentiates from a business process model as it is not necessarily executable, but can be extended to become an executable business process model [144]. More intuitively, a process fragment can be understood as a logically coherent set of process elements belonging together [82]. For example, in the business process model depicted in Figure 3.2, a process fragment consists of the tasks *Check stock availability*, *Confirm order* and *Reject order* and the connecting XOR-split gateway.

The literature offers various approaches for recommending process model fragments, including approaches that are based on tags, graphs or semantic representations of business process models.

The approach by Hornung et al. [65] uses tags for the recommendation of process model fragments. It automatically generates tags from the labels of a business process model under development, which can then be used to recommend relevant process model fragments to the user. However, this approach requires manual identification and tagging of process model fragments from a given repository of business process models before recommendations can be made. Koschmider et al. [81] propose an extension of this approach, where the automatic tagging mechanism incorporates additional information, such as a textual process description that can optionally be provided by the user.

Several approaches [19, 31, 95] utilize abstractions of business process models as directed graphs to recommend process model fragments. These approaches employ graph-mining techniques to extract fragments from a repository of process models, before they use common subgraph distance [19] or edit distance [19,31,95] to measure the similarity of the business process model under development and patterns in the repository. Mazanek and Minas [102] introduce another graph-based approach that relies on graph grammars for business process models. However, their method is strictly syntax-based, which means that the elements in the suggested process model fragments only have default labels. Mazanek et al. [103]

Table 3.1: *Overview of approaches for recommending process model fragments*: For each approach, we show its type, the automation level of the two components for identifying and matching process model fragments, the ability to recommend labeled fragments and the applicability on large repositories of business process models.

Approach type / Authors	**Automated identification component**	**Automated matching component**	**Labeled process model fragments**	**Large process model repositories**
Tag-based				
Hornung et al. [65]	✗	✓	✓	✗
Koschmider et al. [81]	✗	✓	✓	✗
Graph-based				
Li et al. [95]	✓	✓	✓	✗
Deng et al. [31]	✓	✓	✓	✗
Cao et al. [19]	✓	✓	✓	✗
Mazanek and Minas [102]	✓	✓	✗	✓
Mazanek et al. [103]	✓	✓	✗	✓
Based on semantic representations				
Hornung et al. [66, 67]	✗	✗	✓	✗
Wieloch et al. [186]	✓	✗	✓	✗

extend this approach by incorporating sketch recognizers, allowing for the recommendation of process model fragments for sketched business process models.

Semantic representations of business process models are used in the approaches by Hornung et al. [66, 67] and Wieloch et al. [186]. Hornung et al. use the Web Ontology Language OWL [3], a Semantic Web language, for representing business process models and for computing the semantic similarity between process model fragments in a repository and the process model under development. Additionally, they integrate constraints imposed by business rules using the Semantic Web Rule Language [68]. In contrast to Hornung et al.'s approach, which requires manual identification of process model fragments, Wieloch et al.'s method uses an algorithm [170] to automatically detect so-called single entry single exit fragments. To recommend process model fragments, Wieloch et al. leverage the semantic annotations of processes according to the Business Process Modeling Ontology [18]. This ontology is part of a specialized approach to model business processes, encompassing knowledge about organizational context, workflow activities, and Semantic Web services.

The previously described approaches share two fundamental components: one that *identifies* process model fragments in a repository of business process models, and another that *matches* the process model fragments with the business process model under development. The approaches by Mazanek et al. [102, 103] are an exception to this, as they are purely syntax-based, i.e., the elements in the recommended process model fragments are not labeled. As such, these approaches offer limited modeling support only. The other approaches differ in the automation level of the identification and matching components. While the graph-mining approaches [19,31,95] automate both components, the tag-based approaches [65,81] require manual work for the identification component. The approaches based on semantic representations of business process models [66,67,186] have the limitation that these representations are crucial for the matching component. However, semantic annotations are typically not available in practice, or additional manual effort is required to obtain them.

The automation level of the two components has a direct effect on the feasibility of the approaches for large organizations, which may possess repositories containing hundreds or even thousands of business process models [135]. Since the tag-based approaches and the approaches based on semantic representations require manual work, they are not feasible for organizations with repositories containing hundreds of business process models. The graph-mining approaches do not require manual work to identify and match process model fragments. However, as demonstrated by Wang et al. [177], they cannot deal with large repositories of process models from practice.

Table 3.1 summarizes our survey of approaches for recommending process model fragments.

3.3.2 Recommending Activities

In this thesis, we are specifically interested in recommendation approaches of the second category, i.e., those that recommend *activities*, or, more precisely, labels for activity nodes. In contrast to approaches that suggest process model fragments, activity-recommendation approaches are used iteratively to support users modeling business processes in an interacting way.

According to our notion of business process models as captured in Definition 1, all nodes, not just activity nodes, can be labeled. However, nodes other than activity nodes, such as control-flow nodes, often have empty labels, which is why the focus is usually on activity (node) labels. If needed, approaches that recommend activities can be extended to recommend labels for all nodes in a model.

To put it formally, activity recommendation targets a situation in which a process model under development contains exactly one activity node that has not yet received a label[1], such as seen in Figure 1.1. We refer to such a model as an *incomplete process model*:

[1]Note that process model nodes may have empty labels ($\lambda(n) = \epsilon$), such as the XOR-join in Figure 3.2, which is different from a node being unlabeled ($\lambda(n) = \perp$).

Definition 2 (Incomplete process model) *An incomplete process model* $M_I = (N, A, E, \tau, \lambda, \hat{n})$ *is a process model* (N, A, E, τ, λ) *that has exactly one unlabeled activity node* $\hat{n} \in A$ *with a non-empty preset, i.e.,* $\lambda(n) \in \mathcal{L}$ *is given for all* $n \in N \setminus \{\hat{n}\}$*,* $\lambda(\hat{n}) = \perp$ *and* $\bullet\hat{n} \neq \emptyset$*.*

Given an incomplete process model M_I, *activity recommendation* sets out to suggest one or more suitable labels for the unlabeled activity node $\hat{n}$.

Compared to the research focused on recommending process model fragments, which we discussed earlier, a relatively smaller body of work is dedicated to activity recommendation. Existing approaches employ standard machine learning, embeddings or pre-trained language models, as we explore in further detail below.

Jannach et al. [72, 73] propose a variety of approaches that are based on different concepts from standard machine learning, i.e., *k*-nearest neighbors, co-occurrence, and frequently linked elements. Originally designed to support users in the field of data analysis workflows, these approaches are evaluated using a dataset of over 6,600 machine learning workflows. However, as we will show in Chapter 5, the approaches can also be used to suggest activities for business process models.

For their evaluation, Jannach et al. divide the dataset into training and test subsets and employ different procedures to simulate recommendation cases from complete models in the test set. The *given-n* procedure reflects a cold-start setting, where limited context is available for the recommendation. On the other hand, the *leave-one-out* and *hide-last-two* procedures maintain nearly complete models, thus providing a lot of information for the recommendation. As metrics, they use recall and mean reciprocal rank.

The embedding-based approach RLRecommender was proposed by Wang et al. [177]. Based on the Knowledge Graph Embedding (KGE) model TransE [11], their approach extracts relations between activities of the business process models and embeds both activities and relations into a continuous low-dimensional space. The embeddings and their distances in the space are then used to recommend activities. However, a limitation of this approach is that it only considers one preceding activity in the process model under development, along with its inter-relation with the unlabeled activity, as a context for the recommendation. In Chapter 5, we will demonstrate that this leads to comparatively low performance of RLRecommender.

To evaluate their approach, Wang et al. employ a dataset consisting of 23,576 models, which they split into training, validation and test sets. They assess the performance of their approach for each position in the test set models, using one preceding activity, and measure the results using hit rate and F1 as metrics.

Goldstein et al. [48] propose an approach that uses the pre-trained language model Universal Sentence Encoder (USE) [20] to leverage semantic similarity of sequences in business process models. Their approach involves generating sequence embeddings using USE and computing cosine similarities of the embeddings to compare sequences from the process model under development with those in the available repository of business process models. The label that followed the most similar sequence is then recommended. Although Goldstein et al.'s approach represents a first step towards the use of transfer-learning techniques from NLP for activity recommendation, it is limited to the use of a pre-trained language model without fine-tuning.

For their evaluation, Goldstein et al. employ small datasets consisting of nine to 40 business process models. They evaluate their approach on each dataset separately in a leave-one-group-out cross validation, where each process model is evaluated against the rest of the models that is used for training. Since the datasets each consist of models describing the same or highly similar processes, the evaluation cases closely resemble the training examples. In the course of the experiments in Chapter 8, we will demonstrate that Goldstein et al.'s approach is not feasible for recommending activities when dealing with extensive process model repositories. This is due to the hundreds of thousands of cosine similarities that need to be calculated in this case, resulting in significant runtime costs. Consequently, recommendation generation times may reach ten minutes or more, which we consider unacceptable in real-world applications, where repositories may contain thousands of business process models [34]. For the experimental evaluation, Goldstein et al. simulate recommendations cases from the complete models in the datasets. However, the procedure for this simulation is not clearly explained in detail. To evaluate the quality of recommendations, they employ two types of metrics. The first type includes standard recommender system metrics, namely precision and recall. The second type measures the quality of predictions that are semantically similar to the ground truth. To assess this, the metrics BLEU, METEOR, and cosine similarity are used, which quantify the similarity between two terms. However, their work lacks details about how the metrics can be used to evaluate a list containing multiple recommendations rather than a single recommendation.

The described approaches can be reviewed based on the semantic information in the business process models used for recommending activities. In general, activity-recommendation approaches can leverage both the formal and the natural language semantics in business process models. Formal semantics are valuable for understanding structural relationships between activities. Considering them is crucial for identifying patterns, such as activities that frequently co-occur or follow each other. Natural language semantics provide information about the meaning and context of each activity, which allows generalizing activity patterns in the repository. Approaches that incorporate natural language semantics can, for example, recognize the similarity between activities such as *send invoice* and *send bill*, and use this additional information to make more informed recommendations. Among

Table 3.2: *Overview of approaches for recommending activities*: For each approach, we show its type, its capability to incorporate formal or natural language semantics, its ability to generate labels beyond the vocabulary of the available model repository, and its applicability on large repositories of business process models.

Approach type / Authors	**Formal semantics incorporated**	**Natural language semantics incorporated**	**Labels beyond repository vocabulary**	**Large process model repositories**
Based on standard machine learning				
Jannach et al. [72, 73]	✓	✗	✗	✓
Embedding-based				
Wang et al. [177]	✓	✗	✗	✓
Based on language model				
Goldstein et al. [48]	✓	✓	✗	✗

the activity-recommendation approaches discussed above, only the approach based on a pre-trained language model considers natural language semantics in addition to formal semantics. The embedding-based approach and the methods based on standard machine learning solely rely on activity patterns learned from the given repository, disregarding the meaning of activities. This makes these approaches inapplicable in situations where a process model under development entirely consists of activities that were not included in the repository's models, since the extracted patterns cannot be used to make a recommendation in these cases.

A limitation that all described activity-recommendation approaches have in common is that they can only provide recommendations in the form of labels contained in the repository of business process models at hand. In other words, they are restricted to the vocabulary of the model repository. This leads to poor recommendations for process models that strongly differ from those in the repository.

We summarize our survey of approaches recommending activities in Table 3.2.

3.3.3 Further Support Approaches

Beyond the scope of guidelines and the recommendation of process model fragments or activities during modeling time, there are several other ways to support business process modeling [41], for example, by detecting lexical ambiguity [126] or violations of naming conventions [89] in business process models. In general, several works (e.g., [108], [149], [91]) are concerned with the analysis of activity labels to support business process modeling. Other approaches focus on the executability of business process models and support the identification of relevant services that are needed to execute the model [12, 21]. However, these approaches are designed to refactor business process models rather than to support modelers during modeling time.

3.4 Summary

In this chapter, we provided an introduction to business process modeling as a crucial subfield within the broader scope of business process management. We emphasized the importance of having adequate support for business process modeling and explored related research. Our analysis reveals the following research gaps that motivate our research:

- **Large process model dataset.** The evaluation of support approaches for business process modeling is often limited to small datasets. Moreover, some approaches are not even feasible when dealing with large repositories of business process models, especially those that recommend process model fragments. One of the reasons for these issues is that researchers rarely have access to large collections of process models from practice. As part of our work, we aim to bridge this gap by publishing the–at the time of writing–largest publicly available collection of business process models. We present this dataset in Chapter 4.
- **Explainable approach.** The activity-recommendation approaches that are based on embeddings or a pre-trained language model are highly sophisticated but lack transparency. This lack of transparency makes it difficult for the user to decide between the provided recommendations. To address this gap, we propose a rule-based approach tailored for activity recommendation in Chapter 5. Rule-based approaches are explainable by design, which helps to improve the transparency and trustworthiness of recommendation approaches [190].
- **Existing knowledge graph completion approaches.** While there is an activity-recommendation approach that employs a KGE model, it lacks the ability to integrate more than one preceding activity as a context for the recommendation. Therefore, in Chapter 6, we explore how existing KGE models can be applied to the activity-recommendation problem such that the entire process model under development is considered as recommendation context. We also explore the potential of an existing rule-based knowledge graph completion method for this purpose. Moreover, we compare the performance of these methods to the performance of approaches specifically designed for activity recommendation.
- **Natural language semantics.** Natural language semantics contained in business process models are important for activity recommendation, since they provide valuable information about the meaning and context of activities. However, despite their relevance, there is no existing approach that effectively incorporates natural language semantics while also being applicable on large repositories of

business process models. In Chapters 7 and 8, we therefore explore different ways to consider natural language semantics for activity recommendation, which are feasible also when dealing with large model repositories.

- **Labels beyond repository vocabulary.** The ability to process and recommend activity labels beyond the vocabulary of a given process model repository can be highly beneficial, particularly when the process model under development deviates significantly from the process models in the repository. However, existing approaches are unable to recommend labels that are not already present in the repository. To address this limitation, we introduce a novel transformer-based approach in Chapter 8.
- **Evaluation.** To make experimental findings applicable in broader contexts, it is crucial to conduct diverse experiments aligned with practical situations. However, current activity-recommendation research falls short in this aspect. In our evaluations in Chapters 5, 6, 7, and 8, we conduct comprehensive experiments that enhance existing evaluation frameworks in terms of metrics, simulations of recommendation cases, and availability or absence of similar business process models in the repository.

Chapter 4

The SAP-SAM Dataset

In this chapter, we present the *SAP Signavio Academic Models (SAP-SAM)* dataset, which is the–at the time of writing–largest publicly available collection of business process models and related business models, e.g., of business decisions. As part of our work, we have published the SAP-SAM dataset[1] and conducted an initial analysis of its contents. To assist students and researchers interested in SAP-SAM, we provide source code for querying the dataset, along with the analysis source code used in this chapter[2].

This chapter is organized as follows. Section 4.1 describes the origins of SAP-SAM, explaining how the models contained in the dataset have been collected. Subsequently, Section 4.2 provides an exploration of the properties of SAP-SAM. Section 4.3 discusses potential applications of the dataset and Section 4.4 its limitations. Finally, Section 4.5 examines datasets related to SAP-SAM.

4.1 Origins

SAP-SAM contains 1,021,471 business process models and related business models that were created using the software-as-a-service platform of the SAP Signavio Academic Initiative (SAP-SAI)[3]. SAP-SAI allows academic researchers, teachers, and students to create, execute, and analyze process models, as well as related business models. The usage of SAP-SAI is restricted to non-commercial research and education. Upon registration, users consent that the models that they create can be made available for research purposes, either anonymized or non-anonymized. SAP-SAM contains those models for which users have consented

[1]SAP-SAM is publicly available at `https://zenodo.org/record/7012043`.

[2]The source code can be found at `https://github.com/signavio/sap-sam`.

[3]See `https://www.signavio.com/academic-and-research-alliances/`.

to non-anonymized sharing. Still, anonymization scripts were run to post-process the models, in particular to remove e-mail addresses, student registration numbers, and—to the extent possible—names.

The models in SAP-SAM were created between July 2011 and (incl.) September 2021 by a total of 72,996 users, based on a count of distinct user IDs that are associated with the creation or revision of a model. The models were extracted from the MySQL database of SAP-SAI and are in SAP Signavio's proprietary JSON-based data format. The total number of models contained in SAP-SAM includes vendor-provided example models, which are automatically added to newly created workspaces, i.e., process repositories that users register. About 470,000 models in the dataset have the name of an example model[4]. However, given that the example models could have been edited or even completely replaced in the automatically added model files, this number can only be a rough estimate of the number of example models in the dataset.

4.2 Properties

The SAP-SAM dataset consists of models in various modeling and natural languages, with differing levels of complexity. In this section, we provide an overview of the key properties of SAP-SAM.

Modeling languages. SAP-SAM contains models in the following modeling languages:

- Business Process Model and Notation (BPMN): BPMN is a standardized language for modeling business processes [118]. SAP-SAM distinguishes between BPMN process models, collaboration models, and choreography models, and among BPMN process models between BPMN 1.1 and BPMN 2.0 models.
- Decision Model and Notation (DMN): DMN is a standardized language for modeling business decisions, complementing BPMN [120].
- Case Management Model and Notation (CMMN): CMMN is an attempt to supplement BPMN and DMN with a modeling language that focuses on agility and autonomy [119].
- Event-driven Process Chain (EPC): EPC [141] is a process modeling language that enjoyed substantial popularity before the advent of BPMN.
- Unified Modeling Language (UML): UML is a language used to describe software (and other) systems. It is subdivided into class and use case diagrams.
- Value Chain: A value chain is an informal modeling language for sketching high-level end-to-end processes and process frameworks.

[4]The names of the example models provided by the SAP-SAI system can be found in Appendix A.

- ArchiMate: ArchiMate is a language for the integrated modeling of information technology and business perspectives on large organizations [86].
- Organization Chart: Organization charts are tree-like models of organizational hierarchies.
- Fundamental Modeling Concepts (FMC) Block Diagram: FMC block diagrams support the modeling of software and IT system architectures.
- (Colored) Petri Net: Petri nets [125] are a popular mathematical modeling language for distributed systems and a crucial preliminary for many formal foundations of BPM. In SAP-SAM, colored Petri nets [74] are considered a separate modeling language.
- Journey Map: Journey maps model the customer's perspective on an organization's business processes.
- Yet Another Workflow Language (YAWL): YAWL is a language for modeling the control flow logic of workflows [168].
- jBPM: jBPM models allowed for the visualization of business process models that could be executed by the jBPM business process execution engine before the BPMN 2.0 XML serialization format existed. However, recent versions of jBPM rely on BPMN 2.0-based models.
- Process Documentation Template: Process documentation templates support the generation of comprehensive PDF-based process documentation reports. These templates are technically a model language, although they may practically be considered a reporting tool instead.
- XForms: XForms is a (dated) standard for modeling form-based graphical user interfaces [17].
- Chen Notation: Chen notation diagrams [22] allow for the creation of entity-relationship models.

Figure 4.1 depicts the number of models in the different modeling languages in the dataset, as well as the according percentages (in brackets). We aggregate languages which are used for less than 100 models, respectively, into *Other*: Process Documentation Template (86 models), jBPM 4 (76 models), XForms (20 models), and Chen Notation (3 models). The primarily used modeling language is BPMN 2.0, which confirms that it is the de-facto standard for modeling business processes [24]. Therefore, we will focus on BPMN 2.0 models as we examine further properties.

Natural languages. Since SAP-SAI can be used by academic researchers, teachers and students all over the world, the models in SAP-SAM are created using different natural languages. For example, SAP-SAM includes BPMN 2.0 models in 41 different languages. Figure 4.2 shows the ten most frequently used languages for BPMN 2.0 models. Note that the vendor-provided example models, which are

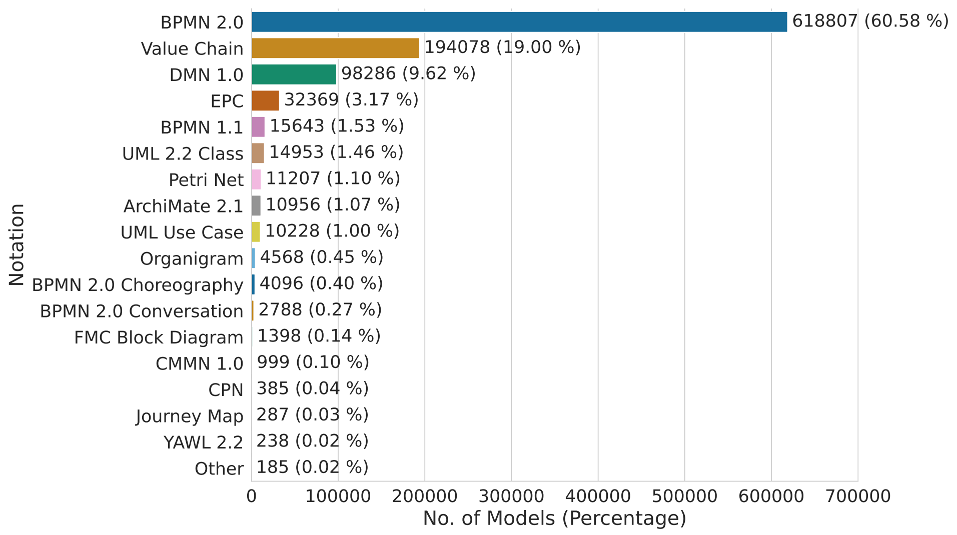


Figure 4.1: Usage of different modeling languages

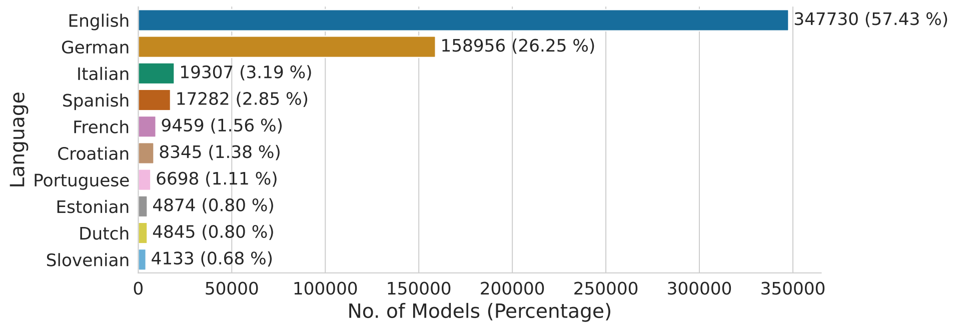


Figure 4.2: Usage of different natural languages for BPMN 2.0 models

added to newly created workspaces, exist in English, German, and French. When a SAP-SAI workspace is created, the example models added to it are in German or French if the language configured upon creation is German or French, respectively; otherwise, the example models are in English. This contributes to the fact that more than half of the BPMN 2.0 models (57.43 %) are in English.

Elements types. Figure 4.3 illustrates the occurrence frequency of different element types in the BPMN 2.0 models of SAP-SAM. It can be recognized that the element types are not equally distributed, which confirms the findings of prior research [115]. The number of models that contain at least one instance of a particular element type is much higher for some types, e.g., *sequence flow* (98.88 %) or *task* (98.11 %), than for others, e.g., *collapsed subprocess* (25.23 %) or *start*

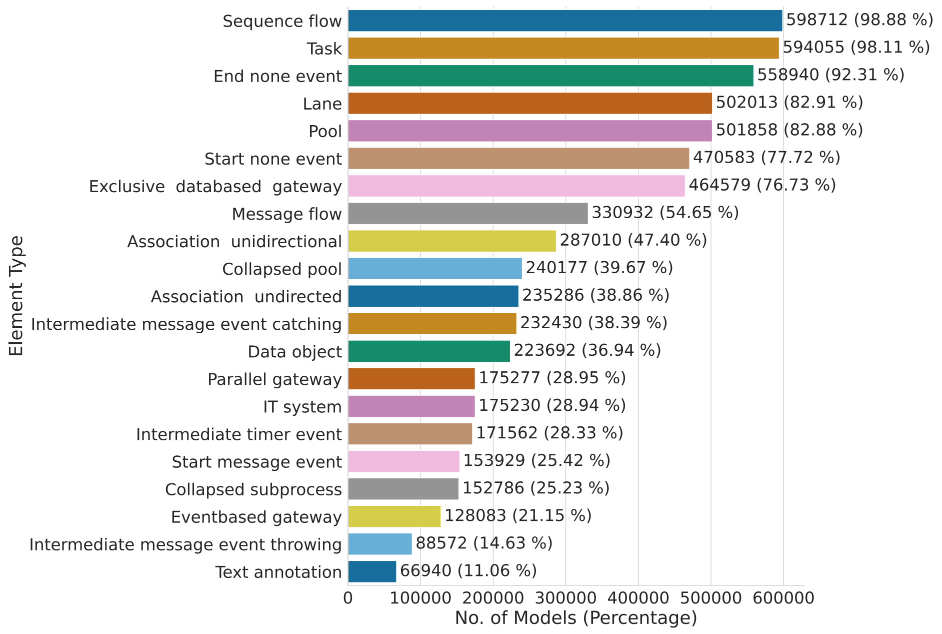


Figure 4.3: Occurrence frequency of different BPMN 2.0 element types

Table 4.1: Statistics of the number of elements per BPMN 2.0 model by type (grouped)

Element type groups	Mean	Std	Min	25%	50%	75%	Max
Activities	8.6	8.4	0	4	7	10	1,543
Events	5.2	5.1	0	2	5	6	157
Gateways	3.7	4.4	0	2	3	4	303
Connecting Objects	23.1	21.8	0	14	20	25	2,066
Swimlanes	3.8	2.6	0	3	4	5	227
Data Elements	1.3	3.4	0	0	0	2	266
Artifacts	0.9	4.0	0	0	0	1	529

message event (25.42 %). Note that Figure 4.3 only includes element types that are used in at least 10 % of the BPMN 2.0 models. More than 30 element types are used by less than 1 % of the models. On average, a BPMN 2.0 model in SAP-SAM contains 11.3 different element types (median: 11) and 46.7 different elements, i.e., instances of element types (median: 40).

Table 4.1 shows the number of elements per model by type. For a compact representation, we aggregate similar element types by arranging them into groups. The employed mapping from element types to element type groups can be found in Table A.1 in Appendix A. On average, connecting objects, which include associations and flows, make up the largest proportion of the elements in a model.

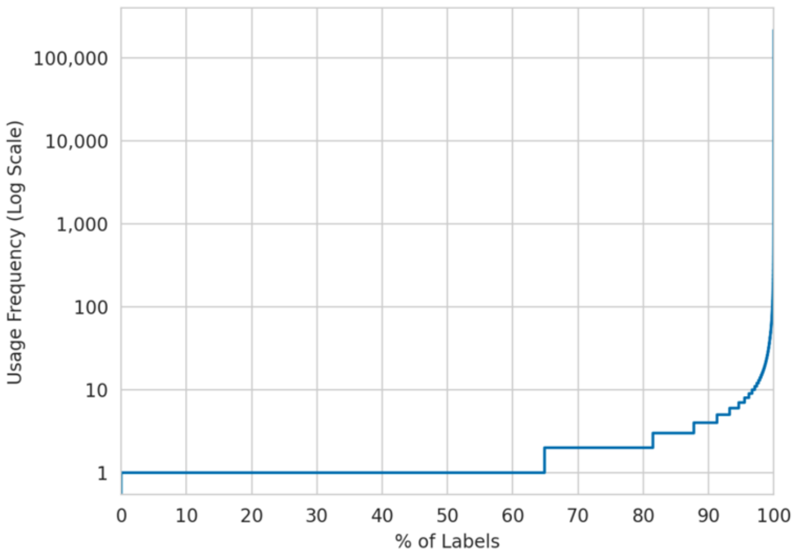


Figure 4.4: Empirical cumulative distribution function of the label usage frequency in BPMN 2.0 models

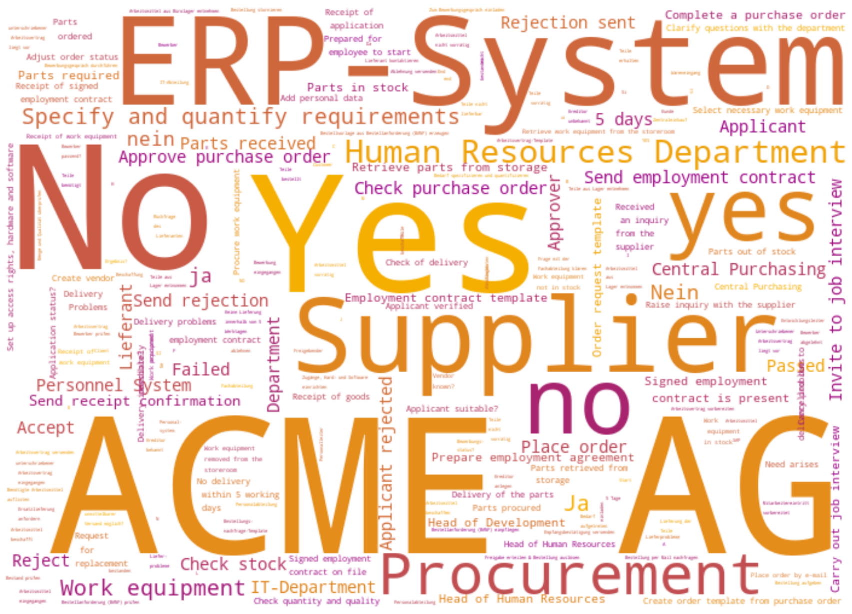


Figure 4.5: Word cloud of the labels contained in BPMN 2.0 models

Labels. In a BPMN 2.0 model, every element can be labeled by the modeler. This leads to a total of 2,820,531 distinct labels for the 28,293,762 elements of all BPMN 2.0 models in SAP-SAM. Figure 4.4 illustrates the distribution of label usage frequencies. Specifically, the figure shows the empirical cumulative distribution function, which indicates the cumulative proportion of labels corresponding to each usage frequency. The distribution of label usage frequencies is highly skewed, which becomes even more evident when considering the log scale used for the visualization. Approximately 65 % of the labels are used only once, meaning they are assigned to just one element in all BPMN 2.0 models. Conversely, the nearly vertical line on the right side of the graph reveals that a small number of labels are used frequently. In fact, 10 % of the labels are used for around 74 % of the elements in the BPMN 2.0 models. The uneven distribution of label usage can be partially attributed to vendor-provided examples in the dataset, as labels from these example processes appear with high frequency.

To give an insight into the most frequently used labels, Figure 4.5 shows a word cloud of the labels contained in the BPMN 2.0 models of SAP-SAM.

4.3 Applications

Large process model collections like SAP-SAM are a valuable and critical resource for research. Process models from practice codify organizational knowledge about business processes and methodical knowledge about modeling practices. Both types of knowledge can be used by research, for example, for deriving recommendations for the design of process models. In addition, large process model collections are required for evaluating newly developed algorithms and techniques regarding their applicability in practice.

To illustrate the potential value of SAP-SAM for the BPM community, the following list describes possible application scenarios that we consider to be particularly relevant. It is neither prescriptive nor comprehensive; researchers can use SAP-SAM for many other purposes.

Knowledge Generation. Process models depict business processes, codifying knowledge about the operations within organizations. This knowledge can be extracted and generalized to a broader context. Hence, SAP-SAM can be considered as a knowledge base to generate new insights into the contents and the practice of organizational modeling. Example applications include:

- Reference model mining [131]: Reference models provide a generic template for the design of new processes in a certain industry. They can be mined by merging commonalities between existing processes from different contexts into a new model that abstracts from their specific features. By applying this technique to subsets of similar models from SAP-SAM, we can mine new reference models for process landscapes or individual processes, including, e.g., the organizational perspective. Similarly, we could identify, analyze, and compare different variants of the same process.

- Identifying modeling patterns [42]: Process model patterns provide proven solutions to recurring problems in process modeling. They can help in streamlining the modeling process and standardizing the use of modeling concepts. A dataset like SAP-SAM, which contains process models from many different modelers, provides an empirical foundation both for finding new modeling patterns and for validating existing ones. This also extends to process model antipatterns, i.e., patterns that should be avoided, as well as modeling guidelines and conventions.

Modeling Assistance. The modeling knowledge that is codified in SAP-SAM can also be used for automated assistance functions in modeling tools. Such assistance functions support modelers in creating or updating process models, accelerating and facilitating the modeling process. However, for the development of innovative assistance functions based on novel machine learning techniques, a substantial training dataset is typically required to generate meaningful and valuable results. With its large amounts of contained modeling structures and labels, SAP-SAM offers such a substantial training set, for example, for the following applications:

- Activity recommendation: By providing recommendations on possible labels for activities that are newly inserted by a modeler in a process model under development, activity recommendation can speed up modeling and facilitate consistency of the terms and modeling patterns that are used by an organization. As we will demonstrate in Chapter 8, SAP-SAM can be used to train and evaluate methods for this purpose.
- Automated abstraction techniques [178]: One important function of BPM is process model abstraction, i.e., the aggregation of model elements into less complex, higher-level structures to enable a better understanding of the overall process. Such an aggregation entails the identification and assignment of higher-level categories to groups of process elements. SAP-SAM can provide the necessary training data for an NLP-based automated abstraction.

Evaluation. Managing large repositories of process models is a key application of BPM [35]. Researchers have developed many different approaches to assist organizations with this task. To make these approaches as productive as possible, they need to be tested on datasets that are comparable to those within organizations.

Since SAP-SAM goes well beyond the size of related datasets, it can be used for large-scale evaluations of existing process management approaches on data from practice. Examples for these approaches include process model querying [127], process model matching [4], and process model similarity [34].

4.4 Limitations

Considering the nature of SAP-SAM as a model collection that has been generated by academic researchers, teachers, and students, the following limitations must be considered:

- SAP-SAM contains models that exist multiple times, either as direct duplicates (copies) or as very similar versions. This includes vendor-provided example models or standard academic examples that are frequently used in academic teaching and research. The existence of these models can be used to evaluate variant identification and fuzzy matching approaches in process querying, but it negatively affects the diversity, i.e., the breadth of the dataset.
- Models contained in SAP-SAM may be of low technical quality, in particular the models that are created by process modeling beginners, i.e., early-stage students, for learning purposes. Although it can be interesting to analyze the mistakes or antipatterns in such models, the mistakes that students make are most likely not representative of mistakes made by process modeling practitioners.
- Since many of the models in SAP-SAM have most likely been created for either teaching, learning, or demonstrating purposes, they presumably present a simplistic perspective on business processes. Even when assuming that all researchers, teachers, and students are skilled process modelers and have a precise understanding of the underlying processes when modeling, the purpose of their models is typically fundamentally different from the purpose of industry process models. Whereas academic models often emphasize technical precision and correctness, industry models usually focus on a particular business goal, such as the facilitation of stakeholder alignment.

Let us note that this list may not be exhaustive; in particular, limitations that depend on a particular use case or evaluation scenario need to be identified by researchers who use this dataset. Still, it is also worth highlighting that the partially messy nature of the model collection reflects the reality of industrial data science challenges, in which a sufficiently large amount of high-quality data (or models) is typically not straightforwardly available [69]; instead, substantial efforts need to be made to separate the wheat from the chaff, or to isolate use-cases in which

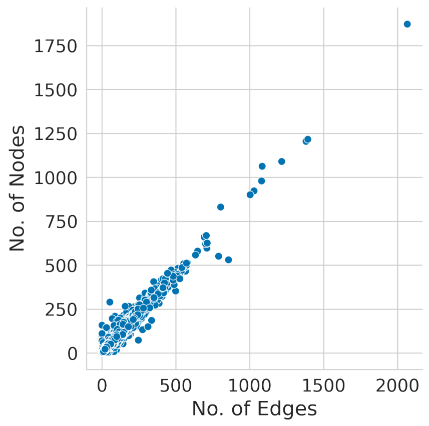


Figure 4.6: Correlation of the number of nodes and edges in BPMN 2.0 models

the flaws in the data do not have an adverse effect on business value, or any other undesirable organizational or societal implications. However, most process models go beyond *A*-*B*-*C* toy examples from exercises and the overall SAP-SAM dataset is of sufficient diversity, relevance and quality for facilitating research.

When using SAP-SAM for academic research purposes, it typically makes sense to filter it, i.e., to reduce it to a subset of models that satisfy desirable properties. In the context of this thesis, we have found the following filters to be useful:

- We filter out potential vendor-provided example models by excluding any models with names that match those of the example models provided by the SAP-SAI system during workspace creation.
- We consider only process models with English labels.
- We remove process models with either a small or a large number of elements. As can be expected for BPMN 2.0 models and is shown in Figure 4.6, the number of nodes and the number of edges in a model are highly correlated. Hence, it is sufficient to filter based on the number of nodes.
- We exclude process models where the element labels have an average length of less than, for example, three characters to ensure that only models with useful labels are included.

4.5 Related Datasets

While there are several existing process model collections, they are relatively small when compared to SAP-SAM. For example, the *BPMN Artifacts Dataset* consists of 79,713 BPMN models that were mined from 18,534 open source projects on GitHub.com [164]. Another collection, *Camunda BPMN for research* [5], contains 3,721 models that were created during BPMN training sessions provided by Camunda. The *hdBPMN dataset* [139] includes 704 handwritten BPMN 2.0 models that can be parsed as BPMN 2.0 XML. Further, *RePROSitory* [26] is an open collection of business process models that allows users to contribute their own data. At the time of writing, RePROSitory contains approximately 700 models.

In the process mining community, the *BPI challenge datasets*, e.g., the BPI challenge 2020 [169], have become important benchmarks. Unlike SAP-SAM, these datasets consist of *event logs* from practice. Therefore, the applications of the BPI challenge datasets only partially overlap with those of SAP-SAM.

Some models included in SAP-SAM have already been published [184] as Business Process Management Academic Initiative collection. The previously published dataset contains 29,810 models that were collected over a shorter period of time. In contrast to SAP-SAM, this dataset explicitly contains several revisions, i.e., versions, of process models. In the following chapters, we will either use this subset of SAP-SAM or the whole SAP-SAM dataset.

[5]https://github.com/camunda/bpmn-for-research

Chapter 5

Rule-based Activity Recommendation

In this chapter, we introduce a rule-based approach to tackle the activity-recommendation task. Originating from the field of inductive logic programming [28], rule-based approaches have proven to be competitive in applications such as knowledge graph completion [106]. One of the key advantages of rule-based over black-box approaches is their inherent explainability, which enhances the transparency, trustworthiness, and user satisfaction of a rule-based recommendation approach [190].

As shown in Figure 5.1, our rule-based approach consists of two main phases: rule learning and rule application. First, *rule learning* derives logical rules that capture activity inter-relations from the process models in a provided repository. Our rule learner is specifically tailored for activity recommendation, supporting a process-oriented language. Second, *rule application* employs the learned rules to recommend the most suitable labels for a new activity node in a process model under development, i.e., the recommended activity. The rule-learning phase has to be performed only once, for a given repository of process models, whereas the rule-application phase is repeated throughout the process-modeling task to iteratively provide activity recommendations.

The remainder of this chapter is organized as follows. Section 5.1 describes the process model abstraction, including the behavioral relations, used in the approach. Sections 5.2 and 5.3 cover the two main phases of the approach, rule learning and rule application, respectively. Section 5.4 presents the results of an extensive experimental evaluation, followed by conclusions and limitations in Section 5.5.

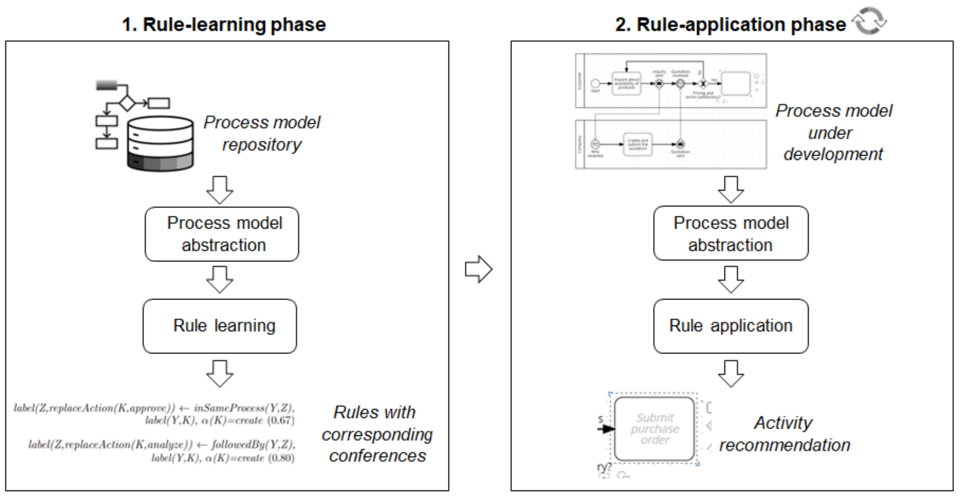

1. Rule-learning phase
Process model repository
Process model abstraction
Rule learning
label(Z,replaceAction(K,approve)) ← inSameProcess(Y,Z), label(Y,K), α(K)=create (0.67)
label(Z,replaceAction(K,analyze)) ← followedBy(Y,Z), label(Y,K), α(K)=create (0.80)
Rules with corresponding conferences
2. Rule-application phase
Process model under development
Process model abstraction
Rule application
Submit purchase order
Activity recommendation

Figure 5.1: Illustration of the two main phases of our rule-based approach: 1. Rule learning, and 2. Rule application

5.1 Behavioral Abstraction

In this section, we provide essential details on the employed formalization of business process models, which includes the behavioral relations that our recommendation approach uses as a basis.

Our work is not limited to any specific modeling language for capturing business processes, such as Petri nets or BPMN. Instead, we employ the generic representation (N, A, E, τ, λ) of business process models described in Definition 1. However, our rule-based approach only considers activity nodes in business process models, i.e., $N = A$, and does not distinguish between various types of activity nodes, making τ redundant. To represent the execution flow of activities without control-flow nodes, we introduce a function $\rho : E \to \mathcal{P}(\mathcal{R})$ that associates an edge with a set of behavioral relation types (e.g., *directly follows*), where $\mathcal{R}$ denotes the set of all behavioral relation types, and $\mathcal{P}(\mathcal{R})$ denotes the power set of $\mathcal{R}$. Considering these specifics of our rule-based approach, we change the tuple notation (N, A, E, τ, λ) of a business process model from Definition 1 to (N, E, λ, ρ).

To convert a business process model in an arbitrary modeling language to the generic representation (N, E, λ, ρ), we use an abstraction procedure. This procedure has several degrees of freedom: We might, for example, drop (or keep) certain types of nodes and have to select the types of behavioral relations that we use and assign to edges via the function ρ. In the following, we use a transformation from Petri nets to the generic process model representation (N, E, λ, ρ) as an illustration of the abstraction procedure and the various relations that our approach can use as basis. However, similar abstraction procedures can be derived for other modeling languages. For instance, BPMN models can first be translated into Petri nets [36] before applying the abstraction approach.

Given a Petri net, we consider transitions, which correspond to activities, as nodes in (N, E, λ, ρ) and omit its places. The label function λ results from the transitions' labels. Then, for any pair of nodes $m, n \in N$, we decide if we create a directed edge $e = (m, n) \in E$ and which relation types $\rho(e)$ from a set $\mathcal{R}$ to assign to this directed edge. For this procedure, we follow Wang et al. [177], who propose three abstraction strategies, based on different sets of behavioral relations. We refer to [177] for details and here stick to an intuitive explanation.

- **Directly-follows abstraction:** This abstraction strategy only considers which activities may follow each other during process execution, captured in the *followedBy* relation. Formally, if a node m can be directly followed by a node n, we add an edge $e = (m, n)$ with $\rho(e) = \{$*followedBy*$\}$. Naturally, this strategy loses part of the semantics expressed in the original Petri net. For instance, it does not distinguish between transitions that exclude each other (XOR split) and those that can be executed concurrently (AND split).
- **Causal abstraction:** The second strategy reduces the abstraction loss by distinguishing between *alwaysCausal* and *sometimesCausal* relations, and their inverse counterparts. A pair of activities (m, n) is in the *alwaysCausal* relation if any occurrence of m is always followed by an occurrence of n, whereas the *sometimesCausal* relation applies if this is sometimes the case (due to an XOR-split in the process). Conversely, m and n are in the *inverseAlwaysCausal* relation if any occurrence of n is always preceded by an occurence of m, while the *inverseSometimesCausal* relation holds if this is sometimes the case (due to an XOR-join in the process). Since this distinction is assymetric, e.g., an $alwaysCausal$ relation does not guarantee an $inverseAlwaysCausal$ relation between two activities, we assign the forward and the inverse relation between m and n to the edge $e = (m, n)$, e.g., $\rho(e) = \{$*alwaysCausal,inverseSometimes-Causal*$\}$.
- **Causal and concurrent abstraction:** Finally, the third strategy introduces additional relations that can be used to describe types of concurrency between activities, on top of the aforementioned *causal* ones. These relations are called *alwaysConcurrent*, *sometimesConcurrent*, and *neverConcurrent*, reflecting whether two activities can, must, or must not occur concurrently.

In the remainder, we use $\mathcal{R}^X$ to denote a set of relation types that has been used in an abstraction strategy X. For example, Figure 5.2 shows a BPMN 2.0 model (top)

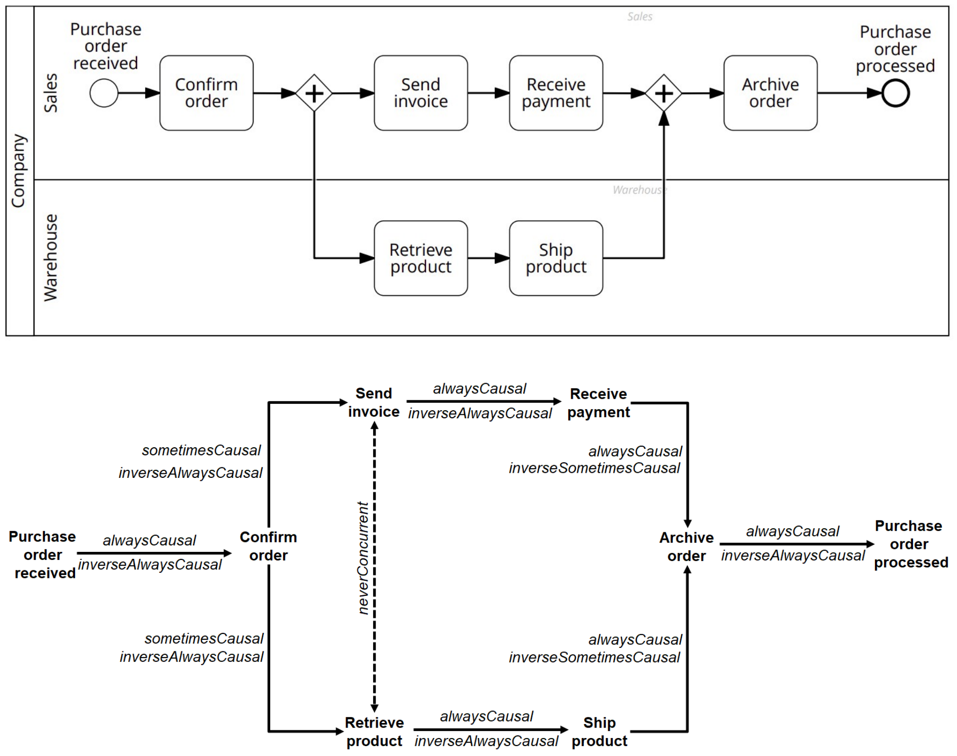


Figure 5.2: An example BPMN 2.0 model (top) and its abstracted version using the relation types $\mathcal{R}^{causal+concurrent}$ (bottom)

and its abstracted version based on $\mathcal{R}^{causal+concurrent}$(bottom). Although the abstracted version of the business process model omits from some details, the overall structure and sequence of activities is preserved with respect to the original model. Note that a $\mathcal{R}^{causal}$-model can be obtained by omitting the dashed edges from Figure 5.2, while a $\mathcal{R}^{followedBy}$-model corresponds to the $\mathcal{R}^{causal}$-model in which all relation types are replaced by *followedBy*.

5.2 Rule Learning

During the rule-learning phase, we generate logical rules that capture regularities in the use of activity labels within a given repository of business process models $\mathcal{B}$.

One potential solution to the rule-learning problem involves defining a refinement operator that guides the learning algorithm to specialize a rule $head \leftarrow body$ by incrementally adding atoms to the rule's body. This approach is commonly implemented in classic ILP systems, as well as in more recent top-down methods, such as AMIE [46]. The refinement operator's definition implicitly encompasses a wide range of possible rules that can be constructed through its repeated application. However, we are proposing a different approach to maintain full control over the types of generated rules.

Specifically, our approach employs a set of rule patterns, i.e., templates, from which we derive a set of rules that apply to the repository $\mathcal{B}$. The use of rule templates ensures that we focus on rules that are useful for activity recommendation in business process modeling, which makes the rule learning overall more targeted than the use of already available rule-learning systems.

In the following, we first introduce the employed rule templates of our approach (Section 5.2.1), before we describe how instantiations of these templates are learned from the provided model repository (Section 5.2.2).

5.2.1 Rule Templates

For the definition of the rule templates, we first need to describe the abstracted business process models in terms of logical formulas, i.e. atoms.

Rule atoms. We translate each business process model $B = (N, E, \lambda, \rho) \in \mathcal{B}$ as follows:

- For each node $n \in N$ we add an atom *label*$(n, \lambda(n))$, e.g., *label(n,check purchase order)*, to express that n has the label $\lambda(n)$.
- For each edge $e = (m, n) \in E$ and each relation type $r \in \rho(e)$ we add an atom $r(m, n)$ that captures the type of relation between m and n, e.g.,*followedBy(*m, n*)* or *alwaysCausal*(m, n).
- For each pair of nodes $m \neq n \in N$ we add the atoms *inSameProcess*(m, n) and *inSameProcess*(n, m) to express that m and n appear in the same business process model.

Given a set of relations $\mathcal{R}^X$, we thus use $|\mathcal{R}^X| + 2$ ($+2$ for *inSameProcess* and *label*) binary predicates to capture the structure of the process models in the repository $\mathcal{B}$.

Rule templates and instantiations. When defining our templates, we use h, j, k and l to refer to placeholders for labels of an activity, e.g., $l =$ *check purchase order*. An *instantiation of a rule template* is a rule in which those placeholders are all substituted by activity labels from the universe of node labels $\mathcal{L}$. The variables W, X, Y and Z in the templates stand for activity nodes. A *rule grounding* is an instantiation of a rule template, where these variables are replaced by concrete nodes, e.g., $W = a_1, X = a_2, Y = a_3, Z = a_4 \in N$.

In our approach, we use a special form of *horn rules*. Particularly, we are interested in rules that have the form *label(Z,l)* $\leftarrow \ldots$, which are rules that capture the regularities of activity Z being labeled with l.

To capture inter-relations between activities, we define the following rule templates for a setting using the directly-follows abstraction, i.e., with *followedBy* as the only relation type in $\mathcal{R}^{followedBy}$:

R.1 *label(Z,l)* $\leftarrow$ *inSameProcess(Y,Z), label(Y,k)*

R.2 *label(Z,l)* $\leftarrow$ *followedBy(Y,Z), label(Y,k)*

R.3 *label(Z,l)* $\leftarrow$ *inSameProcess(X,Y), inSameProcess(Y,Z), label(X,j), label(Y,k)*

R.4 *label(Z,l)* $\leftarrow$ *inSameProcess(X,Y), followedBy(Y,Z), label(X,j), label(Y,k)*

R.5 *label(Z,l)* $\leftarrow$ *followedBy(X,Y), followedBy(Y,Z), label(X,j), label(Y,k)*

R.6 *label(Z,l)* $\leftarrow$ *followedBy(W,X), followedBy(X,Y), followedBy(Y,Z), label(W,h), label(X,j), label(Y,k)*

An example for an instantiation of rule template R.1 is given by:
label(Z,approve purchase order) $\leftarrow$ *inSameProcess(Y,Z), label(Y,parts required)*. This rule captures that when an activity Z occurs in a process model that already contains an activity labeled *parts required*, a possible recommendation for a label for Z is *approve purchase order*.

In general, each of the defined templates captures a certain type of probabilistic regularity about activity inter-relations in process models. More specifically, the above rule templates describe which activities or combinations of activities with certain labels have to be in the same process or must appear before activity Z to predict *l* as the label of Z. We will later introduce confidence as a metric to estimate the probability that a rule makes correct predictions. The probability of a rule that instantiates template R.1, for example, expresses how likely it is that, if an activity (label) k is used in a process, activity l appears in that process as well, whereas the probability of a R.2-rule tells us how probable it is that an activity k is directly followed by an activity with label l.

Rule specificity. Certain rules inherently relate to each other. For instance, any grounding $Y = a_1, Z = a_2 \in N$ of a R.2-rule is a grounding of the corresponding R.1-rule as well. This is the case because *inSameProcess*(Y, Z) is true for $Y = a_1$ and $Z = a_2$ if *followedBy*(Y, Z) is true for $Y = a_1$ and $Z = a_2$, i.e., if a_1 is followed by a_2, then a_1 and a_2 are naturally also part of the same process model. Since the inverse is not true, i.e., *followedBy*$(Y, Z) \leftarrow$ *inSameProcess*(Y, Z) does not have to hold, we say that a R.2-rule is more *specific* than a R.1-rule.

Similar inter-relations also exist for the other rule templates. Figure 5.3 shows a complete specificity lattice of the rule templates. Most of the arcs in the specificity lattice can be explained analogously to the simple rule shown above, or the fact

$$\begin{tikzcd}
 & (R.1) & \\
(R.2) \arrow[ur] & & (R.3) \arrow[ul] \\
 & (R.4) \arrow[ul] \arrow[ur] & \\
 & (R.5) \arrow[u] & \\
 & (R.6) \arrow[u] &
\end{tikzcd}$$

Figure 5.3: Template specificity lattice

that the body of one rule is a subset of another rule's body. Rules that instantiate template R.6 are the most specific rules. Whenever a rule r is more specific than a rule r', rule r tends to make fewer and more specific predictions compared to rule r'.

Templates in other abstraction settings. The rule templates in the $\mathcal{R}^{causal}$ setting can be derived by replacing each occurrence of *followedBy* in the templates by each of the four types of causal relations in $\mathcal{R}^{causal}$. Due to repeated occurrences of *followedBy* in certain templates, this results in a total of 90 templates for the $\mathcal{R}^{causal}$ setting, primarily due to $4 \times 4 = 16$ different versions of template R.5 and $4 \times 4 \times 4 = 64$ of template R.6. For brevity, we refer to the versions derived from one of the templates R.1-R.6 as a *template group* in the remainder. To additionally incorporate the three types of concurrent relations in the $\mathcal{R}^{causal+concurrent}$ setting, we introduce a further template group R.7, which contains three templates that are similar to template R.2, but in which the *followedBy* relation in R.2 is replaced by one of the three *concurrent* relations, e.g., *label(Z,l)* $\leftarrow$ *alwaysConcurrent(Y,Z), label(Y,k)*.

Approach extensibility. Note that our approach is further extendable, since the rule templates can be modified or complemented with additional ones. It is also possible to support even more specific and longer rules templates. In Chapter 7, we will add rule templates that take the natural language semantics of process models into account. However, it should be taken into account that longer rule templates and a higher number of templates greatly expand the search space, which may limit the applicability of the approach on large datasets. Thus, it is always a trade-off between expressiveness and efficiency which guides the final selection. With rule template R.6, for example, we already added a rather specific rule template, which requires that three activities with certain labels appear in a sequence. This condition will usually result in highly accurate predictions, however, at the same time we know that this rule can only be applied if the model under development is very similar to models in the repository.

5.2.2 Rule Generation

To receive a set of rules from the given repository of process models, we instantiate the rule templates by replacing all placeholder variables with the labels $\mathcal{L}_{\mathcal{B}} \subset \mathcal{L}$ used in the repository $\mathcal{B}$. In theory, this means that we, for example, have $|\mathcal{L}_{\mathcal{B}}| * |\mathcal{L}_{\mathcal{B}}| = |\mathcal{L}_{\mathcal{B}}|^2$ possible instantiations of templates R.1 and R.2. However, when learning the rules, it would be infeasible to instantiate the rule templates with all possible combinations of labels and to check then if the rules apply in the given repository. Instead, we generate only such rules for which the conjunction of rule body and rule head hold at least once in the repository.

Instantiations. For each rule template, we start with a relation atom between two activity nodes, for instance, *inSameProcess(X,Y)*, and limit the instantiations of the template to those activities X and Y that are indeed in this relation in the given repository, i.e., activities X and Y occur in the same model at least once. This specifies the values in the associated *label* atoms. If we are looking for instantiations of rule template R.4, for example, and two activity nodes with the labels *loan needed* and *determine needs* are in the *inSameProcess*-relation in the repository, then we add this pair of labels to the set of actual instantiations of j and k in template R.4. Then, we repeat this procedure for the remaining relation atoms of the template on the narrowed set of actual instantiations. With every additional relation atom between two activity nodes in the rule template, the number of actual instantiations decreases. Once there are no relation atoms left, we instantiate the rule template with the determined combinations of labels.

Rule confidence. For each rule that is an instantiation of one of the rule templates, we compute its *confidence* as a measure of its quality. For this, we follow the definition by Galárraga et al. [46], which states that the *support* of a horn rule *head* $\leftarrow$ *body* shall be computed by counting all rule groundings for which both the head and body of the rule are true. Then, to compute a rule's confidence, we divide its support by the number of those groundings that make the body true. Thus, the confidence of a rule can be understood as the probability that the rule makes a

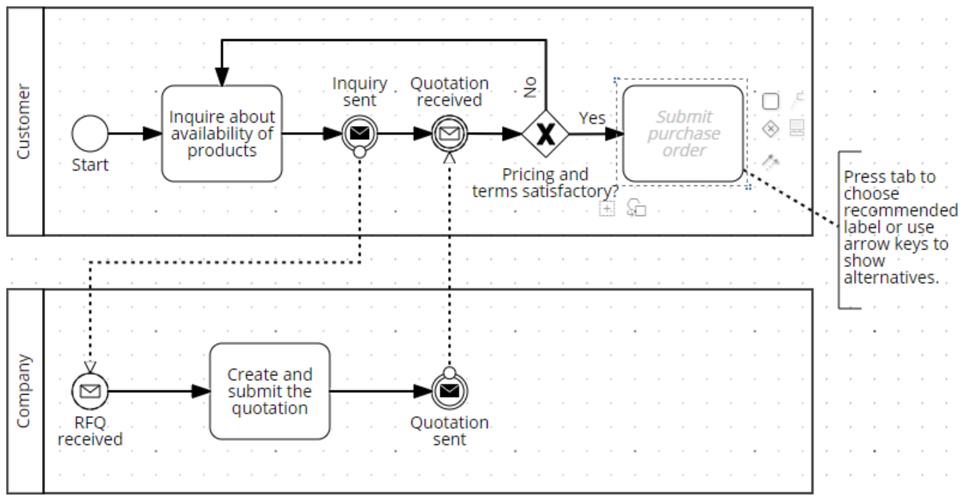

Customer
Start
Inquire about availability of products
Inquiry sent
Quotation received
No
Yes
Pricing and terms satisfactory?
Submit purchase order
Press tab to choose recommended label or use arrow keys to show alternatives.
Company
RFQ received
Create and submit the quotation
Quotation sent

Figure 5.4: A process model under development, where an activity recommendation is given by *Submit purchase order*

correct prediction within the given repository of business process models $\mathcal{B}$. For instance, the following two rules are related to activities that are in the same process with a *parts required* activity:

$$\begin{array}{lll} r_1 = & \textit{label(Z,submit purchase order)} & \leftarrow \textit{inSameProcess(Y,Z)}, \\ & & \textit{label(Y,parts required)} \\ r_2 = & \textit{label(Z,analyze quotation)} & \leftarrow \textit{inSameProcess(Y,Z)}, \\ & & \textit{label(Y,parts required)} \end{array}$$

The body of both rules is the same. Suppose that it holds 15 times over $\mathcal{B}$, i.e, the pattern described by the body appears 15 times in the process models from $\mathcal{B}$. In the example at hand, this means that there are 15 activity nodes in the repository that are labeled *parts required*. Considering that the head is additionally true, assume that these numbers go down to 10 and 5, respectively. For example, in ten out of the 15 cases, a *parts required* activity appears in the same process model as a *submit purchase order* activity. Then, we have $\textit{support}(r_1) = 10$, $\textit{support}(r_2) = 5$, $\textit{confidence}(r_1) = 10/15 = 0.667$, and $\textit{confidence}(r_2) = 5/15 = 0.333$.

5.3 Rule Application

Given an incomplete business process model B with its unlabeled node $\hat{n}$, we use the rules learned from the repository $\mathcal{B}$, as described in Section 5.2, and apply them on $\hat{n}$, while taking the current state of B into account.

Recommendations. To do this, we set $Z = \hat{n}$ for all rules that we have learned and check if the model under development contains activities that can ground the rules, such that the bodies of the rules are true for the model. An example for a rule that instantiates template R.4 in the $\mathcal{R}^{causal}$ setting is given by $(*)$. It is also a rule that could lead to the top-ranked recommendation for the model under development depicted in Figure 5.4, where $\hat{n}$ is the rightmost node.

$$\begin{aligned} &\textit{label}(\hat{n}, \textit{submit purchase order}) \leftarrow \textit{inSameProcess}(X,Y), \textit{inverseAlwaysCausal}(Y,\hat{n}), \\ &\qquad \textit{label}(X, \textit{create and submit the quotation}), \textit{label}(Y, \textit{quotation received}) \end{aligned} \quad (*)$$

If we compare this rule to Figure 5.4, we can see that the body of the rule is indeed true, as we can map X and Y to nodes that have the respective labels.

Once the body of a rule is true, an activity recommendation is given by its head. More specifically, it is directly given by the second argument of the head's *label* atom. Rule $(*)$, for example, recommends *submit purchase order* as the label for $\hat{n}$.

Confidence aggregation. During the rule-application phase, we gather the recommendations of all rules where the body is true with respect to the incomplete process model B and weight the recommendations according to the confidence of their respective rules. If several rules lead to the same recommendation, i.e., predict the same label, we aggregate their confidence scores, such that we can assign the recommendation a single score and rank it accordingly.

For this, we consider two aggregation methods, which we will compare in our experiments in Section 5.4. With the *maximum*-aggregation method [106, 121], we assign the maximum confidence of the applicable rules to the recommendation. If two recommendations have the same maximum probability, we sort them based on their second-highest probability, if available. Analogously, if two recommendations share maximum and second-highest probability, we continue until we find a probability that makes a difference. The *noisy-or* method multiplies the complement to 1 of all confidence scores and assigns the complement to 1 of this product to the recommendation. This method is based on the noisy-or distribution, which represents a simplification of dependency relations in Bayesian networks [137]. After applying an aggregation method, we obtain a ranked list of recommendations for the recommendation task at hand, each with its own confidence score.

Then, we remove all recommendations from the set that refer to a label that is already used in B, since it is generally undesirable to have multiple activities with the same label in a single model [1].

Recommendation transparency. One of the advantages of our approach is that the rules that serve as a basis for the recommendations allow to better understand them. In particular, the rules can be used to explain the recommendations to the user. With respect to the recommendation that results from rule ($*$), such an explanation can be phrased like this: Since the previous activity is *quotation received* and the process also includes a *create and submit the quotation* activity, there is a rather strong indication (high confidence score) that the activity should be labeled *submit purchase order*.

Such an explanation might raise the confidence of the user in the given recommendation and might make it easier for them to make a choice between the presented alternatives. In addition, the recommender system could also provide links to the business process models in the repository that support this recommendation, i.e., where the corresponding rules that lead to the recommendation are true. Hence, the user could take a look at similar processes, which might further help them with the current modeling task.

Finally, should a user find that suggestions that they consider to be wrong have been learned from the available dataset, it is possible to identify the rules that resulted in this recommendation and to remove them from the rule base. In fact, this is considerably easier than it would have been to avoid such recommendations when using, e.g., embeddings-based or neural network approaches, which are more of a black box, for which it is harder to pinpoint and omit specific relations that were learned.

5.4 Evaluation

In this section, we report about our experimental studies, in which we assess the quality of the activity recommendations provided by our rule-based and other approaches. Before we get to the experiments, we introduce the employed dataset (Section 5.4.1) and the evaluation setup (Section 5.4.2). In the first part of the experiments (Section 5.4.3), we compare our approach to other activity-recommendation approaches, using different evaluation procedures. The second part (Section 5.4.4) comprises an ablation study that provides insights into the types of rules that are most important for our recommendation approach.

[1] Note that this assumption is not contradicting the fact that the same activity can be performed several times in a concrete execution of a process. However, this should not be caused by having the same activity twice in the model, but via a cycle in the process flow. There might be some specific cases, where our assumption is not true, e.g., using the same label for activities in different swim lanes in a BPMN model. For such cases, it is possible to implement a language-specific exception.

5.4.1 Dataset

To conduct the experiments, we use models from the BPMAI [184] collection. These models are available in different revisions, which is useful for our purposes, since we consider each revision as a separate process model and thus ensure that most activities appear repeatedly across different processes in the repository.[2]

For our evaluation, we used all BPMN 2.0 models of the collection with 3 to 50 activities described by English labels. The resulting dataset comprises $15,365$ process models and $27,235$ unique activity labels. Note that these process models result from $3,688$ processes and their revisions. On average, the process models involve 15.7 activities while half of the processes comprise 14 activities or less (median). The standard deviation is 9.2.

5.4.2 Evaluation Setup

The evaluation setup involves cross validation as well as different evaluation procedures, metrics, and approaches to be assessed.

Cross validation. For the evaluation, we employ a 10-fold cross validation. Thus, we randomly split the data into ten folds and use nine of those to train a recommendation approach. The remaining fold is then used to establish recommendation tasks in the evaluation. We repeat this procedure, such that each of the folds is used once as the evaluation set. In the remainder, we report the mean results obtained over the 10 folds of the cross validation.

Evaluation procedures. We evaluate our work in various modeling situations, as is common practice for activity-recommendation approaches [72]. Therefore, we use three evaluation procedures, reflecting varying states of process models under development, which results in a variety of recommendation tasks. Specifically, we adopt the *given-k* and *hide-last-two* procedures by Jannach and Fisher [72], and additionally introduce the *full-breadth* procedure. In the following, we first provide a general overview of how the procedures work before diving into their specific details.

The evaluation procedures work in two steps. First, given a business process model B, one of its nodes is selected as the node $\hat{n}$, for which a label must be recommended. Then, as visualized in Figure 5.5, we alter the state of the process model under development by removing some of the other nodes and their associated edges from the model, according to one of the evaluation procedures. The

[2]Note that with the decision to keep these different revisions, we follow the setup used to evaluate the embedding-based activity-recommendation approach RLRecommender [177]. In Chapters 6 and 7, we will also evaluate in settings where only the last revision of a process model is considered.

remaining model and the selected node $\hat{n}$ then define a specific activity-recommendation task. The different procedures result in different degrees of information that is available as a basis for recommendations, i.e., they represent different ways to simulate the current status of a business process model under development. The specifics of the given-*k*, hide-last-two and full-breadth procedures are as follows.

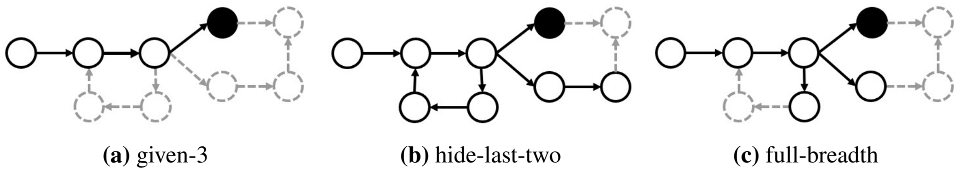

(a) given-3 **(b)** hide-last-two **(c)** full-breadth

Figure 5.5: Illustration of the different evaluation procedures

- **given-k.** In the given-k procedure, we pick a path of length $k+1$ which is a longest path from a source node (node with no incoming edges) to the activity at position $k+1$ and aim to predict the label of this activity. Especially for low values of k, the given-k procedure allows us to compare different recommendation approaches in a *cold-start* setting, in which only little information is given. Important here is that this setting only provides a single sequence of activities as information for a recommendation task. In the evaluation, we use the values 1,3, and 5 for k.
- **hide-last-two.** The opposite to this is the hide-last-two procedure, which maintains a nearly complete process model. Particularly, one sink node n_s (node with no outgoing edges) is randomly chosen and hidden. Then, we randomly select a node that precedes n_s as the node $\hat{n}$ for which a label shall be predicted, while taking all other (non-hidden) activities into account.
- **full-breadth.** Finally, we have implemented a full-breadth evaluation procedure, where one activity, which is neither a source nor sink node, is randomly chosen as the one to be predicted. Then, using s to denote the shortest path from a source node to the selected activity, activities that are on a path of length s starting from a source node are used as a context for the prediction, while all other activities are hidden.

Figure 5.5 illustrates the given-3, hide-last-two and full-breadth procedures. Overall, the given-3 procedure provides the least information as a basis for recommendations, whereas hide-last-two maintains the given process model almost completely. Finally, the amount of information given by the full-breadth procedure usually lies in between them.

Evaluation metrics. To quantify the relevance of the provided recommendations, we employ two established evaluation metrics.

First, we use the *hit rate* Hits@10 to report on the fraction of hits in the top 10 recommendations, i.e., the fraction of cases where the activity label that was actually used in the process model is among the ten most likely recommendations provided by a recommendation approach. This metric is well suited for activity recommendation and other recommendation applications, where it is sufficient that the recommendation list contains one item that the user selects [52].

Second, we report on the Mean Reciprocal Rank (MRR). The reciprocal rank of a recommendation list has the value 0 if the actually chosen activity is not in the provided list and $1/p$ otherwise, where p denotes the position of the hit in the list. Then, the MRR is computed by taking the mean of the reciprocal ranks of all generated recommendation lists. Note that, as for the hit rate, we also consider a recommendation list of length 10 for computing the MRR. This provides a close approximation of the MRR that is based on the full ranking, while at the same time being more realistic than assessing a ranking over all recommendations, as the list of recommendations shown to users will in practice have a limited length as well.

While the hit rate only captures if the recommendation list covers the actually used activity label, the MRR also takes the position of the correct prediction into account. Note that the change of the MRR is much larger when an activity is ranked on position 2 instead of position 1 (0.5) compared to the difference between ranks 9 and 10 (0.01). The MRR thus weights higher positions more heavily.

In the ablation study, we additionally report the average number of generated recommendations (i.e., the length of the recommendation list) per recommendation task as $\varnothing|\text{Rec}|$.

Approach configurations. We evaluate different configurations of our rule-based recommendation approach by varying two aspects. First, we vary the behavioral relations taken into accounts, i.e., we consider configurations that use the *followedBy*, *causal*, or *causal+concurrent* relations described in Section 5.1. Second, we vary the aggregation method over the maximum and noisy-or methods described in Section 5.3. Combining these two aspects results in six different configurations, which we denote, for example, as RULES *followedBy*^{maximum}.

Baselines and other approaches. To contextualize the results of our activity-recommendation approach, we compare them to results obtained from baselines and methods, derived from various existing works:

- COOCCUR [72]: This technique is based on the conditional probabilities of the simultaneous occurrence of activity pairs in a process. Hence, this strategy recommends activities that co-occur most often with the activities that are already present in the process model under development.

- kNN [72]: kNN is a weighted k-nearest-neighbors-based technique. It represents each process model as a vector containing Boolean values that capture whether or not the corresponding activity is present in a model. kNN recommends activities that appear in similar models in the repository, where the vectors of the processes are used to compute the similarity.
- LINK-CTX [73]: Unlike the prior techniques, the link-based LINK-CTX technique takes the order of activities in process models into account. Specifically, it considers the current modeling context and counts in the given repository of processes which activities occurred directly after the last element in the process under development. The score of an activity is hence calculated as the number of times it is a successor to the context's last activity in the repository. LINK-CTX then recommends the activity with the highest score.
- CHAIN-CTX [73]: The chain-based method CHAIN-CTX generalizes LINK-CTX by considering not only activity chains of length two but also longer chains of activities. If longer chains in the modeling context are matched in the repository processes, then CHAIN-CTX gives higher scores to the corresponding recommended activities.
- HYBRID-CTX [73]: The HYBRID-CTX technique combines the kNN strategy, kNN, with LINK-CTX to incorporate two methods that focus on different patterns. HYBRID-CTX is a weighting strategy which gives more weight to the kNN technique for larger processes under development, while LINK-CTX receives a higher weight for smaller ones.
- RLREC [177]: The RLRecommender approach embeds activities and behavioral relations into a continuous low-dimensional space. The embedded vectors and their distances are then used to provide activity recommendations.

While the contextualized methods (LINK-CTX, CHAIN-CTX, and HYBRID-CTX) consider the longest path to the unlabeled activity in the process model under development as the current modeling context for the recommendation, RLREC [177] considers one preceding activity in the process model under development and its inter-relation with the unlabeled activity as a context for the recommendation. The methods COOCCUR and kNN can be understood as simple baselines. The other methods are more sophisticated techniques that have especially been designed to perform well in the given (or a highly similar) activity-recommendation scenario.

Similar to the configurations of our approach, we assess the performance of RLREC for configurations that consider different behavioral relations, i.e., RLREC *followedBy*, RLREC *causal*, and RLREC *causal+concurrent*. The other approaches cannot distinguish between different relations, which means that we apply

	given-1		given-3		given-5		full-breadth		hide-last-two	
Method	H@10	MRR	H@10	MRR	H@10	MRR	H@10	MRR	H@10	MRR
CoOccur	0.290	0.099	0.333	0.105	0.302	0.081	0.215	0.058	0.207	0.049
kNN	0.296	0.121	0.721	0.321	0.749	0.313	0.804	0.434	0.914	0.658
Link-Ctx	**0.495**	**0.389**	0.864	0.672	0.875	0.671	0.777	0.577	0.812	0.615
Chain-Ctx	**0.495**	**0.389**	0.928	0.781	0.929	0.765	0.816	0.644	0.855	0.697
Hybrid-Ctx	**0.495**	**0.389**	0.889	0.721	0.909	0.728	0.857	0.731	0.732	0.646
RLRec *followedBy*	0.470	0.344	0.830	0.590	0.838	0.602	0.738	0.504	0.776	0.524
RLRec *causal*	n/a	n/a	n/a	n/a	n/a	n/a	0.742	0.528	0.786	0.561
RLRec *causal+conc*	n/a	n/a	n/a	n/a	n/a	n/a	0.742	0.528	0.786	0.561
Rules *followedBy*^{maximum}	0.482	0.380	**0.930**	**0.793**	**0.941**	**0.797**	0.886	0.833	0.929	0.878
Rules *followedBy*$^{noisy\text{-}or}$	0.483	0.377	0.930	0.792	0.935	0.791	0.883	0.832	0.928	0.873
Rules *causal*^{maximum}	n/a	n/a	n/a	n/a	n/a	n/a	0.890	0.837	0.929	0.886
Rules *causal*$^{noisy\text{-}or}$	n/a	n/a	n/a	n/a	n/a	n/a	0.890	0.841	0.930	0.880
Rules *causal+conc.*^{maximum}	n/a	n/a	n/a	n/a	n/a	n/a	0.891	0.840	**0.931**	**0.888**
Rules *causal+conc.*$^{noisy\text{-}or}$	n/a	n/a	n/a	n/a	n/a	n/a	**0.892**	**0.845**	**0.931**	0.882

Table 5.1: Experimental results of the different approaches (best results per evaluation procedure and metric are in bold)

them in the $\mathcal{R}^{followedBy}$ setting only. However, the approaches can be appropriately applied to the generic representations of process models that we also use as input for our approach and RLRec, as they only consider which pairs of activities occur in the same process model and which directly follow each other. The representations can losslessly capture this in the *inSameProcess* and *followedBy* relations.

Note that, since the given-k evaluation procedure always captures the current status of the process under development as a sequence of successive activities, it is pointless to consider different causal or concurrent relations for this setting. Therefore, we only consider the $\mathcal{R}^{followedBy}$ setting for the given-k procedure.

5.4.3 Evaluation Results

The results of our experiments are shown in Table 5.1. Except for the specific cold-start evaluation procedure given-1, our rule-based activity recommendation approach outperforms all other methods.

The CoOccur baseline performs comparably poor, while recommendation strategies such as Link-Ctx, which also take structural process patterns into account, achieve better results. This is because they avoid recommending activities that have high co-occurrence statistics, but are not relevant at the current model position. The simple kNN technique performs surprisingly well in cases where more information is given.

RLRec *causal* and RLRec *causal+conc* achieve equal results because RL-

Recommender bases its recommendations on one edge $(m, \hat{n})$ between the activity $\hat{n}$ that has to be labeled and another activity m in the given incomplete process only. In other words, the method does not collect and aggregate predictions from other edges, i.e., relations of activity pairs. Therefore, RLRecommender obtains the longest path to $\hat{n}$ as the context for the recommendation. This specifies the considered edge for the recommendations as edge $(m, \hat{n}) \in E$, where m is in the pre-set of $\hat{n}$, i.e., $m \in \bullet\hat{n}$ and m lies on the longest path to $\hat{n}$. The relation for the prediction thus cannot have a $concurrent$ type, since we only consider sequential activities in the determination of the longest path. This limitation in incorporating contextual information also leads to the comparably low H@10 and MRR numbers.

Our rule-based approach can best exploit its potential when more details are given for the recommendation, i.e., if the given process under development already contains several activities and when applying the more precise relation extraction strategies $\mathcal{R}^{causal}$ and $\mathcal{R}^{causal+concurrent}$. The use of the maximum aggregation in general leads to better overall results, only in the full-breadth procedure is the noisy-or aggregation the better choice. Nevertheless, the differences between our configurations are relatively small in comparison with the differences between our approach and the other ones. This is in particular true when considering the full-breadth and the hide-last-two evaluation procedures. In the latter procedure, for example, our approach is almost 20% better in terms of MRR. This illustrates that our approach is much more capable of leveraging contextual information, while the other methods only benefit from the additional information to a limited degree.

The average time required to provide a recommendation with our rule-based approach is generally below 0.65s when running on an Intel® Xeon® E5-2623 v3@16x3.00 GHz CPU computer with 256G RAM. The leave-one-out procedure using maximum aggregation is an exception to this, which may require 1.1s on average to provide recommendations.

5.4.4 Ablation Study

We investigate the importance of the individual rule templates (and groups) through an ablation study[3]. In particular, we evaluate the performance of our approach when only learning and applying rules from one template group, e.g., a configuration R.1 only considers rules that are instantiations of template R.1. Note that we apply the $\mathcal{R}^{followedBy}$ setting for the given-5 evaluation procedure while we adopt the $\mathcal{R}^{causal+concurrent}$ setting for the full-breadth and hide-last-two procedures. Further, we employ the maximum aggregation for all settings.

The results in Table 5.2 reveal that the templates that include at least one *fol-*

[3]For brevity, we use *template* and *template group* interchangeably in this section.

	given-5			full-breadth			hide-last-two		
Rule templates	H@10	MRR	∅|Rec|	H@10	MRR	∅|Rec|	H@10	MRR	∅|Rec|
R.1	0.739	0.298	7280.5	0.773	0.428	10455.1	0.877	0.609	10888.6
R.2	0.865	0.676	40.0	0.784	0.673	227.5	0.828	0.723	140.1
R.3	0.761	0.310	428.6	0.764	0.432	2616.4	0.881	0.627	2849.4
R.4	0.928	0.791	24.2	0.843	0.802	28.6	0.909	0.874	58.7
R.5	0.901	0.759	5.5	0.687	0.643	3.6	0.776	0.736	4.3
R.6	0.898	0.767	2.1	0.508	0.487	1.2	0.654	0.632	1.3
R.7				0.399	0.355	20.1	0.394	0.349	13.6
RULES	0.941	0.797	7280.5	0.891	0.840	10455.1	0.931	0.888	10888.6

Table 5.2: Results of the ablation study

lowedBy relation, i.e., R.2, R.4, R.5 and R.6, achieve good results in the given-5 case, in which the least information is given for the recommendation. The more information is given, the better the performance of rule templates R.1 and R.3, which consider co-occurrence through *inSameProcess* relations. When using rule template R.4 exclusively, we achieve the best results, which reflects the importance of rule templates that consider structural and co-occurrence patterns simultaneously. Considering *concurrent* relations in isolation, as done in R.7, does not work well, which is not surprising.

The results also reflect that certain rule templates are more specific than others. For example, template R.2 is more specific than R.1, thus, it leads to fewer, but more targeted predictions with a higher confidence, which typically yield a higher recommendation accuracy. However, templates R.5 and R.6 are so specific that exclusive use of them cannot fill a recommendation list of length 10, resulting in comparably lower performance. Given this trend, considering longer rule templates, e.g., that depend on longer activity sequences, is likely unfruitful.

Finally, it is interesting to recognize that the combined configuration, RULES, achieves better results than R.1 to R.7 in every procedure. Furthermore, we also conducted inverted experiments, where we used all but one rule template. While some combinations yielded slightly better results than achieved by combining all rules, this improvement was never consistent across all evaluation procedures. This shows that all rule templates make valuable contributions in certain situations, which is why the full approach yields a recommendation quality that is overall higher than the quality achieved by partial configurations.

5.5 Conclusion

In this chapter, we presented a rule-based approach for activity recommendation in business process modeling. We demonstrated different configurations of our approach, highlighting its extendable nature. Our extensive experiments showed that it outperforms a variety of other methods [72, 73] including an embedding-based approach [177]. In contrast to these methods, our rule-based approach is, furthermore, able to provide explanations alongside the given recommendations.

While our experimental evaluation focused solely on comparisons with existing activity-recommendation methods, our approach was inspired by rule-based methods used, e.g., for knowledge graph completion. Therefore, in Chapter 6, we explore the application of methods originally developed for knowledge graph completion to activity recommendation in business process modeling. We will also compare the performance of various such approaches in an experimental study.

The presented approach has certain limitations, as it only learns rules for entirely equivalent activity labels. Due to this inflexibility, the existing method cannot fully generalize the information embedded within activity labels and requires some degree of similarity between the process model under development and those in the repository. However, these limitations can be addressed by considering the natural language semantics of the process models. We will demonstrate this by extending the rule-based approach in Chapter 7 and introducing a novel transformer-based approach in Chapter 8.

Chapter 6

Activity Recommendation as Knowledge Graph Completion

The activity-recommendation problem can be framed in terms of a representation-learning task, enabling the use of Knowledge Graph Embedding (KGE) models [177]. A knowledge graph captures data on a specific domain in the form of entities and their inter-relations. In our setting, we propose to turn a process model repository and a process model under development into a large knowledge graph, which thus captures knowledge on the relations between activities in the models, e.g., activities that commonly follow each other. Then, the recommendation of an appropriate activity is interpreted as a completion task in the graph, to which so-called knowledge graph completion approaches can be applied. This completion task has attracted lots of attention in the last decade, with the majority of the approaches being based on the idea to embed the knowledge graph into a low dimensional space [180]. More recently, rule-based approaches have shown to be competitive alternatives [106].

In this chapter, we thus investigate how methods that have originally been developed for knowledge graph completion can be applied for activity recommendation in business process modeling and compare the performance of different approaches in an experimental study. In an error analysis, we found some weaknesses of the methods that can partially be fixed through problem-specific post-processing.

The remainder of this chapter is organized as follows. Section 6.1 introduces different approaches for constructing a knowledge graph from given business process models. Section 6.2 discusses the results of our experimental study, in which we compare various approaches to generate activity recommendations. Finally, Section 6.3 concludes the chapter.

6.1 Knowledge Graph Construction

In this section, we discuss different approaches to construct a knowledge graph from given process models such that we can apply KGE models and a rule-based method for knowledge graph completion to the activity-recommendation problem.

A knowledge graph is a collection of triples *(subject, relation, object)*. While the subject and object entities are represented as nodes in the graph, the relation between two entities is given as a labeled edge. To apply methods that have originally been developed for knowledge graph completion to the activity-recommendation problem, we need to describe each business process model $B \in \mathcal{B}$ of the given repository as well as the process model under development in terms of such triples. For the knowledge graph construction, we use the process model abstraction $B = (N, E, \lambda, \rho)$ from Section 5.1. The partial knowledge graph resulting from a model B can, for example, comprise the nodes N and the activity labels $\mathcal{L}$ as entities. We developed three different approaches to translate a business process model into a set of triples, which vary in the used sets of entities and relations.

The simplest approach uses the relations *hasLabel* and *followedBy*. These two relations are the basis for all approaches, as they capture the core information of a business process model as knowledge graph, i.e., a triple $(n, \textit{hasLabel}, \lambda(n))$ expresses that a node n has the label $\lambda(n)$, whereas an edge (m, n) becomes the triple $(m, \textit{followedBy}, n)$. The other two approaches are alternative extensions which, in addition to the structural patterns captured by *followedBy*, consider co-occurrence patterns by using the relations *inSameProcess* and *inProcess*, respectively. This results in the following three translation approaches:

1. For each node $n \in N$, we add a triple $(n, \textit{hasLabel}, \lambda(n))$. Furthermore, we add for each edge $(m, n) \in E$ a triple $(m, \textit{followedBy}, n)$.

2a. This approach extends the first one by additionally using the relation *inSame-Process*: In addition to the first approach, we add for each pair of nodes $m \neq n \in N$ the triples $(m, \textit{inSameProcess}, n)$ and $(n, \textit{inSameProcess}, m)$ to link all nodes that belong to the same process.

2b. This approach is another extension of the first one and an alternative to 2a, where the additional relation is given by *inProcess*: Let $\mathcal{P}$ be a set of process identifiers and π be a function that maps each given business process model B to its unique identifier $\pi(B) \in \mathcal{P}$. In addition to the triples of the first approach, we add for each node $n \in N$ a triple $(n, \textit{inProcess}, \pi(B))$.

The different translation approaches are closely related. In particular, in 2b the two triples *inProcess*(m, p) and *inProcess*(n, p) imply *inSameProcess*(m, n) in 2a and vice versa. However, there is a slight difference. In 2a two activities m

and n are not in the same process, if the triple *inSameProcess*(m, n) does not exist. In 2b they are not in the same process, because we have *inProcess*(m, p) and *inProcess*(n, p') with $p \neq p'$. Detecting that two activities m and n are in the same process with approach 1 is a bit more complicated, as it requires identifying a sequence of *followedBy* triples (the direction does not matter) which establishes a path that connects m and n. This path might be relatively long. When working with translation approach 1 it is thus more complicated to make use of the implicit information that two activities are or are not in the same process model. As we will see in the evaluation in Section 6.2, the different approaches have their merits depending on the choice of the applied knowledge graph completion method.

For the construction of a knowledge graph from the business process models of the given repository and the process model under development, each of the business process models is translated to a partial knowledge graph by one of the above approaches. The partial knowledge graphs of the different process models are connected by the activity labels that they share, while an activity node always belongs to exactly one process model. If, for example, two process models both contain an activity *Register*, then the two respective activity nodes are both linked to the node that represents the label *Register* in the graph.

Given an obtained knowledge graph, we are interested in predictions of the completion task $(\hat{n}, \textit{hasLabel}, ?)$, where $\hat{n}$ denotes the unlabeled node in the process model under development for which we want to suggest a suitable label. For this task, existing completion approaches can be employed, as shown next.

6.2 Evaluation

In this section, we report on the design and results of our study, in which we investigate the performance of different approaches for applying existing knowledge graph completion methods that have not specifically been designed for activity recommendation. Additionally, we compare the approaches to the embedding-based activity-recommendation approach RLRecommender [177] and to our rule-based approach presented in Chapter 5. Before discussing the evaluation results in Section 6.2.3, we provide information about the employed dataset and the evaluation setup in Sections 6.2.1 and 6.2.2, respectively.

6.2.1 Dataset

For the experiments, we employ the BPMAI process model collection [184]. Unlike in the evaluation of our rule-based approach in Section 5.4, we only use the last revisions of the BPMN 2.0 models in the collection. In contrast to using all revisions, we thus represent the possible case that the recommendation methods sometimes only have few or even none domain-specific reference models for the suggestion of activities available. Moreover, the runtimes of the evaluated KGE models remain in a reasonable range of maximum 48 hours for the hyperparameter search given a particular translation approach for knowledge graph completion[1].

Out of all last-revision BPMN 2.0 models, we use those that contain a chain of at least 3 different activities when executed and no more than 50 activities in total. In addition, we only include process models with English labels. These selection criteria result in a dataset consisting of $3,672$ process models. On average, the processes involve 14.3 activities, with a standard deviation of 8.3. The median number of activities per process model is 13.

6.2.2 Evaluation Setup

Our evaluation setup involves the dataset split and statistics, evaluation procedures and metrics, as well as approaches to be evaluated and their hyperparameters.

Dataset split. We randomly separate the dataset into three parts for training, validation and test, respectively. More precisely, we use 80 % of the process models for training each method, while 10 % of the models are used for validation and another 10 % for evaluation.

Evaluation procedure. For the evaluation, we create one recommendation task for every process model in the validation and test split using the full-breadth evaluation procedure described in Section 5.4.2.

Evaluation metrics. To assess the relevance of the generated recommendations, we utilize the hit rate Hits@10 and the MRR, which we previously introduced in Section 5.4.2.

Approach configurations. We combine each of the three translation approaches defined in Section 6.1 with three existing knowledge graph completion methods. For this, we employ the TransE [11] and DistMult [188] KGE models (also referred to as *embedding-based methods*), as well as the rule learner AnyBURL [106][2]. For the application of the KGE models, we use the PyTorch-based framework libKGE [14]. While these models are rather old, they have proven to yield good results when trained appropriately [136].

Other approaches. Additionally, we include our in Chapter 5 proposed rule-based approach (RULES $\textit{followedBy}^{\textit{maximum}}$) and the embedding-based approach RLRec-

[1]Note that we performed the experiments dividedly on two computers: Intel® Xeon® CPU E5-2640 v3@40x2.40 GHz and Intel® Xeon® Silver 4114 CPU@40x2.20 GHz.

[2]We also tested other popular KGE models (ComplEx, ConvE) but they yielded comparatively poor results that we do not report here.

Translation approach	Entities	Relations	Training triples	Validation triples	Test triples
1	$70,876$	2	$105,150$	358	362
2a	$70,876$	3	$955,492$	358	362
2b	$74,548$	3	$153,827$	358	362

Table 6.1: Overview of the dataset statistics for the embedding-based methods

ommender by Wang et al. [177] (RLREC *followedBy*), which have both been developed with a special focus on activity recommendation. Note that we apply the $\mathcal{R}^{followedBy}$ setting, since we also used the *followedBy*-relation for the knowledge graph completion methods (see Section 5.1). Additionally, we use the maximum aggregation method in our rule-based approach (see Section 5.3).

Dataset statistics. Table 6.1 shows the statistics of the dataset for each translation approach when combined with embedding-based methods, i.e., the number of entities and relations as well as the number of triples in the training, validation, and test sets. While the translation approaches 1 and 2a yield $70,876$ entities, which is the sum of the total number of activity nodes ($48,677$) and activity labels ($22,199$), approach 2b also considers processes as entities, which adds the total number of processes in the evaluation ($3,672$) to the sum.

The large difference in the number of triples between the training set and the validation and test sets has two reasons. First, we are only interested in one special prediction task, which is the prediction of objects in triples $(\hat{n}, \mathit{hasLabel}, \lambda(\hat{n}))$, where $\hat{n}$ denotes the node for which we want to recommend an activity label $\lambda(\hat{n})$. Therefore, the validation and test sets comprise for each process model in the validation or test split only one triple $(\hat{n}, \mathit{hasLabel}, \lambda(\hat{n}))$, whereas the training set contains all triples of the process models in the training split that result from the translation approach.

Second, we want to consider the context of the process model under development, i.e., the activities and relations so far included in the process model. Since, for embedding-based methods, the entity that appears in the completion task needs to be part of the learned embeddings, this means that the contexts of the process models in the validation and test splits must be included in the training set. As such, the training set additionally contains all context triples of the validation and test process models. This is not necessary for rule-based approaches such as AnyBURL, since they naturally allow for the consideration of the context, which thus does not have to be included in the training set.

Hyperparameters. For the KGE models, we basically employed the large hyperparameter space presented by Ruffinelli et al. [136]. However, we additionally

Translation approach	Method	Hits@10	MRR
1	TransE	34.0 %	8.4 %
	DistMult	35.9 %	15.0 %
	AnyBURL	36,1 %	14,8 %
2a	TransE	20.7 %	4.2 %
	DistMult	34.3 %	19.1 %
	AnyBURL	37.2 %	15.2 %
2b	TransE	33.4 %	8.0 %
	DistMult	36.5 %	15.6 %
	AnyBURL	35.8 %	14.5 %
RLREC *followedBy*		35.1 %	23.8 %
RULES *followedBy*^{maximum}		47.5 %	41.4 %

Table 6.2: Experimental results of the different approaches

considered the embedding sizes 32 and 64. While our previously proposed rule-based activity recommendation approach has no hyperparameters, we have set the following hyperparameters for AnyBURL: We increased the length of cyclic rules to 5 and the length of acyclic rules to 2 (a higher parameter is not supported by AnyBURL). Also note that rule-based approaches usually do not fine-tune their hyperparameters against the validation set. Thus, both AnyBURL and our rule-based activity-recommendation approach from Chapter 5 do not make any use of the validation set.

6.2.3 Evaluation Results

The results of our experiments are shown in Table 6.2. With Hits@10 numbers that are at least 10 % worse than our rule-based activity recommendation approach and low MRR numbers, no combination of translation approaches and knowledge graph completion methods works well. The two specialized methods, in particular the rule-based method, that are shown in the last two lines of the table work significantly better.

Analysis of predictions. In order to understand the reasons behind the unexpected results, we examined some specific cases. Figure 6.1 shows a process model in the validation set, where the activity *Search for Units Available* has been randomly selected as the one to be predicted. The use of translation approach 1 combined with TransE yields the following top 10 recommendations:

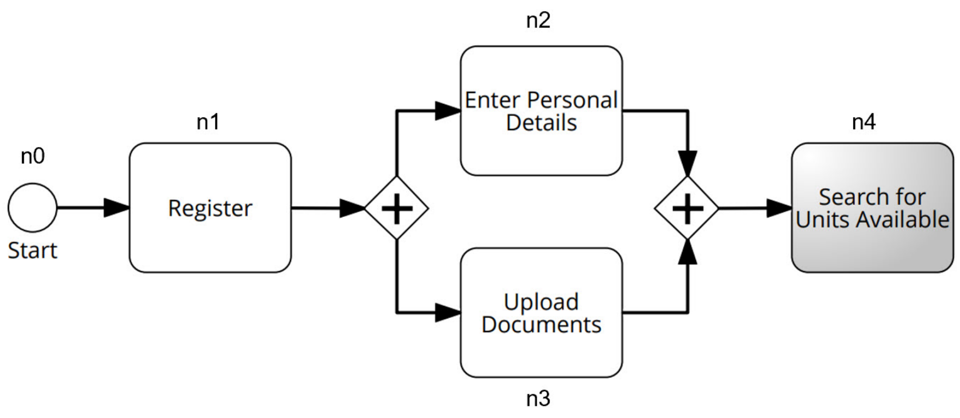


Figure 6.1: An example of a process model in the validation set, where the completion task (n_4, *hasLabel*, ?) has to be solved

1. Register
2. Upload Documents
3. Enter Personal Details
4. Enter Details of Unit for Rent
5. Upload Pictures
6. n4
7. Select Units for Inspection
8. Search For Units Available
9. Start
10. n3

The correct activity *Search for Units Available* is at position eight of the recommendation list, which means that the MRR would be 1/8 = 0.125, if this was the only completion task of the whole evaluation. Other items of the recommendation list include some nodes of the given process model, i.e., n3 and n4, as well as activities like *Register* or *Enter Personal Details* that have already been used in the process model. Clearly, the recommendation of nodes is not useful since we are interested in the prediction of activities. Also, it is likely that activities that have already been inserted into the process model are not added a second time. Therefore, we decided to do a post-processing in which we filter out other recommendations than labels, i.e., nodes and processes, as well as activities that are already present in the given process model. In the example of Figure 6.1, this means that the correct prediction moves to position four of the recommendation list, which is an MRR improvement from 0.125 to 0.25.

Translation approach	Method	Hits@10	MRR
1	TransE	41.7 % (+ 7.7 %)	24.9 % (+ 21.1 %)
	DistMult	38.7 % (+ 2.8 %)	29.8 % (+ 14.8 %)
	AnyBURL	36.9 % (+ 0.1 %)	24.4% (+ 9.6 %)
2a	TransE	37.0 % (+ 16.3 %)	20.8 % (+ 16.6 %)
	DistMult	37.3 % (+ 3.0 %)	29.2 % (+ 10.1 %)
	AnyBURL	38.6 % (+ 1.4 %)	24.2 % (+ 9.0 %)
2b	TransE	42.5 % (+ 9.1 %)	29.3 % (+ 21.2 %)
	DistMult	39.8 % (+ 3.3 %)	31.4 % (+ 15.8 %)
	AnyBURL	36.4 % (+ 0.5 %)	24.4 % (+ 9.8 %)
RLREC *followedBy*		35.1 %	23.8 %
RULES *followedBy*^{maximum}		47.5 %	41.4 %

Table 6.3: Experimental results of the different approaches with post-processing

Results with Post-processing. The results of the experiments with post-processing are shown in Table 6.3. The percentages in brackets indicate the improvement of the associated Hits@10 or MRR numbers in comparison to the results without post-processing.

The KGE model TransE profits most from the post-processing. However, the TransE results strongly depend on the translation approach. This effect is smaller when using DistMult or AnyBURL. For the embedding-based methods TransE and DistMult, translation approach 2b is the best, which shows that the additional explicit information given by the triples with the relation *inProcess* is useful. In contrast, approach 2a works comparably poor in combination with the embedding-based methods. One reason for this could be that in approach 2a the nodes are strongly interconnected via the relation *inSameProcess*. Thus, the interconnection of the nodes via *inSameProcess* is more prominent than via the relation *followedBy*. This can be disadvantageous since the co-occurrence patterns depicted by *inSame-Process* are often less relevant than the structural patterns captured by *followedBy*, which avoid recommending activities that have high co-occurrence statistics, but are not relevant at the current model position.

Unlike the embedding-based methods, AnyBURL achieves the best results when using the translation approach 2a. While this approach only needs one triple $(m, \textit{inSameProcess}, n)$ to express that two nodes m and n are in a process p, approach 2b needs the two triples $(m, \textit{inProcess}, p)$ and $(n, \textit{inProcess}, p)$. This has a direct impact on the regularities that can be captured by AnyBURL. A rule as $\textit{hasLabel}(X, \textit{register}) \leftarrow \textit{inSameProcess}(X, Y), \textit{hasLabel}(Y, \textit{upload documents})$

is within the supported language bias, while the equivalent rule, based on translation approach 2b, has a body length of three and is thus out of scope.

If we now compare the results of the standard knowledge graph completion methods with post-processing to the results of the specialized methods RLRecommender and the rule-based activity-recommendation approach, we observe that the gap between standard and specialized methods has become less significant after post-processing. As discussed in Chapter 5, RLRecommender [177] is based on a rather specific approach to use embeddings in which only one related activity in the process model is used for the recommendation of an activity, thus its results are slightly worse compared to the results of our approaches for using TransE and DistMult. This holds in particular for translation approach 2b where both TransE and DistMult are better in Hits@10 and MRR.

Our rule-based approach, which has specifically been designed for activity recommendation, is still at least 5 % better in Hits@10 and 10 % better in MRR. These significant differences illustrate that a standard knowledge graph completion method cannot compete with an approach which has specifically been designed for activity recommendation in business process modeling. This holds even though we tried out different translation approaches and developed a problem-specific filtering as post-processing step to increase the quality of the results.

It seems that the specific regularities which are important for making a good activity recommendation seem to influence the resulting embedding space only to a limited degree. These specifics are reflected in the types of rules supported by our rule-based recommendation approach that are also more expressive compared to the general rule types supported by AnyBURL. We can conclude that standard methods for knowledge graph completion are not flexible enough to adapt to the given problem, resulting in a relatively low prediction quality.

6.3 Conclusion

In this chapter, we presented different approaches to use embedding- and rule-based knowledge graph completion methods for activity recommendation in business process modeling. By incorporating a problem-specific filtering as a post-processing step, we were able to improve the quality of recommendations. Nevertheless, our rule-based activity recommendation approach still achieved better results than the application of standard knowledge graph completion methods, which revealed their lack of flexibility to adapt to the given problem.

Although the primary focus of this chapter was the application of knowledge graph completion methods to activity recommendation for business process modeling, the findings are also interesting from a broader perspective. Typically, works proposing knowledge graph completion methods evaluate their performance on well-established datasets that have been used in the community for many years. In contrast, our research applied existing methods to a different dataset and an evaluation scenario reflecting a task from practice. Our findings suggest that the performance of knowledge graph completion methods in standard benchmarks may not always translate to success in practical use cases.

Chapter 7

Rule-based Recommendation with Natural Language Semantics

In Chapter 5, we introduced a rule-based approach to recommend activities during business process modeling. Although our experimental evaluation revealed that the rule-based approach outperforms both standard machine learning [72, 73] and embedding-based [177] methods, it is limited by the fact that it only learns rules for completely equivalent activity labels.

To address this limitation, this chapter presents a *semantics-aware recommendation approach* that takes into account the natural language semantics of business process models. Specifically, we extend our existing approach by introducing new rule types that capture *action* and *business-object* patterns in process models. In this manner, the proposed approach is able to, e.g., learn which actions (or business objects) commonly co-occur or follow each other. In addition to improving the rule-learning phase with these new rule types, we also develop a novel extension of the rule-application phase that considers the semantic similarity of actions and business objects. In an experimental study, we show that the semantic extensions indeed improve the existing approach. Additionally, we investigate the impact of the different rule types in an ablation study, which reveals that particularly structural and action patterns are useful for the recommendation of activities.

The remainder of this chapter is organized as follows. Section 7.1 illustrates the advantages of considering natural language semantics for activity recommendation, before Sections 7.2 and 7.3 present the semantic-based extensions in rule learning and rule application, respectively. Section 7.4 discusses the experimental evaluation, followed by the conclusion in Section 7.5.

7.1 Overview of Semantic Extensions

In this section, we motivate the semantic extensions of our rule-based approach, which we propose in this chapter. As a basis for this motivation, we use the example process model under development depicted in Figure 7.1, in which the user has just inserted an unlabeled activity on the right-hand side. The recommendation task is to suggest an appropriate label for this newly inserted activity node.

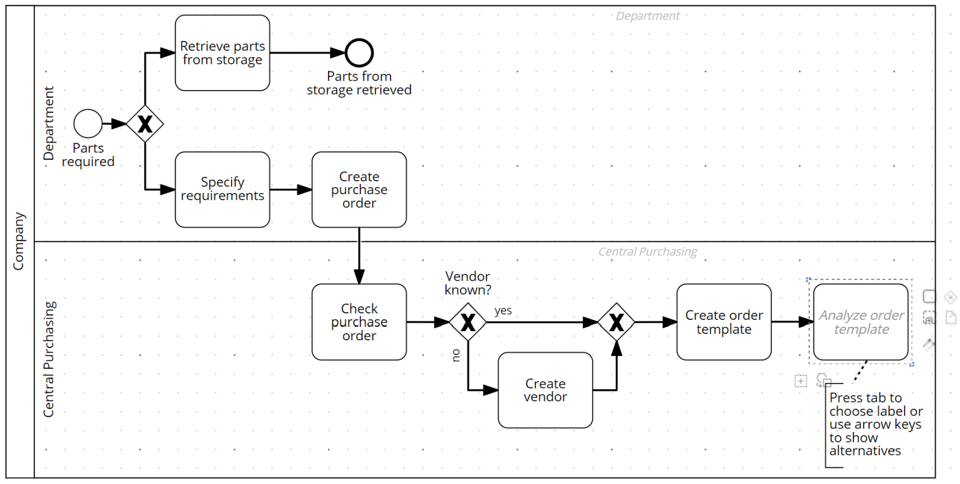


Figure 7.1: A process model under development, where an activity recommendation is given by *Analyze order template*

In our work, we use a repository of business process models as a basis to learn recommendation rules from. So far, our rule-based approach learns rules that capture activity inter-relations in the given repository. In this manner, we find regularities such as "*create order template* is followed by *approve purchase order*" or "*check purchase order* and *create delivery* appear in the same process". Such rules are then employed to suggest suitable labels in an activity-recommendation task. For example, this might result in the recommendations of *approve purchase order* and *create delivery* for the task depicted in Figure 7.1, if such rules can indeed be learned from similar models in the available repository.

However, if the labels included in the process model under development and the ones in the available repository are disjoint, our existing work would not be able to provide any recommendations, since it has not learned any rules that relate to the current recommendation task. Therefore, in this work we recognize that an analysis of the natural language semantics of activity labels can yield more general patterns, allowing us to improve the completeness and quality of activity recommendations.

In particular, our work sets out to provide additional recommendations based on *action patterns*, *business-object patterns*, and *semantic similarity*:

- **Action patterns.** By parsing activity labels, we may learn that for activities that apply to the same business object, a label with a *create* action (e.g., *create request*) is commonly followed by a label involving an *analyze* action (e.g., *analyze request*). This allows us to learn a general pattern that "*create* β is followed by *analyze* β", where β can be replaced by any specific business object. Based on this rule, we would then be able to recommend *analyze order template* as a suitable label for the activity in Figure 7.1, given its preceding *create order template* activity. The action pattern thus enables providing this recommendation, even if the available repository does not specifically contain any of these labels.
- **Business-object patterns.** By also considering inter-relations between business objects, we may learn from a repository that labels related to *order template* and *delivery* business objects commonly appear in the same process, while being complemented by the same action. We might, for example, observe a model that contains the activities *check order template* and *check delivery*, and another model that includes the activities *process order template* and *process delivery*. From this, we may learn the business-object pattern that there is a co-occurrence of α *order template* and α *delivery*, where α can be replaced by any specific action. Given this pattern, we can then recommend a *create delivery* label for the task in Figure 7.1, even if we never observed *create order template* and/or *create delivery* in the available repository.
- **Semantic similarity.** The activity labels used in the given repository of process models can also consist of action and business objects that are not exactly the same as the ones used in the process model under development. However, if we have learned the pattern that "*generate* * is followed by *analyze* *", where * can be replaced by any specific business object, then it is likely that actions that are similar to *generate*, e.g., *create*, are also followed by *analyze*. Analogously, we can expect that the business-object pattern "* *order form* and * *delivery* are in the same process" can also be applied to business objects that are similar to *order form*, e.g., "* *order template* and * *delivery* are in the same process". Both observations are based on the similarity of actions or business objects and lead in the example of Figure 7.1 to the recommendations *analyze order template* and *create delivery*, respectively, even if neither *create* nor *order template* can be observed in the available repository.

In our approach, the identification of action and business-object patterns constitute rule-learning extensions, while we can consider the similarity of actions and business objects during rule application. In the following two sections, we delve into the details of these extensions.

7.2 Rule Learning with Semantic Patterns

Our rule-based approach employs a set of rule templates to generate logical rules. The rules then capture regularities in the use of labels within a given repository of business process models $\mathcal{B}$. To take into account the natural language semantics of process models, we complement the *rigid rule templates*, previously discussed in Section 5.2.1, by novel *semantic rule templates*. While the rigid rule templates describe activity inter-relations in terms of complete activity labels, the semantic rule templates consider natural language semantics in the form of action and business-object patterns. More specifically, they capture regularities in the use of specific actions and business objects throughout a model repository. Corresponding to established work on the semantic analysis of activity labels in process models [109], we use the term business object to refer to an entity to which a label relates, e.g., *requirements*, *order template* or *parts from stock*. Actions of a label operate on business objects, e.g., *specify* (action) *requirements* (business object).

To formalize the semantic patterns that we want to detect in a set of business process models $\mathcal{B}$, we concentrate on *separable* activity labels for the semantic rule templates. We refer to an activity as separable if it allows for a clear separation between the actions and the business objects of its label, for example, as in the activities *specify* (action) *requirements* (business object), *update and review* (actions) *requirements* (business object) or *match* (action) *goods receipt and purchase order* (business objects). In particular, separable activities can involve multiple actions or business objects. For a separable label λ, we denote by $\alpha(\lambda)$ the action part of the label while $\beta(\lambda)$ denotes the business-object part, such that $\lambda = \alpha(\lambda)\ \beta(\lambda)$ or $\lambda = \beta(\lambda)\ \alpha(\lambda)$. The functions α and β can be instantiated using existing approaches for the analysis of activity and event labels, cf., [92, 130]. Since we do not distinguish between other semantic roles than actions and business objects, this means in practice that, given a label, we first identify the action(s) of the label and consider the rest of the label as business object(s).

Rule atoms. For the semantic rule templates, we add the following atoms to the translation of a business process model $B = (N, E, \lambda, \rho) \in \mathcal{B}$ in addition to the ones previously specified in Section 5.2.1, where we denote by $\mathcal{L}_\mathcal{B} \subset \mathcal{L}$ the labels that are used in repository $\mathcal{B}$:

- For each separable label $\lambda \in \mathcal{L}_\mathcal{B}$ with action part *a* we add a formula $\alpha(\lambda) = a$ to capture the actions of the labels that are used in the repository, e.g., α(*create order template*)=*create* or α(*update and review invoice*)=*update and review*.

- Analogously, we add for each separable label $\lambda \in \mathcal{L}_\mathcal{B}$ with business object b a formula $\beta(\lambda) = b$ to also add the business objects of the labels, e.g., $\beta($*create order template*$)$=*order template* or $\beta($*update and review invoice*$)$=*invoice*.

Moreover, we define the replace functions *replaceAction*, *replaceActionAndFlip*, *replaceBusinessObject* and *replaceBusinessObjectAndFlip*, which we apply on separable labels only. The function *replaceAction* : $(\lambda, a) \rightarrow$ *replaceAction*(λ, a) copies label λ and replaces its action part $\alpha(\lambda)$ by the action part a while the business-object part remains the same, e.g., *replaceAction*(*create request, analyze*)=*analyze request*. The function *replaceActionAndFlip* has the same effect as the function *replaceAction* but additionally flips the order of action and business-object part of the generated label, e.g., *replaceActionAndFlip*(*analyze request, approved*)=*request approved*. The functions *replaceBusinessObject* and *replaceBusinessObjectAndFlip* can be explained analogously.

The aforementioned *replace* functions cover all possible combinations of action and business-object parts of separable labels in the repository. Since separable labels consist of two parts, they can occur in both orders, i.e., action part - business-object part and business-object part - action part. Since either might be used in practice, we support both orders. While the *replaceAction* and *replaceBusinessObject* functions combine different parts maintaining their positions in the labels, the *replaceActionAndFlip* and *replaceBusinessObjectAndFlip* functions combine different parts flipping their order in the labels.

Rule templates and instantiations. We define four additional template sets, which adapt the R templates defined in Section 5.2.1 to capture patterns on actions and business objects, as well as their flipped forms. We here provide the A templates for action patterns (referred to as A.1 to A.6) in detail, whereas the other sets are described more briefly. To capture action patterns, we define the following rule templates in the $\mathcal{R}^{followedBy}$-setting, where a and a' indicate action parts used in $\mathcal{L}_\mathcal{B}$:

A.1 *label(Z,replaceAction(K,a′))* $\leftarrow$ *inSameProcess(Y,Z), label(Y,K),* α*(K)=a*

A.2 *label(Z,replaceAction(K,a′))* $\leftarrow$ *followedBy(Y,Z), label(Y,K),* α*(K)=a*

A.3 *label(Z,replaceAction(K,a′))* $\leftarrow$ *inSameProcess(X,Y), inSameProcess(Y,Z), label(X,j), label(Y,K),* α*(K)=a*

A.4 *label(Z,replaceAction(K,a′))* $\leftarrow$ *inSameProcess(X,Y), followedBy(Y,Z), label(X,j), label(Y,K),* α*(K)=a*

A.5 *label(Z,replaceAction(K,a′))* $\leftarrow$ *followedBy(X,Y), followedBy(Y,Z), label(X,j), label(Y,K),* α*(K)=a*

A.6 *label(Z,replaceAction(K,a′))* $\leftarrow$ *followedBy(W,X), followedBy(X,Y), followedBy(Y,Z), label(W,h), label(X,j), label(Y,K),* α*(K)=a*

To yield rules that instantiate the templates, the placeholders h and j have to be replaced by concrete labels from $\mathcal{L}_{\mathcal{B}}$, whereas a and a' have to be replaced by concrete action parts used in $\mathcal{L}_{\mathcal{B}}$.

The A templates provide action-based counterparts for the R templates. For example, whereas R.1 captures patterns on the co-occurrence of entire labels, the corresponding A.1 template captures co-occurrence patterns between actions. An example instantiation of this template is:

$$label(Z,replaceAction(K,approve)) \leftarrow inSameProcess(Y,Z),\ label(Y,K),\ \alpha(K)=create.$$

This rule captures that when an activity Z occurs in a process model that already contains an activity with a label K = *create someObject*, a possible recommendation for a label for Z is *approve someObject*. This recommendation is defined by *replaceAction(K, approve)*, which replaces the action of label K with *approve* while preserving its business object (*someObject*).

Similarly, the following is an instantiation of the behavioral template A.2:

$$label(Z,replaceAction(K,analyze)) \leftarrow followedBy(Y,Z),\ label(Y,K),\ \alpha(K)=create.$$

This rule describes that an activity *Y* with action *create* is followed by an activity *Z* with action *analyze* while the business objects of *Y* and *Z* are the same.

The templates A.3-A.6 combine behavioral or co-occurrence action patterns with regularities that involve whole activity labels, as in the R templates. For example, the probability of a rule that instantiates template A.3 tells us how likely it is that, if an activity *X* labeled *j* is used in the same process with an activity *Y* with label *K*, where *K* includes action part *a*, then the label of activity *Z* in the same process consists of action part *a'* and the business-object part of *K*.

Additional template sets. The function *replaceActionAndFlip* is used in another type of rule templates denoted by AF.1-AF.6, where we replace each occurence of *replaceAction* in the templates A.1-A.6 by *replaceActionAndFlip*. For example, an instantiation of rule template AF.2 is:

$$label(Z,replaceActionAndFlip(K,check)) \leftarrow inSameProcess(Y,Z),\ label(Y,K),\ \alpha(K)=create.$$

This rule captures that when an activity *Z* occurs in a process model that already contains an activity with a label *K=create someObject*, a possible recommendation for a label for *Z* is *someObject check*. This recommendation is defined by *replaceActionAndFlip(K,check)*, which replaces the action of label *K* with *check* while preserving its business object (*someObject*) and additionally flips the order of action part and business-object part in the label *Z* as compared to the order in label *K*.

The A and AF rule templates thus capture the regularities of two actions following each other and two actions co-occurring in one process, where the functions

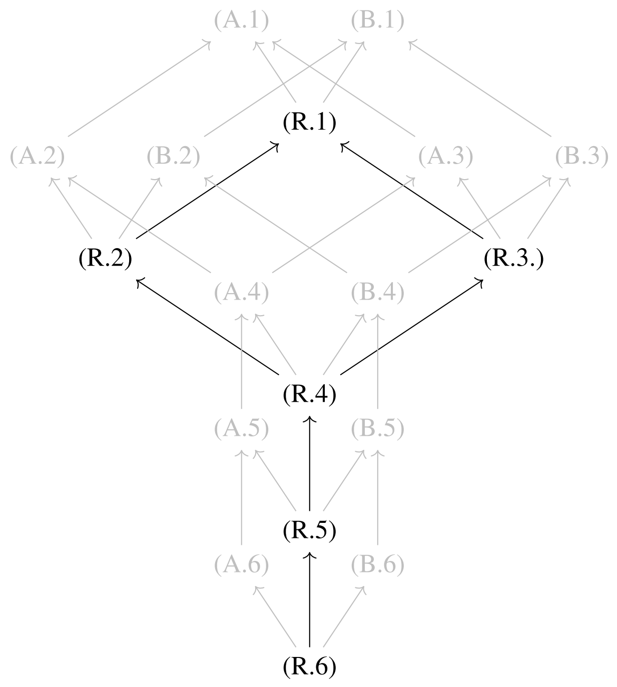


Figure 7.2: Extended template specificity lattice

replaceAction and *replaceActionAndFlip* are instructions on how the label *l* of activity *Z* is composed.

Analogously, we receive another two types of rule templates B.1-B.6 and BF.1-BF.6 for capturing business-object patterns by replacing *action* and *replaceAction* in A.1-A.6 by *businessObject* and *replaceBusinessObject* or *replaceBusinessObjectAndFlip*, respectively.

The rule templates in the $\mathcal{R}^{causal}$- and in the $\mathcal{R}^{causal+concurrent}$-setting can be derived as described in Section 5.2.1 for the rigid rule templates.

Rule specificity. Turning back to the specificity lattice of the rigid rule templates in Figure 5.3, we can extend it by the semantic rule templates as shown in Figure 7.2. The grey arcs show the specificity relations of the A and B rule templates, which are similar to those of the rigid templates. Also, Figure 7.2 illustrates the inter-relations between the rigid rule templates R.1-R.6 and the semantic rule templates A.1-A.6 and B.1-B.6. For instance, whenever the body of an R.1-rule is true, then the corresponding A.1 and B.1 rules are true as well. In general, the A and B rules are weaker forms of their R counterparts, which reflects that the semantic rule templates are more widely applicable than the rigid rule templates, i.e., the rules can be applied when labels just share an action or a business object, rather than being fully identical. Thus, the semantic rule templates make our approach as a whole more broadly applicable. For clarity, we did not include the AF and BF templates in Figure 7.2, since the specificity of these templates is equivalent to their non-flipped A and B counterparts.

Rule generation. Analogously to the rule generation based on the rigid rule templates described in Section 5.2.2, we instantiate the semantic rule templates by replacing all placeholder variables with labels, actions, or business objects from the repository. Due to the functions *replaceAction*, *replaceActionAndFlip*, *replace-BusinessObject* or *replaceBusinessObjectAndFlip*, we additionally limit the instantiations of the rule templates to those activities Y and Z that have separable labels and that relate to the same business object (for action templates in A and AF) or to the same action (for business-object templates in B and BF). In the case of rule template A.6, for instance, the labels of the activities that replace Y and Z need to be separable, and their business objects have to be equal.

For each rule that is an instantiation of one of the semantic rule templates, we compute its confidence as described in Section 5.2.2.

Default rules. Finally, to further enhance the ability to provide recommendations for unseen activity labels and for a sufficient number of recommendations, we also introduce *default rules* in this chapter. Our approach learns ten default rules [45], which recommend the most common labels from a repository. Since these default rules simply predict the activities that occur most often in the repository, the confidences of these rules are low, such that they should only appear in the top ten recommendations list if no other recommendation can be made, i.e., if a prediction task is not covered by any of the actual rules. An example for a default rule is given by *label(Z,send invoice)* $\leftarrow$ *true*. The confidence of this rule is computed by dividing the number of occurrences of the label *send invoice* by the number of all activity nodes in the repository.

7.3 Rule Application with Semantic Similarity

To apply rules derived from semantic rule templates, we can follow the same procedure as outlined for rigid rule templates in Section 5.3. However, if a rule instantiates one of the semantic rule templates, then the activity recommendation isn't directly provided by the rule's head. Instead, an additional step is required: We need to evaluate the function in the second argument of the head's *label* predicate. To illustrate this, consider the following rule, which is an instantiation of rule template A.2 in the $\mathcal{R}^{causal}$-setting:

$$\textit{label}(\hat{n},\textit{replaceAction}(K,\textit{analyze})) \leftarrow \textit{alwaysCausal}(Y,\hat{n}),\ \textit{label}(Y,K),\ \textit{action}(K,\textit{create}) \tag{2$*$}$$

Comparing this rule to the process model in Figure 7.1, we can map Y to the activity node with label *K=create order template*, such that the body of the rule is true. The rule thus provides the recommendation *analyze order template*, since we replace the *create* action of the label *create order template* with *analyze*.

Similarity-based recommendations. We equip our rule-application procedure with an optional extension so that it can also make recommendations if the bodies of rules are not exactly true for the given model, but for which the rule's action or business-object part is semantically similar to the actions or business objects in the process model at hand. Consider, for example, the case that rule (3$*$) is given instead of (2$*$), i.e., instead of a rule related to a *create* action, it now captures a pattern for *generate*:

$$\textit{label}(\hat{n},\textit{replaceAction}(K,\textit{analyze})) \leftarrow \textit{alwaysCausal}(Y,\hat{n}),\ \textit{label}(Y,K),\ \textit{action}(K,\textit{generate}) \tag{3$*$}$$

Because of the semantic similarity of the *generate* and *create* actions, we can nonetheless map Y to the activity node with label *K=create order template* in Figure 7.1. In this way, rule (3$*$) is fulfilled in a semantically similar sense. As before, this would then lead to the recommendation *analyze order template*.

This optional extension makes the rule application procedure more general: Instead of collecting only the recommendations of all rules, where the body is exactly true with respect to the incomplete process model B, we also consider the recommendations stemming from rules, where the bodies are true in a semantically similar sense.

While such recommendations based on semantic similarity can be highly valuable, we still take into account that these stem from rules where the bodies are not exactly true. Therefore, we diminish the confidence scores of these recommendations by a factor that measures the similarity between the actions or business objects used in the model under development and the one involved in the rule with a value between 0 and 1. If, for instance, rule (3$*$) has the confidence 0.92 and the similarity score of the pair (*generate, create*) is given by 0.70, then the recommendation *analyze order template* receives the score $0.92 \cdot 0.70 = 0.644$.

To obtain a score that measures the similarity between two action or business-object parts, we can employ any technique from natural language processing that measures the similarity between two terms. Such techniques typically use vector representations of words, i.e., embeddings, which capture semantic information of words so that similar ones are closer in the vector space, e.g., word2vec [111].

The embedding for a term consisting of multiple words is typically obtained by taking the average of the individual word embeddings, e.g., the embedding of *order template* is the average of the embeddings of *order* and *template*. The similarity of two terms is then calculated as the similarity of the corresponding embeddings, which is often measured using the cosine similarity.

Confidence aggregation. To establish a ranked recommendation list, we assign weights to the recommendations that stem from (semantically similar) rules. Analogously to our approach when using only rigid rule templates, we employ the respective (diminished) confidence scores of the rules for the weighting. Then, we can apply either maximum or noisy-or aggregation to the recommendations of all rules, as detailed in Section 5.3, to obtain the ranked recommendation list for the task at hand.

7.4 Evaluation

In this section, we present an extensive experimental study that we conducted to evaluate our semantics-aware approach for activity-recommendation. We begin by discussing the dataset employed for this study (Section 7.4.1), followed by an outline of the evaluation setup (Section 7.4.2). In the first part of the experiments (Section 7.4.3), we compare our approach to other activity-recommendation approaches in two different scenarios, employing diverse evaluation procedures. The second part (Section 7.4.4) consists of an ablation study, where we investigate the impact of different rule templates and the added value of considering semantic-aware patterns. Finally, we present a study that examines the extension with similarity-based recommendations (Section 7.4.5).

7.4.1 Dataset

Similar to the experimental studies in Chapters 5 and 6, we employed models from the BPMAI process model collection. However, in this study, we use Petri nets in addition to BPMN 2.0 models. From these models, we only included those with 3 to 50 activities described by English labels. Given that the BPMAI collection contains multiple versions (i.e., revisions) of business process models, we obtain a total of 4 128 process models and 18 908 model versions. On average, the process models have 14.5 activities, with a standard deviation of 8.1. The models cover a wide range of domains, resulting in a total of 29 223 distinct activity labels (out of a total of 311 007 activities).

We employ the models in two different application scenarios. In the first scenario, we use the entire set of 18 908 process model versions, i.e., including multiple revisions per process, which reflects the situation that the given repository contains models that are similar to the one for which recommendations shall be provided. In the second scenario, we focus on the opposite case by only selecting the last revision of each of the 4 128 models. This scenario thus results in harder recommendation tasks, given that the repository used to train a recommendation approach will have fewer models (if any) that are similar to the process model for which recommendations shall be made [1].

As input for the semantic rule templates, we identified that 82.5% of the labels are separable, i.e., consist of an individual action and a business-object part (see Section 7.4.2 for details on the parsing procedure). The remaining 17.5% includes labels that are truly non-separable, e.g., *receive error report new bill* or *analyze field and identify processes*, yet also includes ones that simply lack semantics, e.g., *task* or *p t*, turn out to actually not be in English, despite the model being marked as such, or consist of just an action or business object, rather than both, e.g., *accept*, *shipping*, or *simple claim*. Note that we purposefully do not filter out even the nonsensical or non-English labels, in order to avoid biasing the results in favor of our approach.

7.4.2 Evaluation Setup

The evaluation setup involves cross validation, various evaluation procedures and metrics, implementation details, as well as different approaches to be evaluated.

Cross validation. We evaluate the approaches employing a 10-fold cross validation as described in Section 5.4.2.

Evaluation procedures. In addition to evaluating our work in two different application scenarios (with and without model revisions), we also assess the accuracy in various modeling situations. To accomplish this, we employ three evaluation procedures: given-3, full-breadth, and hide-last-two. We described these procedures in detail in Section 5.4.2.

When using the whole dataset (with revisions), we create one recommendation task for every business process model in the evaluation set for every evaluation procedure, i.e., we select a single node per process model as $\hat{n}$ for each procedure. For the last-revision dataset, we compensate for the smaller amount of available process models by instead evaluating all recommendation tasks per model that the given evaluation procedure can provide, e.g., for the full-breadth case, any node in the model that is neither source nor sink node is used as node $\hat{n}$ in a recommendation task.

[1]While the first application scenario reflects the one used in the evaluation in Chapter 5, the second scenario was employed in Chapter 6.

Implementation details. To operationalize the semantic rule templates, our implementation uses the label-parsing approach by Rebmann and Van der Aa [130] to instantiate the α and β functions that, respectively, determine the action and the business-object parts of activity labels. To improve the recognition of actions in ambiguous labels, such as *offer immediate help*, we post-process all labels for which the parser does not detect any action (and only business objects) using the *spaCy* library [64]. Specifically, we use spaCy's part-of-speech tagging feature to determine if any terms in the label are commonly recognized as verbs. If so, this term is then marked as an action, while the other words remain tagged as business objects. This results, e.g., in correctly recognizing *offer* as the action in the aforementioned label. As described in Section 7.2, we only consider *separable labels* when identifying action and business-object patterns, i.e., labels that comprise one action part followed by a business-object part, or vice versa.

We also employ spaCy to compute the similarity between two action or business-object parts. spaCy determines the similarity of two terms using the cosine similarity of the corresponding embeddings. The similarity score lies between 0 and 1, where a higher value indicates a greater similarity. The employed spaCy large English model, en-core-web-lg, contains almost 700 thousand 300-dimensional vectors generated from a large corpus of written text and is thus fully sufficient for our use.

Evaluation metrics. For the evaluation of the provided recommendations, we employ the hit rate Hits@10 and the MRR. Both of these metrics have been previously discussed in Section 5.4.2.

Approach configurations. We evaluate our rule-based approach using the rigid as well as the semantic rule templates in all experiments, while the similarity-based extension is used in the last part of the experiments in Section 7.4.5. Note that we assess the added value of the semantic rule templates in Section 7.4.4 as part of an ablation study, whereas the similarity-based extension is assessed in Section 7.4.5. Further, we evaluate our rule-based approach in different configurations that vary in the applied confidence-aggregation method and the used set of behavioral relations. In particular, we combine the in Section 5.3 introduced maximum and noisy-or aggregation with each of the in Section 5.1 presented abstraction strategies, i.e., *followedBy, causal* and *causal+concurrent*. This leads to six different configurations, which we denote, for instance, by RULES $\textit{followedBy}^{\textit{maximum}}$. Since the given-3 evaluation scenario always yields a sequence of successive activities, there is no point in considering concurrent or causal relations. Therefore, we only report on the $\mathcal{R}^{\textit{followedBy}}$ setting for the given-3 scenario.

Baselines and other approaches. To provide context for the results of our activity recommendation approach, we conducted a comparative analysis with baselines and methods derived from other works [72,73]. The baselines used in our study include CoOccur [72] and kNN [72], as well as the contextualized methods Link-Ctx [73], Chain-Ctx [73], and Hybrid-Ctx [73]. Detailed explanations of these baselines and methods can be found in Section 5.4.2. In addition, we introduce the contextualized versions CoOccur-Ctx and kNN-Ctx of the respective baselines, along with an additional baseline named MostFreq:

- MostFreq [72]: This method always recommends the ten activities most frequently used in the available process model repository.
- CoOccur-Ctx [73]: This contextualized version of CoOccur only considers activities that are part of the current modeling context and recommends activities that co-occurred most often with them.
- kNN-Ctx [73]: kNN-Ctx is a contextualized version of kNN. Compared to kNN, kNN-Ctx increases the weight of the neighbor process models that contain activities, which are also included in the current modeling context of the process model under development.

Note that the contextualized methods consider the longest path to the unlabeled activity in the process model under development as the current modeling context for the recommendation.

7.4.3 Evaluation Results

Table 7.1 shows the experimental results of the evaluated approaches and configurations on the whole dataset, i.e., in an application scenario where multiple similar business process models can be found in the repository. As highlighted in bold, our rule-based recommendation approach achieves the best results in every evaluation procedure.

Baseline comparison. Before we get to the results that we measured for different configurations of our rule-based approach, we take a closer look at the other approaches. As expected, the results of the simple baselines MostFreq, CoOccur and kNN are comparatively low. However, the kNN method achieves up to three times better results than CoOccur, which indicates that the sole consideration of pairwise co-occurrence patterns is less suited for activity recommendation than considering the processes and their similarities as a whole. The methods CoOccur-Ctx and kNN-Ctx work better than their non-contextualized counterparts. This shows that the current modeling context as considered by Jannach et al. [73] can be of importance, i.e., it can be useful to give greater consideration to certain parts of the process model under development.

	given-3		**full-breadth**		**hide-last-two**	
Approach	H@10	MRR	H@10	MRR	H@10	MRR
MOSTFREQ	0.033	0.010	0.015	0.005	0.006	0.003
COOCCUR	0.312	0.104	0.231	0.076	0.212	0.067
kNN	0.684	0.227	0.607	0.200	0.642	0.211
COOCCUR-CTX	0.312	0.104	0.271	0.087	0.263	0.082
kNN-CTX	0.817	0.633	0.768	0.570	0.833	0.651
LINK-CTX	0.852	0.669	0.725	0.556	0.766	0.598
CHAIN-CTX	0.934	0.820	0.774	0.643	0.827	0.713
HYBRID-CTX	0.892	0.730	0.807	0.627	0.861	0.695
RULES *followedBy*^{maximum}	0.938	0.824	0.889	0.798	0.930	0.857
RULES *followedBy*$^{noisy\text{-}or}$	**0.938**	**0.827**	0.875	0.783	0.895	0.771
RULES *causal*^{maximum}	n/a		0.887	0.803	0.931	0.872
RULES *causal*$^{noisy\text{-}or}$	n/a		0.883	0.786	0.910	0.781
RULES *causal+conc.*^{maximum}	n/a		**0.889**	**0.811**	**0.933**	**0.876**
RULES *causal+conc.*$^{noisy\text{-}or}$	n/a		0.884	0.791	0.911	0.784

Table 7.1: Experimental results of the different approaches on the whole dataset (best results per evaluation procedure and metric are in bold)

Methods that (additionally) consider structural patterns rather than co-occurrence patterns only avoid recommending activities that could be useful in the model under development but not at the current modeling point. Therefore, LINK-CTX, CHAIN-CTX, HYBRID-CTX and RULES achieve better results than COOCCUR. CHAIN-CTX achieves better results than LINK-CTX, which demonstrates that taking into account longer chains of activities is useful. By combining LINK-CTX and kNN-CTX, HYBRID-CTX can improve the results of the individual methods in every evaluation procedure. In the full-breadth and hide-last-two procedures, where more context is given, HYBRID-CTX works even better than CHAIN-CTX, which indicates that considering different patterns in the use of activities can be useful. All in all, the comparison of the baselines shows similar trends as the comparison in the work by Jannach et. al [73]: HYBRID-CTX and CHAIN-CTX achieve the best results, the performance of COOCCUR-CTX is rather poor and kNN-CTX can keep up with the methods that take the order of activities into account.

Results per configuration. Overall, the use of the maximum aggregation leads to better results of our approach. Only in the given-3 procedure, the noisy-or aggregation is the better choice, because of a slightly better MRR. The noisy-or aggregation multiplies the confidence scores of all rules that lead to one particular recommendation. Naturally, rules with *inSameProcess* predicates fire more often than those with *followedBy* predicates, thus, this aggregation method gives more weight to co-occurrence patterns than to structural patterns, which is generally unfavorable. This effect exists only to a limited degree in the given-3 procedure, because of the small context in this case.

In general, the more precise the abstraction of business process models, i.e., the more relations used in the abstraction strategy, the better the results of our rule-based approach. However, the hit rate H@10 of RULES *causal*^{maximum} is slightly worse than the one of RULES *followedBy*^{maximum} in the full-breadth case. This can be explained by the fact that we learn the rules on the whole processes of the training set while we apply them on the partial processes of the test set. When generating a partial process from a given process model, some of the nodes and edges are removed, which can lead to a change of the relation types assigned to a remaining edge. If, for example, a node *m* can be followed by node *n* or node *o* in the complete process model and node *o* is removed for the partial model, then node *m* can only be followed by node *n*. This changes the corresponding relation type of edge (*m,n*) from *sometimesCausal* to *alwaysCausal*. Consequently, the rules learned from highly similar process models in the repository cannot be applied to the obtained partial process. Therefore, it could be better to learn the rules on partial processes rather than complete processes. In practice, this would necessitate information about the modeling behavior of the user, which is not available in the for the experiments employed dataset.

All in all, the variations in the results of the different configurations of our rule-based approach are small compared to the differences to the other approaches. Especially in the cases where more context is given, i.e., full-breadth and hide-last-two, our approach outperforms the other approaches by up to 10 % in H@10 and 15 % in MRR. This shows that our approach is much better at using the additional context for providing suitable activity recommendations.

Last-revision dataset. The experimental results of the evaluated approaches on the last-revision dataset, where only few or even no similar models are available for the recommendation of an activity, are shown in Table 7.2. The absolute numbers go down significantly in comparison to the results obtained for the dataset with revisions. However, the general trends to be observed largely remain the same.

Even though the evaluation on the last-revision dataset reflects a challenging scenario, the hit rate Hits@10 is around 50%, which means that, in one out of two cases, the actual activity label is among the top 10 recommendations. Moreover, the comparably high MRR indicates that the actually used label can be found on the first positions of the recommendation list, in case it is indeed in the top-10 recommendations. This means for a concrete modeling task that our approach suggests the correct label in half of the cases within a short recommendation list, whenever a new activity is added to the model. Our approach is thus still beneficial in situations where the modeling domain is only sparsely represented in the given model repository.

	given-3		**full-breadth**		**hide-last-two**	
Approach	H@10	MRR	H@10	MRR	H@10	MRR
MostFreq	0.044	0.017	0.025	0.009	0.013	0.006
CoOccur	0.182	0.069	0.149	0.049	0.139	0.045
kNN	0.304	0.101	0.300	0.090	0.331	0.101
CoOccur-Ctx	0.182	0.069	0.164	0.052	0.165	0.051
kNN-Ctx	0.381	0.276	0.415	0.280	0.433	0.294
Link-Ctx	0.425	0.323	0.418	0.313	0.400	0.297
Chain-Ctx	0.446	0.371	0.443	0.358	0.422	0.341
Hybrid-Ctx	0.432	0.338	0.451	0.328	0.456	0.328
Rules *followedBy*^{maximum}	0.469	0.386	0.500	0.419	0.504	0.413
Rules *followedBy*$^{noisy\text{-}or}$	**0.473**	**0.389**	0.497	0.409	0.488	0.369
Rules *causal*^{maximum}	n/a		0.506	0.428	0.508	0.423
Rules *causal*$^{noisy\text{-}or}$	n/a		0.509	0.419	0.495	0.374
Rules *causal+conc.*^{maximum}	n/a		**0.515**	**0.440**	**0.516**	**0.434**
Rules *causal+conc.*$^{noisy\text{-}or}$	n/a		0.514	0.428	0.500	0.381

Table 7.2: Experimental results of the different approaches on the last-revision dataset (best results per evaluation procedure and metric are in bold)

Ranking. To better understand the ability of our rule-based approach to rank the right activity on the first positions of the recommendation list, we investigate the hit rates Hits@k for k$\leq$10. Figure 7.3 shows these hit rates of our rule-based approach in the $\mathcal{R}^{causal+concurrent}$ setting with maximum aggregation when evaluated in the full-breadth procedure on the whole (top) and on the last-revision (bottom) dataset. Note that the scale of the two graphs differs for the purpose of the figure. When using other settings for our approach or the evaluation, the curves look similarly. Both curves in Figure 7.3 rise steeply at the beginning, while the slope already decreases considerably from Hits@3 onwards. This shows that our approach will in most cases still be able to give the correct recommendation even if only a shorter recommendation list is considered.

Confidence scores. In our evaluation, we have not yet considered the confidence scores, which come along with the recommendations of our rule-based approach. For the hit rate Hits@10 and the MRR, we shortened the generated recommendation list to a top 10 list. However, it is also possible to further shorten the list based on the confidence scores. Then, we only include the recommendations with a confidence score above a certain threshold. Figure 7.4 shows the length of the recommendation list (top) and the recall (bottom) depending on the chosen confidence score threshold when we use our approach in the $\mathcal{R}^{causal+concurrent}$ setting with maximum aggregation and evaluate in the full-breadth procedure on the whole dataset. As before, the graphs look similarly when using other settings for our approach or the evaluation. The recall is similar to the hit rate, i.e., it is the fraction

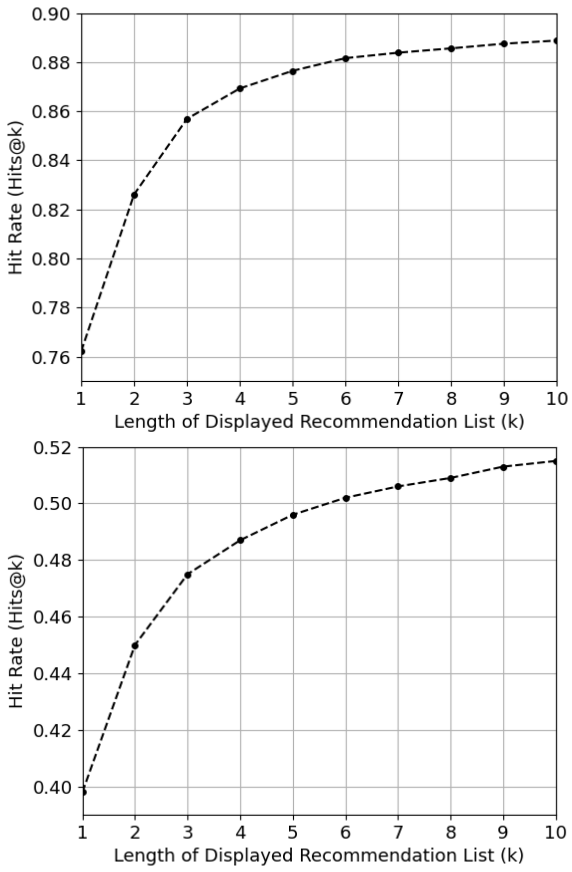


Figure 7.3: Different hits rates of RULES *causal+conc.*^{maximum} in the full-breadth evaluation procedure on the whole (top) and on the last-revision dataset (bottom)

of cases where the activity label that was actually used in the process model can be found in the generated recommendation list. In contrast to the hit rate, which we investigated previously, the length of the recommendation list here depends on the confidence score for the recall.

The top curve in Figure 7.4, which depicts the recommendation list length depending on the confidence score threshold, declines steeply until the confidence threshold 0.2, where the recommendation list length is about 3. Then it slowly further decreases with an exception at the confidence score 0.5, where the length again drops sharply. The bottom graph in Figure 7.4 has the same exception at the confidence score 0.5 and two other exceptions at 0.34 and 0.67. These exceptions result from a large portion of rules having the confidence scores $\frac{1}{2}$, $\frac{1}{3}$ or $\frac{2}{3}$, because there are often two or three options with the same likelihood. Apart from that, the re-

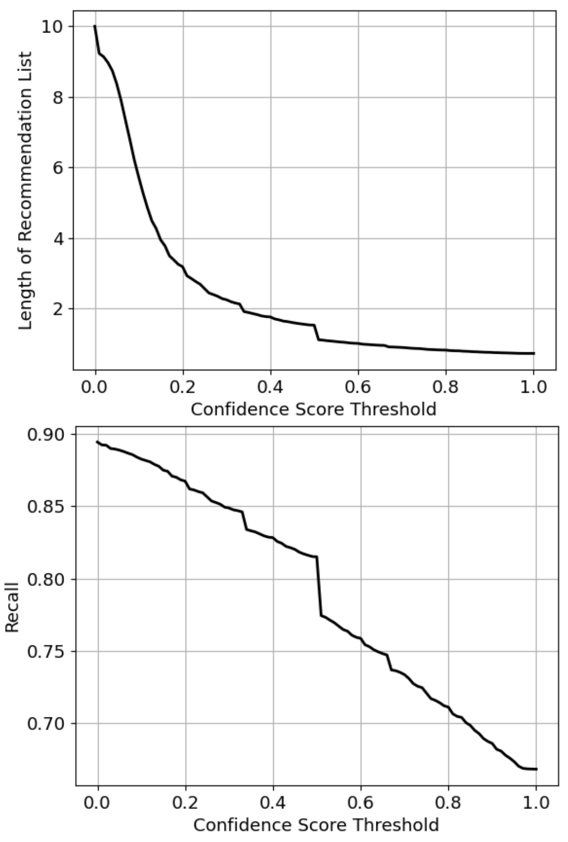


Figure 7.4: Length of recommendation list (top) and recall (bottom) for different confidence score thresholds when using RULES *causal+conc.*^{maximum} in the full-breadth evaluation procedure on the whole dataset

call decreases with increasing confidence score threshold relatively monotonously. Bringing these two graphs together, it could in our example be useful to have a confidence score threshold of 0.2 to make the recommendation lists considerably shorter without decreasing the recommendation quality too much.

Runtime. The average time required to provide a recommendation is generally below 0.74s in our execution environment, which employed an Intel® Xeon® E5-2623 v3@16x3.00 GHz CPU computer with 256G RAM. The hide-last-two scenario using maximum aggregation is an exception to this, which requires 1.59s on average to provide recommendations. The rule learning, which has to be performed only once for a given repository of process models, takes a maximum of 2 086s (34.8 minutes) on the whole dataset and 383s (6.4 minutes) on the last-revision dataset. Since the implementation of our approach is prototypical, it can be assumed that these runtimes can be shortened considerably for an application of the work in practice.

7.4.4 Ablation Study

We investigate the impact of the different rule templates in an ablation study. More specifically, we evaluate the performance of our approach when learning and applying rules from different combinations of rule types and template groups. Note that the numbers in Table 7.3 result from the application of the $\mathcal{R}^{followedBy}$ setting for the given-3 evaluation procedure and the $\mathcal{R}^{causal+concurrent}$ setting for the full-breadth and hide-last-two procedures. Moreover, we employed the maximum aggregation for all procedures, and the last-revision dataset.

	given-3		**full-breadth**		**hide-last-two**	
Employed templates	H@10	MRR	H@10	MRR	H@10	MRR
R.1	0.342	0.143	0.315	0.111	0.326	0.115
R.1, R.2	0.444	0.335	0.467	0.349	0.462	0.345
R.1 - R.3	0.447	0.337	0.464	0.348	0.457	0.343
R.1 - R.4	0.456	0.367	0.488	0.414	0.483	0.404
R.1 - R.5	0.456	0.373	0.489	0.416	0.483	0.408
R.1 - R.6	0.457	0.376	0.489	0.417	0.483	0.409
R.1 - R.7	0.457	0.376	0.499	0.430	0.492	0.421
R and A templates	0.467	0.384	0.513	0.439	0.513	0.433
R and AF templates	0.460	0.379	0.502	0.431	0.497	0.423
R and B templates	0.457	0.376	0.500	0.430	0.493	0.421
R and BF templates	0.457	0.376	0.499	0.430	0.492	0.421
RULES (all templates)	0.469	0.386	0.515	0.439	0.516	0.432

Table 7.3: Results of the ablation study

In the first part of the ablation study, we investigate the rigid rule templates. We start by employing just template R.1 and then add the other rule templates one after another. The first step from using template R.1 to using templates R.1 and R.2 shows that the use of the *followedBy* relation and the associated consideration of structural patterns improves the results significantly. In the next step, we add template R.3, which leads to further improvement in the given-3 case while the results in the other procedures full-breadth and hide-last-two decline. Rules with two *inSameProcess* relations thus rather have a positive effect in the case that less context is given. Adding rule template R.4 with one *followedBy* and one *inSameProcess* relation again improves the results in every evaluation procedure. This emphasizes the importance of considering structural patterns and co-occurrence patterns simultaneously. In the next steps, we add R.5 and R.6, which still improves the results but only to a limited degree. Consequently, considering even longer templates, e.g., that depend on longer activity sequences, is likely unfruitful. On the contrary, adding the rules from template group R.7, which employ the *concurrent* relations, has a strong positive effect on the results again. This is as expected since the results of RULES $causal{+}conc.^{maximum}$ are better than the ones of RULES $causal^{maximum}$ in Table 7.2. Note that in the given-3 case the results of R.1-R.6 and R.1-R.7 are the same as we employ the $\mathcal{R}^{followedBy}$ setting there.

In the second part of the study, we analyze the added value of the semantic rule templates by adding each of the four different types, i.e., A, AF, B and BF templates, to the rigid rule templates. While the BF templates do not influence the results, the other semantic rule templates all contribute to better numbers. Especially the A templates lead to improvements. Given these results, we could use our semantic-aware rule-based approach without the BF templates, since it is likely that they do not make a valuable contribution. Comparing the results of R.1-R.7, where we use the rigid rule templates only, to the results of RULES, we can conclude that the extension of our rule-based approach with the semantic rule templates leads to consistent improvements.

Employing the semantic rule templates in addition to the rigid rule templates is naturally associated with higher runtimes. When using the rigid rule templates only, the average time required to provide a recommendation is generally below 0.70s, where the hide-last-two scenario using maximum aggregation requiring 1.56s is an exception to this. These numbers are to be compared with the runtimes reported in Section 7.4.3 for our method with rigid and semantic rule templates, which are 0.74s and 1.59s, respectively. The rule learning time increases from 1 978s (33.0 minutes) to 2 086s (34.8 minutes) on the whole dataset and from 360s (6 minutes) to 383s (6.4 minutes) on the last-revision dataset.

7.4.5 Experiments with Similarity-based Recommendations

In these last experiments, we want to investigate the performance of our approach when using the similarity-based extension of the rule-application phase, which provides additional recommendations based on the semantic similarity of terms.

Until now, we used the Hits@10 and MRR evaluation metrics, which are strict metrics in the sense that they count a hit only if the given recommendation and the actual activity label are an exact match. If, for instance, the recommendation is *create delivery*, but *create shipment* is used in the business process model, then the recommendation would not count as a hit. The same holds for the pair of activity labels *analyse order template* and *analyze order template*. However, in both cases the given recommendation would arguably still be highly useful for the user, given their similarity from a semantic point of view. Therefore, when evaluating

	given-3		**full-breadth**		**hide-last-two**	
Metric	RULES	RULES + sim	RULES	RULES + sim	RULES	RULES + sim
Hits@10$_{0.7}$	0.827	**0.830**	0.885	**0.894**	0.926	**0.940**
Hits@10$_{0.8}$	**0.644**	**0.644**	0.702	**0.708**	0.727	**0.736**
Hits@10$_{0.9}$	**0.540**	0.533	**0.584**	0.583	**0.591**	**0.591**
Hits@10	**0.469**	0.461	**0.515**	0.509	**0.516**	0.507
MRR$_{0.7}$	0.589	**0.592**	0.652	**0.655**	0.680	**0.684**
MRR$_{0.8}$	**0.494**	0.491	**0.554**	**0.554**	0.565	0**.566**
MRR$_{0.9}$	**0.432**	0.426	**0.485**	0.482	**0.484**	0.481
MRR	**0.386**	0.379	**0.440**	0.434	**0.434**	0.427
Hits@100	0.514	**0.517**	0.564	**0.569**	0.565	**0.572**

Table 7.4: Results with and without similarity-based recommendations (best results per evaluation procedure and metric are in bold)

recommendations made using the semantic-similarity extension, we additionally consider relaxed versions of Hits@10 and MRR that consider the similarity of activitiy labels.

For the relaxed metrics, we compute the similarity of each of the top-10 recommendations to the activity label that was actually used and assign these similarity scores to the recommendations. If *x* denotes a similarity score threshold, then Hits@10$_x$ considers a recommendation to be a hit if its similarity score is equal or higher than *x*. Similarly, MRR$_x$ determines the reciprocal rank of a recommendation list as 1/*p*, where *p* denotes the position of the first recommendation with similarity score equal or higher than *x*. As for the regular MRR, we consider recommendation lists of length 10 for the MRR$_x$, thus, the reciprocal rank is 0 if none of the top-10 recommendations has a similarity score $\geq x$. The MRR$_x$ is then computed by taking the mean of the in this way determined reciprocal ranks of all recommendation lists. Note that we used cosine similarity to determine the similarity scores of recommendations and actually used activity (see the implementation details in Section 7.4.2).

Table 7.4 shows the results of our approach when using the similarity-based extension (RULES+sim) in comparison to the results of the approach with regular rule application (RULES), for varying similarity thresholds. We chose the three similarity thresholds 0.7, 0.8 and 0.9 for the weak version of Hits@10 and MRR. As in the ablation study, we applied the $\mathcal{R}^{followedBy}$ setting for the given-3 evaluation procedure, the $\mathcal{R}^{causal+concurrent}$ setting for the full-breadth and hide-last-two procedures and the maximum aggregation for all procedures. Further, we employed the last-revision dataset.

In the given-3 procedure, where the least context for the recommendation is provided, RULES+sim leads to better results than RULES for the most relaxed threshold, i.e., when $x = 0.7$. For the full-breadth and hide-last-two procedures, RULES+sim performs better for both $x = 0.7$ and $x = 0.8$. From this, we observe that the similarity-based extension can thus provide additional recommendations that are highly similar to the actual label used in a process model. Notably, the more context information is available, the higher the similarity of these recommendations to the actual label.

However, we also observe that the performance in terms of the strict Hits@10 and MRR metrics is better without the similarity-based extension. Because of this rather surprising result, we had a look at the Hits@100 rates, i.e., the fraction of hits in the top-100 recommendations found by the approach. Here, we found that the approach with the extension achieves better Hits@100 scores in every evaluation procedure. On the one hand, this shows that the similarity-based extension can more often recommend the actual activity label used in a process model. On the other hand, though, it also shows that the approach assigns relatively low likelihood to such recommendations, since they often only appear in the top-100 recommendations, as evidenced by the lower Hits@10 and MRR scores. Consequently, it seems that diminishing the confidence scores of the similarity-based recommendations, as described in Section 7.3, does not achieve the desired result and needs to be improved in future work.

7.5 Conclusion

In this chapter, we presented a rule-based approach to support process modelers with activity recommendations, which uses natural language semantics. In particular, our approach considers not only activity inter-relations in terms of complete activity labels but also action and business-object patterns in the use of labels as well as the labels' semantic similarity. Our extensive experiments showed that considering the natural language semantics of the process models is a meaningful addition that improves the quality of the provided recommendations. Still, our rule-based approach is subject to some limitations.

First, in our experiments, the extension with the similarity-based recommendations did not improve our approach in terms of Hits@10 and MRR. However, it led to an improvement in terms of the relaxed metrics, which means that the extension is able to generate recommendations that are similar to the actually chosen activity and can thus be highly useful for the modeler. Also, the Hits@100 results showed that the approach with similarity-based extension recommends the actual activity more often, but not in the first positions. With a better method to integrate the recommendations stemming from the similarity-based extension, the approach could thus be improved even more.

A second limitation relates to the labeling style of the process models. While the rigid rule templates work independently of the labeling style, the semantic rule templates and the semantic-based recommendation work on separable labels only. In the available dataset, this applied to more than 80% of the labels. However, even if none of the labels follow the labeling style needed for the semantic extensions, our approach will be able to make recommendations based on the rigid rule templates. Therefore, the use of the semantic extensions with their labeling style requirement will not reduce the number of process models that can be leveraged for the recommendation approach. Rather, they offer the opportunity to leverage other patterns in the labels to make recommendations.

Third, our approach still requires some similarities of the process model under development and the process models in the repository. For rules that instantiate rigid rule templates, this means that the process model under development and the available models need to share some labels. With the semantic extensions we were able to partially overcome this limitation such that the process model under development and the available models only need to share actions or business objects (for rules that instantiate semantic rule templates) or semantically similar labels (for similarity-based recommendation). Since our method is able to learn action and business-object patterns across different domains, companies with a small number of available process models or repositories with limited representativeness can additionally use other available datasets for learning the rules, e.g., the BPMAI dataset that we used in the experiments.

While we successfully incorporated natural language semantics into our rule-based approach, its limitations indicate potential areas where an activity-recommendation could be more flexible. The limitations exist largely due to the fact that the rule-based approach is entirely dependent on the patterns learned from the process models in the given repository. In the subsequent Chapter 8, we aim to overcome this by leveraging transfer-learning techniques from natural language processing. Therefore, we develop a transformer-based activity-recommendation approach that extends its recommendation capabilities to process models and activities beyond those contained in the available training data.

Chapter 8

Transformer-based Activity Recommendation

Existing activity-recommendation approaches have a limitation in that they can only provide recommendations in the form of labels contained in the given repository of business process models. This leads to poor recommendations for process models that strongly differ from those in the repository. By considering the natural language semantics in process models, our rule-based approach, presented in Chapter 7, is able to alleviate this issue to some extent. However, it can only recommend combinations of actions and business objects already present in the repository.

In this chapter, we aim to overcome this limitation by employing transfer-learning techniques from Natural Language Processing (NLP). We operationalize this by introducing BPART5 – a **B**usiness **P**rocess **A**ctivity **R**ecommendation approach using the pre-trained language model **T5** [129], which is based on the transformer architecture [171]. Our experimental study reveals that BPART5 outperforms the rule-based approach in generating relevant recommendations in terms of semantic accuracy. Furthermore, we show that it is able to leverage the pre-trained language model to generate recommendations that go beyond the vocabulary of the model repository used for training. Specifically, BPART5 recommends numerous activities that are not present in the training data and is also able to provide better recommendations for process models consisting of activities that are not included in the repository's models.

The remainder of this chapter is organized as follows. Section 8.1 presents our BPART5 approach, explaining how we pose the activity-recommendation problem as a set of textual sequence-to-sequence tasks to employ a pre-trained language model. Section 8.2 discusses the details and results of our experimental study, before Section 8.3 concludes the chapter.

8.1 The BPART5 Approach

Given a repository of business process models to learn from, activity-recommendation approaches typically mine relations between activities in the available process models and use the learned patterns to provide recommendations in the form of labels contained in the repository at hand. However, such approaches are restricted to the model repository available for training, which is a strong limitation and results in two key issues. First, this makes these approaches inapplicable in situations where a process model under development entirely consists of activities that were not included in the repository's models, since the extracted patterns cannot be used to make a recommendation in these cases. Second, existing approaches can only recommend activity labels that are already present in the available model repository, which leads to poor recommendations for process models that strongly differ from those in the repository.

To overcome these issues, we propose to capture an instance of the activity-recommendation problem as a set of textual sequence-to-sequence tasks, which enables the application of transfer-learning techniques from NLP. Transfer learning, where a model is first pre-trained on a data-rich task to develop general-purpose abilities and then fine-tuned on a downstream task, has emerged as a powerful technique in NLP [129]. By applying such techniques to the activity-recommendation problem, we can use the general-purpose knowledge of pre-trained language models as an additional source to the problem-specific knowledge contained in a process model repository, thus enabling us to provide relevant recommendations in more settings.

As illustrated in Figure 8.1, our proposed BPART5 approach for activity recommendation consists of two phases: *fine-tuning* and *generation*. It uses the pre-trained language model T5 at its core[1]. Since T5 requires totally ordered, textual sequences as input, whereas process model nodes can be partially ordered, Section 8.1.1 describes how we lift activity recommendation to the format of sequence-to-sequence tasks. In Section 8.1.2, we describe how we use this procedure in the first phase to fine-tune T5 for activity recommendation based on extracted sequences of the process model repository. Finally, Section 8.1.3 describes how we use our fine-tuned T5 model in the second phase to iteratively solve instances of the activity-recommendation problem by first solving multiple sequence-to-sequence tasks and then aggregating their results.

[1]Note that our approach employs T5 as pre-trained language model, but any other model could be used just as well.

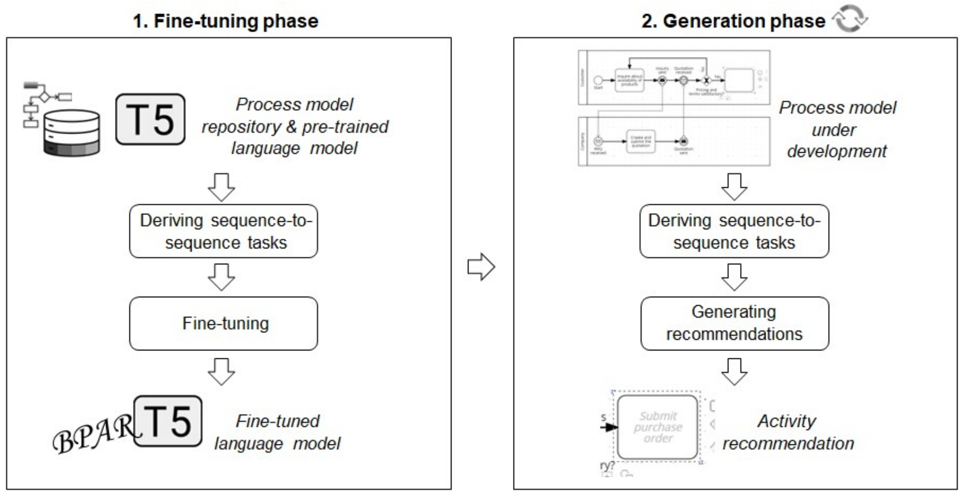

1. Fine-tuning phase
T5
Process model repository & pre-trained language model
Deriving sequence-to-sequence tasks
Fine-tuning
BPART5
Fine-tuned language model
2. Generation phase
Process model under development
Deriving sequence-to-sequence tasks
Generating recommendations
Submit purchase order
Activity recommendation

Figure 8.1: Illustration of the two phases of our BPART5 approach: 1. Fine-tuning, and 2. Generation

8.1.1 Deriving Sequence-to-Sequence Tasks

Sequence-to-sequence tasks are concerned with finding a model that maps a sequence of inputs $(x_1, \dots, x_T)$ to a sequence of outputs $(y_1, \dots, y_{T'})$, where the output length T' is unknown a priori and may differ from the input length T [160]. A classic example of a sequence-to-sequence problem from the NLP field is machine translation, where the *input sequence* is given by text in a source language and the *output sequence* is the translated text in a target language.

In the context of activity recommendation, the *output sequence* corresponds to the activity label $\lambda(\hat{n})$ to be recommended for node $\hat{n}$, which consists of one or more words, e.g., *notify about outcome*. Defining the *input sequence* is more complex, though, since the input to an activity-recommendation task consists of an incomplete process model M_I, whose nodes may be partially ordered, rather than form a single sequence.

To overcome this, we turn a single activity-recommendation task into one or more sequence-to-sequence tasks. For this, we first extract multiple node sequences from M_I that each end in $\hat{n}$. Formally, we employ Definition 1 and write that $S_l^{\hat{n}} = (n_1, \dots, n_l)$ is a node sequence of length l, ending in node $\hat{n}$ ($n_l = \hat{n}$), for which it must hold that $n_i \in \bullet n_{i+1}$ for all $i = 1, \dots, l-1$. Finally, since an input sequence should consist of text, rather than of nodes, we then apply *verbalization* to the node sequence, which strings together the types and (cleaned) labels of the nodes in $S_l^{\hat{n}}$, i.e., $\tau(n_1)\ \lambda(n_1)\ \dots\ \tau(n_{l-1})\ \lambda(n_{l-1})\ \tau(\hat{n})$.

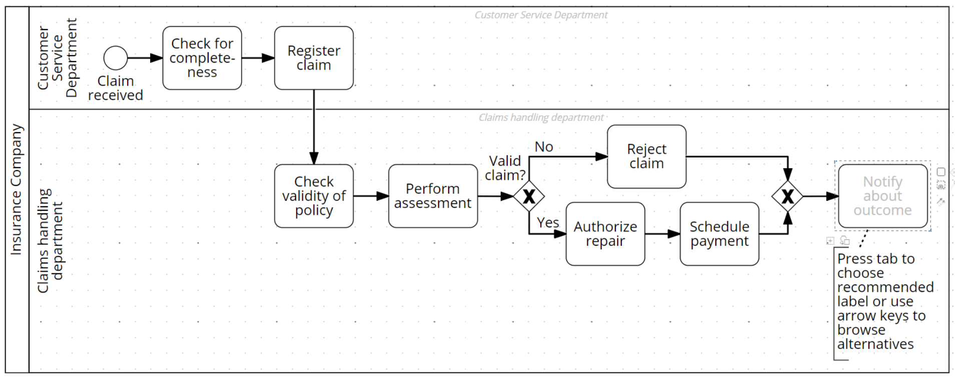


Figure 8.2: A process model under development, where an activity recommendation is given by *Notify about outcome*

As an example, consider the process model depicted in Figure 8.2, which starts when a claim has been received, followed by various activities that handle the claim. This involves a decision point, indicated by an *XOR-split gateway* (diamond shape with an *X*), where a claim is either rejected, or its payment is authorized and scheduled. Following this decision, the model synchronizes the two branches using an *XOR-join gateway*. After this gateway, a new activity has been inserted, for which the activity-recommendation task is to suggest one or more suitable labels. Using sequences of length four, for example, we obtain two verbalized input sequences for the recommendation problem in the figure:

- *task* authorize repair *task* schedule payment *xor task*
- *xor* valid claim *task* reject claim *xor task*

We use this notion of sequence extraction and verbalization to fine-tune T5 for activity recommendation, as described next.

8.1.2 Fine-Tuning

For our approach, we use the sequence-to-sequence model T5 [129], which is based on the transformer architecture [171]. T5 is pre-trained on a set of unsupervised and supervised tasks, where each task is converted into a text-to-text format. We fine-tune T5 for activity recommendation by extracting a large number of sequence-to-sequence tasks from the models in an available repository of business process models $\mathcal{B}$. Specifically, for each model $B \in \mathcal{B}$, we extract all possible sequences of a certain length l that end in an activity node, i.e., $(n_1, \ldots, n_l)$ with $n_l \in A$. Afterward, we apply verbalization on this node sequence to get the textual input sequence, as described in Section 8.1.1, whereas the output sequence corresponds to the label of n_l.

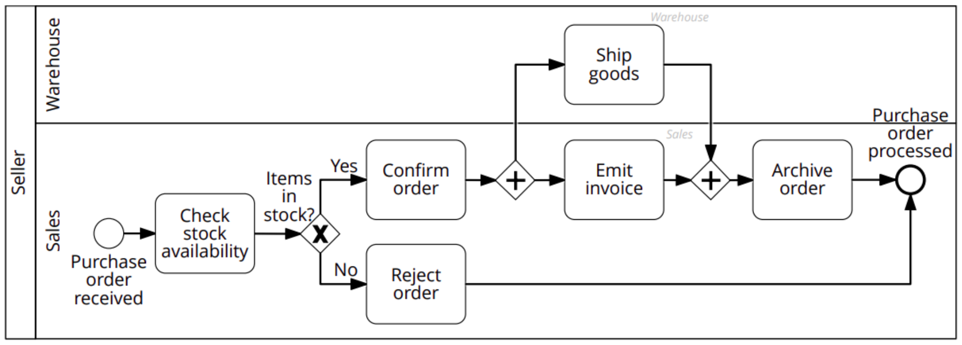


Figure 8.3: An order-to-cash process model

To illustrate this, consider the example training process model depicted in Figure 8.3. Setting $l = 4$, the model contains nine sequences of length four that end in an activity node. After verbalization, these result in the following textual *(input,output)* sequences, which we use to fine-tune T5:

- (*start* purchase order received *task* check stock availability *xor* items in stock *task*, confirm order)
- (*start* purchase order received *task* check stock availability *xor* items in stock *task*, reject order)
- (*task* check stock availability *xor* items in stock *task* reject order *end*, purchase order processed)
- (*xor* items in stock *task* confirm order *and task*, ship goods)
- (*xor* items in stock *task* confirm order *and task*, emit invoice)
- (*and task* ship goods *and task*, archive order)
- (*and task* emit invoice *and task*, archive order)
- (*task* ship goods *and task* archive order *end*, purchase order processed)
- (*task* emit invoice *and task* archive order *end*, purchase order processed)

8.1.3 Generating Recommendations

Given an incomplete process model M_I with an unlabeled activity node $\hat{n}$, for which we want to provide label recommendations, we first extract all sequences of length l that end in $\hat{n}$. We then verbalize all these sequences and feed the resulting input sequences as sequence-to-sequence tasks into our fine-tuned T5 model. For instance, for the example of Figure 8.2, this results in the two input sequences described earlier when using $l = 4$, which are:

- I_1: *task* authorize repair *task* schedule payment *xor task*
- I_2: *xor* valid claim *task* reject claim *xor task*

Output sequence generation. We solve the individual sequence-to-sequence tasks by feeding each input sequence into our fine-tuned T5 model, generating 10 alternative output sequences, i.e., 10 possible label recommendations, per input. To do this, we use *beam search* [50] as a decoding method, with beam width $w = 10$. The beam search algorithm uses conditional probabilities to track the w most likely output sequences at each generation step.

A downside of the beam search algorithm is that it can lead to output sequences that repeat words or even short word sequences, i.e., *n-grams*. Following activity labeling convention [109, 110, 126], we favor the suggestion of short labels that do not contain any recurring terms. For example, rather than suggesting labels such as *check passport and check visa*, our approach would suggest the non-repetitive alternative: *check passport and visa*. To achieve this, we apply n-gram penalties [79, 123] during beam search. Specifically, we penalize the repetition of n-grams of any size (including single words) by setting the probability of next words that are already included in the output sequence to zero.

In general, another disadvantage of the beam search algorithm is that it follows a distribution of highly probable words, while this is not the case for human-generated text of high quality [63]. In contrast, humans try to generate unpredictable and surprising text. However, this does not constitute a disadvantage in the context of activity recommendation, as activity labeling is supposed to contribute to consistency and clarity in process models [126]. Thus, activity labeling is a rational task which requires high precision rather than creativity, making beam search a good choice for generating activity recommendations.

Tables 8.1a and 8.1b show the alternative output sequences (and probabilities) that the fine-tuned T5 model generates for input sequences I_1 and I_2 when using beam search in the example of Figure 8.2.

Output sequences for I_1	Score
notify about outcome	0.64
send notification	0.48
inform about outcome	0.47
send claim rejection	0.38
submit claim to system	0.37
notify claim rejection	0.37
notify customer	0.36
send email notification	0.36
submit claim to management	0.34
notify claimant	0.33

(a) Output sequences and probabilities based on I_1

Output sequences for I_2	Score
send email to customer	0.46
email notification	0.46
notify about outcome	0.42
send email to client	0.40
customer notified	0.36
email	0.34
notification sent to customer	0.31
process end	0.29
send email notification	0.28
process claim	0.26

(b) Output sequences and probabilities based on I_2

Label recommendations	Score
notify about outcome	0.64 (0.42)
send notification	0.48
inform about outcome	0.47
send email to customer	0.46
email notification	0.46
send email to client	0.40
send claim rejection	0.38
submit claim to system	0.37
notify claim rejection	0.37
send email notification	0.36 (0.28)

(c) Final list of label recommendations

Table 8.1: Example maximum aggregation

Result aggregation. Finally, we aggregate the different lists of output sequences, obtained by using beam search to solve individual sequence-to-sequence tasks, in order to end up with a single list of recommended activity labels. To do this, we aggregate the contents of the lists using the maximum method, which we already used in our rule-based approach to rank proposed activities according to the different confidence values of the rules that suggested them (see Section 5.3).

To apply the maximum aggregation method, we establish an aggregated recommendation list, sorted according to the maximal probability score that a recommended label received in the output generation based on beam search. For instance, the *notify about outcome* label receives a score of 0.64, from the output sequences generated for I_1, although the label also appears in I_2's list, yet with a score of 0.42. Thus, this label has the score 0.64 in the recommendation list. In the end, BPART5 provides a list of ten activity recommendations for the unlabeled node $\hat{n}$ that are the most probable candidates, according to the sequences contained in the process model under development, the fine-tuned T5 model, and the maximum aggregation method.

The final list obtained for the process model under development depicted in Figure 8.2 is shown in Table 8.1c. Notably, the top five recommendations represent alternative manners to inform an applicant, e.g., in the form of *notify about outcome*, *send notification*, or *send email to customer*. This indeed appears to be the appropriate process step given that the preceding nodes indicate that the outcome of a claim has been determined, after which it is natural to inform the claimant.

8.2 Evaluation

In our experimental evaluation, we assess the performance of BPART5 and compare it to our rule-based approach, which we presented in Chapter 7. We first introduce the employed dataset in Section 8.2.1, then we describe the experimental setup in Section 8.2.2. Finally, we present the results of our experiments in Section 8.2.3.

8.2.1 Dataset

To conduct the experiments, we employ the SAP-SAM dataset presented in Chapter 4 and filter it as follows. We select all BPMN 2.0 models in English with 3 to 30 nodes (including gateways), where each activity label is composed of at least three non-empty characters. Moreover, we exclude default vendor-provided example models included in SAP-SAM. Note that for filtering and pre-processing the models of SAP-SAM we apply label cleaning, in which we turn non-alphanumeric characters into whitespace, handle special cases as line breaks, change all letters into lowercase and delete unnecessary whitespace. This results in a filtered dataset, which consists of 77,239 process models containing an average of 14.7 nodes (median: 13) and a total of 241,283 unique node labels with an average length of 26.5 characters (median: 24).

8.2.2 Evaluation Setup

The evaluation setup involves the dataset split, different evaluation procedures and metrics, implementation details, as well as configurations of BPART5 and the rule-based approach.

Dataset split. We randomly divided the models in the filtered dataset into three parts for training, validation, and test, respectively. More precisely, we train each approach on 85 % of the process models while we use 7.5 % of the models for validation and evaluation, respectively. From the training split, we extracted a total of 688.584 sequences, which we verbalized and used to fine-tune T5 for BPART5.

Evaluation procedure. For the evaluation, we employ the full-breadth procedure as described in Section 5.4.2. For each process model in the test split, we generate several evaluation cases by carrying out the full-breadth procedure for each activity node, where the shortest sequence to a source node has the minimum length three. This leads to a total of 36.143 evaluation cases, i.e., activity-recommendation tasks.

Implementation details. Our implementation of BPART5 and the metrics uses the Huggingface library [187]. For tokenizing sequences, we used the *fast* T5 tokenizer backed by HuggingFace's tokenizer library, which is based on Unigram [83] in conjunction with SentencePiece [84]. We fine-tuned T5-Small[2] employing the Adam algorithm [78] with weight decay fix as introduced in [100] and constant learning rate 0.0003. Moreover, we set the batch size to 128 and trained the model until the validation loss did not improve for 20,000 steps. The experiments were carried out using two Nvidia RTX A6000 GPUs.

Evaluation metrics. In this chapter, we assess the performance of the recommendation approaches using four different metrics, namely Hits@k, BLEU@k, METEOR@k and Cos@k:

- **Hits@k.** First, we report on the standard *hit rate* Hits@k, which we used in the previous evaluations in Chapters 5, 6, and 7, too.
- **BLEU@k.** The BLEU [122] metric is typically used in machine translation, where a candidate translation is compared to one or more reference translations.

[2]Compared to T5-Base with its 220 million parameters, T5-Small is a model checkpoint that has only 60 million parameters.

In the context of activity recommendation, BLEU basically compares n-grams of the recommended activity with n-grams of the ground-truth activity and calculates a modified precision based on n-gram matches. Similarly to the standard hit rate Hits@k, we can define the BLEU@k hit rate as the maximum BLEU score of the top-k recommendations. This results in a single score for a recommendation list of length k instead of the k BLEU scores of each recommendation in the list.

- **METEOR@k.** Just as BLEU, the METEOR [6] metric is also typically used to assess the quality of machine translations[3]. In our context, METEOR evaluates the quality of an activity recommendation based on unigram matches with the ground-truth activity. In addition to exact matches, it also considers semantic similarity in the form of stemmed matches and WordNet-based [40] synonym matches. Analogously to BLEU@k, the meteor hit rate METEOR@k is given by the maximum METEOR score of the first k recommendations.
- **Cos@k.** The cosine similarity [93] requires representations of the activity recommendation and the ground-truth activity as embeddings, enabling the calculation of the similarity of the two activities in the form of the cosine similarity of their embeddings. Cos@k is then the maximum cosine similarity score of the top-k recommendations. In our evaluation, we use the Universal Sentence Encoder [20] to generate the embeddings of activities, which allows Cos@k to consider the semantic similarity of the recommendations and the actual used activity.

In a user study, Goldstein et al. [49] showed that BLEU, METEOR and cosine similarity strongly correlate with experts' ratings of activity recommendations. Thus, they can be confidently used to measure the quality of activity recommendations. However, their work lacks details about how the metrics can be used to evaluate a recommendations list rather than a single recommendation. We address this gap with the above definitions of BLEU@k, METEOR@k, and Cos@k as the semantic counterparts of Hits@k.

Employing four metrics allows us to gain different kinds of insights. The standard hit rate, Hits@k, is a strict metric in the sense that a hit is realized only if a recommendation and the ground truth are an exact match. If, for example, a recommendation is given by *Notify about outcome* while *Inform about outcome* is

[3]Note that BLEU and METEOR are designed for the comparison of (long) sentences or text corpora. Penalties in the definitions of the metrics can thus cause the metrics to be (close to) zero for short activity recommendations, even if ground truth and recommendation match. Therefore, we manually set the BLEU and METEOR scores to 1 if a recommended activity and the ground-truth activity are an exact match.

Recommended activity label	BLEU	METEOR	Cosine Similarity
Notify about outcome	1.0	1.0	1.0
Send notification	0.0	0.0	0.46
Inform about outcome	0.58	0.63	0.80
Send email to customer	0.0	0.0	0.27

Table 8.2: Values of BLEU, METEOR and cosine similarity for different recommendations given the ground truth *Notify about outcome*

used in the test process model, then the recommendation would not count as a hit and the recommendation approach would be considered unsuccessful in this case. However, given its similarity and the fact that there are several possible manners of describing an activity with a label, the recommendation would still be highly useful for the modeler. In this sense, the semantic hit rates BLEU@k, METEOR@k and Cos@k are more practice-oriented. By taking the similarity of recommendations to the ground truth into account, they measure the semantic accuracy of the recommendations. The values of the semantic hit rates are always bigger or equal to the Hits@k values. To illustrate the different levels of similarity that are measured by the three semantic hit rates, Table 8.2 shows the values of BLEU, METEOR and cosine similarity for four example recommendations from the list in Table 8.1c, given that the actual used activity label is *Notify about outcome*.

Approach configurations. In our experiments, we choose a sequence length l=4 for our BPART5 approach, i.e., we extract sequences of length four that end in node $\hat{n}$. This choice follows findings from prior research [48], which showed that considering three previous nodes for activity recommendation works well across different datasets.

Other approaches. To provide a comparative analysis, we compare the performance of BPART5 to the performance of our rule-based approach (RULES). We presented the rule-based approach in Chapter 7, where we also showed that it outperforms several other activity-recommendation methods. For consistency, we used the $\mathcal{R}^{followedBy}$ setting for our rule-based approach in this chapter's evaluation, as we did not differentiate between other behavioral relations of activities beyond the *followedBy*-relation in BPART5 (see Section 5.1). Moreover, we employ the maximum aggregation method in our rule-based approach (see Section 5.3).

We also considered including the approach proposed by Goldstein et al. [48] in our analysis. Their approach employs a pre-trained language model without fine-tuning to retrieve activities from the given repository of business process models as recommendations. However, their approach involves calculating cosine similarities of the sequences in the process model under development and all sequences in the training dataset. In our experiments, we extracted 688.584 sequences from the process models in the training dataset, resulting in the calculation of 688.584 cosine similarities for each considered sequence in the process model under development. As a result, it took us around ten minutes per evaluation case to generate recommendations using their approach. Given that we evaluate 36,143 recommendation cases, it was infeasible to include this approach in our study.

8.2.3 Evaluation Results

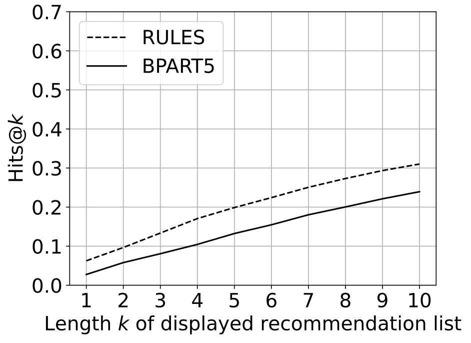
In this section, we first consider the overall results, after which we assess how well BPART5 deals with the key limitations it aims to address: the ability to generate and handle activity labels not contained in the training data.

List size	Approach	Hits@k	BLEU@k	METEOR@k	Cos@k
$k = 10$	RULES	**0.3102**	0.3358	0.4149	0.5925
	BPART5	0.2800	**0.3876**	**0.5154**	**0.6679**
$k = 1$	RULES	**0.0625**	0.0714	0.1049	0.2539
	BPART5	0.0322	**0.1179**	**0.2269**	**0.4112**

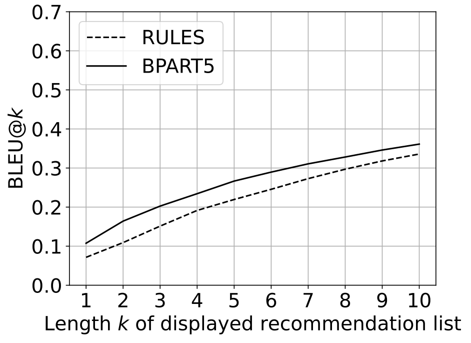
Table 8.3: Experimental results of the different approaches (best results per metric are in bold)

Overall results. The overall results of our experiments, in which we compare BPART5 to the rule-based recommendation approach from Chapter 7, are shown in Table 8.3.[4] Considering a recommendation list of length k=10, RULES outperforms BPART5 by 11% in terms of the rigid hit rate Hits@10. However, when considering the semantic hit rates, which recognize that activity recommendations that are semantically similar to the ground-truth activity are also useful for modelers, then BPART5 turns out to be superior. It outperforms RULES by 15%, 24%, and 12% in BLEU@10, METEOR@10, and Cos@10, respectively. Turning to the hit rates for k=1, i.e., the hit rates of the top recommendation of each list, it is equally apparent that RULES performs better in terms of the standard hit rate, whereas BPART5 achieves better results in terms of the semantic hit rates. Thus, the results indicate that RULES is more accurate in giving recommendations that correspond exactly to the ground truth. BPART5 is better in generating recommendations that are not an exact match but have a high semantic similarity to the

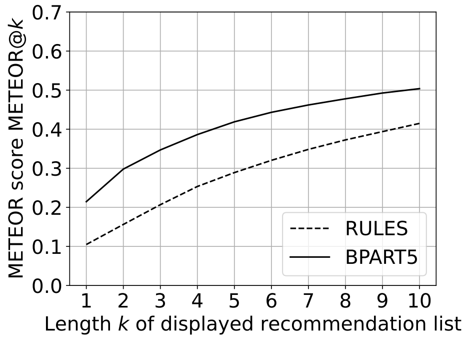
[4]We performed t-tests for all reported differences between the evaluated approaches, which showed that the differences are statistically significant ($p < 0.001$).



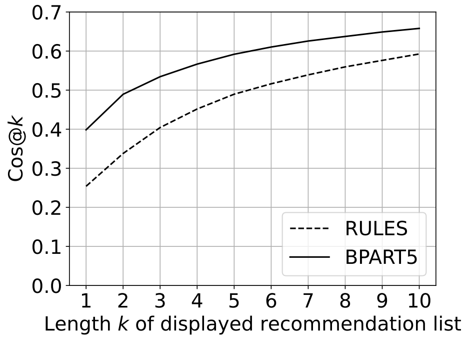
(a) Hit@k



(b) BLEU



(c) METEOR



(d) Cos@k

Figure 8.4: Results for different lengths of the displayed recommendation list

ground truth, though, which means that BPART5 provides in general more relevant recommendations.

Regarding the ranking of suitable activities within a recommendation list, Figure 8.4 shows the courses of the standard hit rate Hits@k and the semantic hit rates BLEU@k, METEOR@k and Cos@k for recommendation lists of lengths k=1 to k=10. Figures 8.4a and 8.4b show that the lines from the Hits@1 to the Hits@10 and from the BLEU@1 to the BLEU@10 values are rather straight. The likelihood of finding a—in terms of these metrics—suitable recommendation thus increases linearly with each additional activity in the recommendation list. In the case of METEOR@k and Cos@k (Figures 8.4c and 8.4d, respectively), the curves rise more steeply for smaller lengths of the recommendation list, which indicates that both approaches are able to rank recommendations that are semantically similar to the ground truth on the first positions of the recommendation list.

Ability to generate new activity labels. To investigate the ability of the approaches to generate new activity labels, i.e., labels that have not been used in the process models used for training, we performed an in-depth analysis of the labels recommended by both approaches. Overall, the approaches made a total of 361.430 label recommendations, which corresponds to the number of evaluation cases (36.143) multiplied by the length of the generated recommendation list per evaluation case (ten). The proportion of recommended labels that are newly generated, i.e., do not exist in the process models in the training dataset, is 0 % for RULES and 36.2 % for BPART5. In the case of RULES, 16.551 of the recommended labels are unique, while BPART5 generated 98.857 unique label recommendations, of which 75.6 % do not exist in the training dataset.

The difference in the unique numbers of generated label recommendations indicates that BPART5 achieves a higher diversity of recommended labels, while RULES is dependent on the knowledge in the process models used for training and thus more limited in its recommendations. Based on the percentages of newly generated labels, it can be inferred that incorporating natural language semantics into RULES enhances the ranking of recommendations, but doesn't necessarily result in the generation of new labels. Additionally, it can be concluded that BPART5 is able to leverage the knowledge contained in the pre-trained language model and recommend activity labels that go beyond the vocabulary of the process models in the training set. On the one hand, BPART5 performs worse in terms of hit rate for this reason, on the other hand, this leads to less dependency on the given process models used for training and therefore a higher semantic accuracy of BPART5.

Handling models with only unseen labels. Finally, we assess how well BPART5 is able to recommend activity labels for business process models that are vastly different from those included in the training set, i.e., that contain only unseen node labels. In general, such cases represent a considerable challenge for activity-recommendation approaches, as they face a recommendation task that is completely unfamiliar to them.

Out of the total of 36.143 evaluation cases, we found 1.726 evaluation cases from 589 process models that meet this criterion, i.e., where none of the node labels in the process model under development were contained in the training data. We evaluated the approaches on this subset in the same manner as in the evaluation on the whole set of evaluation cases.

The results of the study are presented in Table 8.4. While the absolute numbers of the metrics on this subset are naturally low, due to the challenging nature of the cases, the results show that BPART5 clearly outperforms RULES on the subset in terms of all metrics. Although RULES is restricted to the knowledge contained in the process model repository, it is able to generate a few useful recommendations, mainly in the form of default label recommendations. Specifically, it recommends the ten most often used activities of the repository whenever none of the rules

Approach	Hits@10	BLEU@10	METEOR@10	Cos@10
RULES	0.0070	0.0079	0.0628	0.2892
BPART5	**0.0232**	**0.0963**	**0.2112**	**0.4452**

Table 8.4: Results on the subset of evaluation cases with only unseen labels (best results per metric are in bold)

it learned matches the process model under development, as is applicable to the cases at hand. Nevertheless, the difference between the results of both approaches is much larger than on the complete set of evaluation cases. This makes BPART5 the approach of choice when generating recommendations in situations that differ considerably from the available training data.

8.3 Conclusion

In this chapter, we presented the BPART5 approach for activity recommendation, which leverages both formal and natural language semantics contained in process models to enable the application of pre-trained language models. Unlike the rule-based approach, BPART5 is not inherently explainable, since it is based on a transformer-based language model. However, our experiments have demonstrated two key advantages of BPART5.

First, BPART5 consistently outperformed our rule-based approach in terms of providing relevant recommendations, as evidenced by the semantic hit rates. This means that is better in generating recommendations that may not be an exact match with the ground truth but have a high degree of semantic similarity. Second, BPART5 stands out as the first activity-recommendation approach capable of generating label recommendations that extend beyond the vocabulary of the model repository used for training. This is achieved by leveraging a pre-trained language model. Consequently, BPART5 is able to deal with input that significantly deviates from what it has seen before, even when process models consist of unseen labels only. In conclusion, these strengths position BPART5 as the preferred approach, when it comes to recommendations for models with unseen activities.

Chapter 9

Conclusion

In this chapter, we conclude this doctoral thesis. First, in Section 9.1, we provide a summary of the key results obtained throughout our research. Then, in Section 9.2, we offer an outlook on potential directions for future research.

9.1 Summary of Results

In this thesis, we primarily focused on activity recommendation to support business process modeling. To this end, we proposed various approaches that we evaluated in comprehensive experimental studies. Additionally, we published and analyzed the largest publicly available collection of business process models. We employed this collection for our activity-recommendation research and expect it to facilitate the development of approaches for other business process management tasks as well. The key findings and outcomes of this thesis can be summarized as follows:

1. *Publication and analysis of the largest publicly available collection of business process models*: In Chapter 3, we conducted a review of existing support approaches for business process modeling, which assist modelers by recommending activities or process fragments. However, our review revealed a common limitation in the evaluation of these approaches: It often relies on small datasets. Moreover, most of the discussed approaches are infeasible when dealing with large process model datasets. These limitations present a significant issue, given that organizations typically manage thousands of process models in practice. One of the main reasons for these limitations is the lack of access that researchers have to large process model collections. This lack not only affects the evaluation of existing approaches but also presents challenges when developing novel approaches based on large machine learning models, as limited dataset sizes make it difficult to effectively train such models. To

address these issues, we introduced the SAP-SAM dataset in Chapter 4. This dataset contains over one million process models, characterized by a variety of modeling languages, natural languages, and complexity levels, thus providing significant diversity. We have made the SAP-SAM dataset publicly available[1], aiming to foster the development and evaluation of methods and tools for activity recommendation, as well as for other business process management tasks. In addition, we have provided example code to assist users in effectively utilizing this resource[2].

2. *Development of an explainable activity-recommendation approach based on rules*: Even though explanations help users make more informed decisions and choose from the presented alternatives more quickly, this aspect has largely been neglected in the field of activity recommendation. Therefore, in Chapter 5, we introduced an activity-recommendation approach based on rules, which is as such inherently explainable. Specifically, the recommendations can be explained in terms of the rules that generated them. Examining the underlying rules of a recommendation has also proven useful for developers in debugging and refining the approach. Our approach consists of two main phases: rule learning and rule application. The rule-learning phase occurs once for a given repository of process models, while the rule-application phase is repeated throughout the process-modeling task to iteratively provide activity recommendations. Unlike classical top-down or bottom-up approaches, our rule-based approach uses process-oriented rule templates for rule learning. This ensures that we meet the unique requirements of activity recommendation and only learn rules that are relevant for this problem. In the rule application phase, we apply the learned rules to a process model under development and aggregate the recommendations of different rules into a single recommendation list. We can use different methods for this aggregation procedure and also modify the set of rule templates, which highlights the extendable nature of our approach. Through experiments, we demonstrated that our approach outperforms other activity-recommendation approaches in various evaluation procedures. Furthermore, we have shown that our approach can successfully be applied on large process model datasets, indicating its scalability and effectiveness.

3. *Exploration of different approaches to use knowledge graph completion methods for activity recommendation*: In Chapter 6, we addressed the activity-recommendation problem by exploring the use of existing knowledge graph completion methods. Specifically, we suggested transforming the process model repository and the process model under development into a single knowledge graph.

[1]`https://zenodo.org/record/7012043.`

[2]`https://github.com/signavio/sap-sam`

Within this graph, we framed the activity-recommendation task as a completion task. We explored various approaches to construct such a knowledge graph and experimented with both embedding- and rule-based knowledge graph completion methods. Our work is the first to apply knowledge graph completion methods to activity recommendation while considering the entire process model under development as a context for the recommendation. During our investigations, we discovered that standard knowledge graph completion methods lacked the necessary flexibility to adapt to the specific problem at hand. However, by implementing problem-specific filtering as a post-processing step, we were able to enhance the quality of the recommendations. Despite this improvement, our rule-based approach, which was specifically tailored for activity recommendation, consistently outperformed the application of standard knowledge graph completion methods. This suggests that the success of these methods in standard benchmarks may not necessarily translate to a context that deviates from these benchmarks.

4. *Leveraging natural language semantics contained in business process models*: Natural language semantics in business process models are hardly considered in existing activity-recommendation research, even though they are crucial in understanding the context and meaning of included activities. To fill this gap, we dedicated two chapters of this thesis to this topic. First, in Chapter 7, we equipped our rule-based approach with two semantic extensions for rule learning and rule application, which enable a more effective generalization of patterns found in the repository. Specifically, we introduced new rule templates that capture action and business object patterns in process models, and enhanced the rule-application phase to consider semantic similarities of actions and business objects. Second, in Chapter 8, we presented the alternative BPART5 approach that leverages a pre-trained language model to generate activity recommendations, instead of retrieving them from the repository. Through transfer learning, BPART5 benefits from the general-purpose knowledge of the pre-trained language model in addition to the problem-specific knowledge contained in the model repository. Our extensive experiments showed that considering the natural language semantics is a valuable addition and improves the quality of the provided recommendations, especially for process models under development that differ significantly from those in the training repository.

5. *Overcoming vocabulary limitations of the given repository with a transformer-based approach*: An activity-recommendation approach that generates recommendations beyond the vocabulary of the given process model repository adapts more effectively to new recommendation scenarios that were not encountered during training. However, existing approaches can only provide recommenda-

tions in the form of labels contained in the repository. While our rule-based approach with semantic extensions can partially overcome this constraint by combining actions and business objects from the repository into activity recommendations, our BPART5 approach stands out as the first approach to fully overcome this limitation. It is based on the idea of transforming a single activity-recommendation task into one or more sequence-to-sequence tasks in order to employ a transformer-based language model. This idea is employed in both phases of the approach. In the first phase, a transformer-based language model is fine-tuned using sequence-to-sequence tasks derived from the models contained in the repository. In the second phase, a recommendation list is created by aggregating predictions from sequence-to-sequence tasks extracted from the process model under development. Our experiments revealed that our rule-based approach is more accurate than BPART5 in providing recommendations that exactly match the ground truth. On the other hand, they showed that BPART5 is better in generating recommendations that may not be an exact match but are relevant due to their high semantic similarity to the ground truth. This, combined with BPART5's ability to generate recommendations beyond the repository's vocabulary, makes it the preferred approach, when providing recommendations for process models with unseen activities.

6. *Enhanced evaluation through the introduction of new recommendation scenarios, simulation procedures, and metrics*: Our analysis of existing work on support approaches for business process modeling in Chapter 3 revealed that these approaches are often evaluated in a limited scope that lacks practical relevance. This hampers the applicability of experimental findings in broader contexts. For instance, the evaluated recommendation cases are very similar to the training examples. Therefore, we extended the evaluation framework in several ways. In Chapter 5, we introduced the full-breadth procedure for simulating recommendation cases. This novel procedure provides a level of recommendation context that strikes a balance between existing procedures, enabling more diverse evaluation cases. In Chapters 5 and 6, we established two different application scenarios to assess the approaches. The first scenario reflects a situation where the given repository may contain models that are similar to the process model under development. The second scenario simulates a situation where the repository contains fewer models, if any, that resemble the process model under development. In Chapters 7 and 8, we defined new metrics: relaxed versions of Hits@10 and MRR that consider semantic similarities between the ground truth and the recommended labels. These metrics account for the fact that activities can be described in multiple, yet semantically similar, ways. Moreover, in Chapter 8, we evaluated our approaches using the large-scale SAP-SAM

dataset and additionally investigated the performance of our approaches in recommendation cases where the process models under development are vastly different from those in the training set, i.e., contain only unseen labels. Such cases usually present considerable challenges for activity-recommendation approaches, but are of particular interest.

9.2 Future Research

In this thesis, we proposed different activity-recommendation approaches and presented a dataset that can be used to train and evaluate such, as well as other, approaches. Future research could focus on *improving*, *extending*, and *evaluating* our work in several directions. In the following, we outline seven such directions.

Improving. To begin with, the proposed approaches could be improved by addressing their limitations. This includes (i) the aggregation method used in the application phase of our rule-based approach, (ii) the consideration of recommendation context in our approaches that use Knowledge Graph Embedding (KGE) models, and (iii) the sequence length for the derivation of sequence-to-sequence tasks in our BPART5 approach:

(i) *Aggregation method.* Rule aggregation methods, which aggregate the confidence scores of different rules that make the same prediction, represent an active field of research [8]. Our rule-based approach employed two simple yet effective aggregation methods: maximum and noisy-or aggregation. To improve the ranking of the provided recommendations, novel aggregation methods could be tested. This could be particularly useful for the integration of the recommendations stemming from our similarity-based extension.

(ii) *Recommendation context.* In our approaches that use KGE models for the activity-recommendation problem, we incorporated the current state of the process model under development as a context for the recommendation into the training set. Consequently, the KGE models need to be retrained after every activity that has been added to the process model under development, which is very time-consuming. To address this issue, future research could explore ways to avoid the need for complete retraining, e.g., by employing approaches from the field of continual learning [155]. In combination with alternative, not yet tested, techniques working on knowledge graphs, such as Graph Convolutional Networks [143], RDF2Vec [133], or KG-Bert [189], this could lead to effective approaches for activity recommendation.

(iii) *Sequence length.* Our BPART5 approach converts an activity-recommendation task into one or more sequence-to-sequence tasks. The extracted sequences have a pre-defined length, which necessitates node sequences of a specific length to generate recommendations. In future research, it would be valuable to explore the potential of using not only sequences of a specific length, but also arbitrary sequences from the process model under development to generate activity recommendations.

Extending. Moreover, there are untapped sources of information that could be used by future research to extend our work in several regards. This includes (iv) dictionary entries linked to process models and (v) additional process information:

(iv) *Dictionary entries.* The SAP-SAI platform, which was used to create the process models contained in SAP-SAM, supports the creation of dictionary entries. These dictionary entries represent various entities such as organizational roles, documents, or IT systems, and can be linked to models for reuse across a process landscape. The SAP-SAM dataset could be augmented by including the dictionary entries, allowing future work to better understand and analyze complex process landscapes.

(v) *Process information.* Our approaches could be enhanced by incorporating information beyond activity labels and their inter-relations into the recommendation procedure, if available. Potential sources for such information include edge labels, resources associated with activities, or process model annotations. In the case of our rule-based approach, new rule templates could be introduced to incorporate this information. When applying knowledge graph completion methods, additional information could be considered during the construction of the knowledge graph. The BPART5 approach could be enriched by integrating the information into the verbalization process.

Our work could also be extended by (vi) developing an ensemble method that combines our approaches to obtain better recommendations:

(vi) *Ensemble method.* In this thesis, we developed two distinct main approaches, each with its own strengths: a rule-based approach and the transformer-based BPART5 approach. In our experiments, the rule-based approach performed well in generating recommendations that match the ground truth. On the other hand, the transformer-based approach demonstrated its proficiency in adapting to new recommendation scenarios, providing relevant recommendations that have a high semantic similarity to the ground truth. By combining both approaches into a unified ensemble method, potentially a superior recommendation method, which performs well across various metrics and recommendation scenarios, could be developed. A possible approach to constructing such an ensemble method could draw inspiration from the work of Meilicke et al. [105], who successfully developed a method for combining rule-based and embedding-based approaches in the domain of knowledge graph completion through a post-processing step.

Evaluating. Finally, online experiments and user studies could be conducted to (vii) evaluate the proposed approaches using user feedback:

(vii) *User feedback.* In this thesis, we used offline experiments to evaluate activity-recommendation approaches. Specifically, we employed various procedures to simulate varying states of process models under development from completed models. This allowed us to effectively evaluate multiple approaches at large scale. Our findings could be used to identify a group of promising approaches and configurations, which can be further tested through more costly online experiments or user studies [51], investigating the perceived usefulness of the recommendations.

Bibliography

[1] Alain Abran, James W Moore, Pierre Bourque, Robert Dupuis, and L Tripp. Software engineering body of knowledge. *IEEE Computer Society, Angela Burgess*, 25, 2004.

[2] Sanae Achsas and El Habib Nfaoui. Language representation learning models: A comparative study. In *Proceedings of the 13th International Conference on Intelligent Systems: Theories and Applications*, pages 1–7, 2020.

[3] Grigoris Antoniou and Frank van Harmelen. Web ontology language: Owl. *Handbook on ontologies*, pages 91–110, 2009.

[4] Goncalo Antunes, Marzieh Bakhshandelh, Jose Borbinha, Joao Cardoso, Sharam Dadashnia, et al. The process model matching contest 2015. In *Enterprise Modelling and Information Systems Architectures*, pages 127–155. Köllen, 2015.

[5] Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. Neural machine translation by jointly learning to align and translate. In *International Conference on Learning Representations*, 2015.

[6] Satanjeev Banerjee and Alon Lavie. Meteor: An automatic metric for mt evaluation with improved correlation with human judgments. In *Proceedings of the ACL workshop on intrinsic and extrinsic evaluation measures for machine translation and/or summarization*, pages 65–72, 2005.

[7] Jörg Becker, Michael Rosemann, and Christoph Von Uthmann. Guidelines of business process modeling. In *Business Process Management: Models, Techniques, and Empirical Studies*, pages 30–49. Springer, 2002.

[8] Patrick Betz, Christian Meilicke, and Heiner Stuckenschmidt. Supervised knowledge aggregation for knowledge graph completion. In *European Semantic Web Conference*, pages 74–92. Springer, 2022.

[9] Barry W Boehm and Philip N. Papaccio. Understanding and controlling software costs. *IEEE transactions on software engineering*, 14(10):1462–1477, 1988.

[10] Kurt Bollacker, Patrick Tufts, Tomi Pierce, and Robert Cook. A platform for scalable, collaborative, structured information integration. In *Intl. Workshop on Information Integration on the Web (IIWeb'07)*, pages 22–27. Citeseer, 2007.

[11] Antoine Bordes, Nicolas Usunier, Alberto García-Durán, Jason Weston, and Oksana Yakhnenko. Translating embeddings for modeling multi-relational data. In *NIPS*, pages 2787–2795, 2013.

[12] Matthias Born, Christian Brelage, Ivan Markovic, Daniel Pfeiffer, and Ingo Weber. Auto-completion for executable business process models. In *Business Process Management Workshops: BPM 2008 International Workshops, Milano, Italy, September 1-4, 2008. Revised Papers 6*, pages 510–515. Springer, 2009.

[13] Dan Brickley, Ramanathan V Guha, and Brian McBride. Rdf schema 1.1. *W3C recommendation*, 25(2004-2014):10, 2014.

[14] Samuel Broscheit, Daniel Ruffinelli, Adrian Kochsiek, Patrick Betz, and Rainer Gemulla. LibKGE - A knowledge graph embedding library for reproducible research. In *EMNLP: System Demonstrations*, pages 165–174, 2020.

[15] Ross Brown, Jan Recker, and Stephen West. Using virtual worlds for collaborative business process modeling. *Business Process Management Journal*, 2011.

[16] Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. Language models are few-shot learners. *NeurIPS*, 33:1877–1901, 2020.

[17] Erik Bruchez, Alain Couthures, Steven Pemberton, and Nick Van den Bleeken. XForms 2.0. W3C Working Draft, World Wide Web Consortium (W3C), 2012.

[18] Liliana Cabral, Barry Norton, and John Domingue. The business process modelling ontology. In *Proceedings of the 4th international workshop on semantic business process management*, pages 9–16, 2009.

[19] Bin Cao, Jianwei Yin, ShuiGuang Deng, Dongjing Wang, and Zhaohui Wu. Graph-based workflow recommendation: on improving business process modeling. In *CIKM*, pages 1527–1531. ACM, 2012.

[20] Daniel Cer, Yinfei Yang, Sheng-yi Kong, Nan Hua, Nicole Limtiaco, Rhomni St. John, Noah Constant, Mario Guajardo-Cespedes, Steve Yuan, Chris Tar, Yun-Hsuan Sung, Brian Strope, and Ray Kurzweil. Universal sentence encoder, 2018.

[21] Nguyen Ngoc Chan, Walid Gaaloul, and Samir Tata. Context-based service recommendation for assisting business process design. In *E-Commerce and Web Technologies: 12th International Conference, EC-Web 2011, Toulouse, France, August 30-September 1, 2011. Proceedings 12*, pages 39–51. Springer, 2011.

[22] Peter Pin-Shan Chen. The entity-relationship model—toward a unified view of data. *ACM Trans. Database Syst.*, 1(1):9–36, 1976.

[23] Zhe Chen, Yuehan Wang, Bin Zhao, Jing Cheng, Xin Zhao, and Zongtao Duan. Knowledge graph completion: A review. *Ieee Access*, 8:192435–192456, 2020.

[24] Michele Chinosi and Alberto Trombetta. Bpmn: An introduction to the standard. *Computer Standards & Interfaces*, 34(1):124–134, 2012.

[25] Kyunghyun Cho, Bart van Merrienboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. Learning phrase representations using rnn encoder–decoder for statistical machine translation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, page 1724. Association for Computational Linguistics, 2014.

[26] Flavio Corradini, Fabrizio Fornari, Andrea Polini, Barbara Re, and Francesco Tiezzi. Reprository: a repository platform for sharing business process models. In *BPM (PhD/Demos)*, pages 149–153, 2019.

[27] Thomas H Davenport and James E Short. The new industrial engineering: Information technology and business process redesign. *MIT Sloan Management Review*, 1990.

[28] Luc De Raedt. *Logical and Relational Learning*. Springer, 2008.

[29] Luc De Raedt. *Inductive Logic Programming.* Springer, 2010.

[30] Li Deng and Yang Liu. A joint introduction to natural language processing and to deep learning. *Deep learning in natural language processing*, pages 1–22, 2018.

[31] ShuiGuang Deng, Dongjing Wang, Ying Li, Bin Cao, Jianwei Yin, Zhaohui Wu, and Mengchu Zhou. A recommendation system to facilitate business process modeling. *IEEE Trans Cybern*, 47(6):1380–1394, 2017.

[32] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. BERT: Pre-training of deep bidirectional transformers for language understanding. In *NAACL-HLT*, pages 4171–4186, 2019.

[33] Remco Dijkman, Marlon Dumas, and Luciano García-Bañuelos. Graph matching algorithms for business process model similarity search. In *Business Process Management: 7th International Conference, BPM 2009, Ulm, Germany, September 8-10, 2009. Proceedings 7*, pages 48–63. Springer, 2009.

[34] Remco Dijkman, Marlon Dumas, Boudewijn Van Dongen, Reina Käärik, and Jan Mendling. Similarity of business process models: Metrics and evaluation. *Information Systems*, 36(2):498–516, 2011.

[35] Remco Dijkman, Marcello La Rosa, and Hajo Reijers. Managing large collections of business process models-current techniques and challenges. *Computers in Industry*, 63(2):91–97, 2012.

[36] Remco M. Dijkman, Marlon Dumas, and Chun Ouyang. Semantics and analysis of business process models in BPMN. *Inf. Softw. Technol.*, 50(12):1281–1294, 2008.

[37] Marlon Dumas, Marcello La Rosa, Jan Mendling, and Hajo A. Reijers. *Fundamentals of Business Process Management.* Springer, 2013.

[38] Wafaa S El-Kassas, Cherif R Salama, Ahmed A Rafea, and Hoda K Mohamed. Automatic text summarization: A comprehensive survey. *Expert systems with applications*, 165:113679, 2021.

[39] Linus Ericsson, Henry Gouk, Chen Change Loy, and Timothy M. Hospedales. Self-supervised representation learning: Introduction, advances, and challenges. *IEEE Signal Processing Magazine*, 39(3):42–62, 2022.

[40] Christiane Fellbaum. Wordnet. In *Theory and applications of ontology: computer applications*, pages 231–243. Springer, 2010.

[41] M. Fellmann, Patrick Delfmann, A. Koschmider, R. Laue, H. Leopold, and Andreas Schoknecht. Semantic technology in business process modeling and analysis. part 1: Matching, modeling support, correctness and compliance. *EMISA Forum*, 35:15–31, 2015.

[42] Michael Fellmann, Agnes Koschmider, Ralf Laue, Andreas Schoknecht, and Arthur Vetter. Business process model patterns: state-of-the-art, research classification and taxonomy. *Business Process Management Journal*, 2018.

[43] Paul JM Frederiks and Th P Van der Weide. Information modeling: The process and the required competencies of its participants. *Data & Knowledge Engineering*, 58(1):4–20, 2006.

[44] Fabian Friedrich, Jan Mendling, and Frank Puhlmann. Process model generation from natural language text. In *Advanced Information Systems Engineering: 23rd International Conference, CAiSE 2011, London, UK, June 20-24, 2011. Proceedings 23*, pages 482–496. Springer, 2011.

[45] Johannes Fürnkranz, Dragan Gamberger, and Nada Lavrac. *Foundations of Rule Learning*. Cognitive Technologies. Springer, 2012.

[46] Luis Antonio Galárraga, Christina Teflioudi, Katja Hose, and Fabian Suchanek. AMIE: Association rule mining under incomplete evidence in ontological knowledge bases. In *WWW*, pages 413–422, 2013.

[47] Andrea Gasparetto, Matteo Marcuzzo, Alessandro Zangari, and Andrea Albarelli. A survey on text classification algorithms: From text to predictions. *Information*, 13(2):83, 2022.

[48] Maayan Goldstein and Cecilia González-Álvarez. Augmenting modelers with semantic autocompletion of processes. In *BPM Forum*, pages 20–36. Springer, 2021.

[49] Maayan Goldstein and Cecilia González-Álvarez. Evaluating semantic autocompletion of business processes with domain experts. In *ASE*, pages 1116–1120, 2021.

[50] Alex Graves. Sequence transduction with recurrent neural networks. *arXiv preprint arXiv:1211.3711*, 2012.

[51] Asela Gunawardana, Guy Shani, and Sivan Yogev. Evaluating recommender systems. In *Recommender systems handbook*, pages 547–601. Springer, 2012.

[52] Asela Gunawardana, Guy Shani, and Sivan Yogev. *Evaluating Recommender Systems*, pages 547–601. Springer US, New York, NY, 2022.

[53] Qingyu Guo, Fuzhen Zhuang, Chuan Qin, Hengshu Zhu, Xing Xie, Hui Xiong, and Qing He. A survey on knowledge graph-based recommender systems. *IEEE Transactions on Knowledge and Data Engineering*, 34(8):3549–3568, 2020.

[54] Michael Hammer. What is business process management? In *Handbook on business process management 1: Introduction, methods, and information systems*, pages 3–16. Springer, 2014.

[55] Michael Hammer and James Champy. *Reengineering the corporation: A Manifesto for business revolution*. Zondervan, 2009.

[56] Xu Han, Zhengyan Zhang, Ning Ding, Yuxian Gu, Xiao Liu, Yuqi Huo, Jiezhong Qiu, Yuan Yao, Ao Zhang, Liang Zhang, et al. Pre-trained models: Past, present and future. *AI Open*, 2:225–250, 2021.

[57] David J Hand. Pattern detection and discovery. In *Pattern Detection and Discovery: ESF Exploratory Workshop London, UK, September 16–19, 2002 Proceedings*, pages 1–12. Springer, 2002.

[58] Joachim Herbst and D Karagiannis. An inductive approach to the acquisition and adaptation of workflow models. In *Proceedings of the IJCAI*, volume 99, pages 52–57. Citeseer, 1999.

[59] Jonathan L Herlocker, Joseph A Konstan, and John Riedl. Explaining collaborative filtering recommendations. In *Proceedings of the 2000 ACM conference on Computer supported cooperative work*, pages 241–250, 2000.

[60] Sepp Hochreiter, Yoshua Bengio, Paolo Frasconi, Jürgen Schmidhuber, et al. Gradient flow in recurrent nets: the difficulty of learning long-term dependencies, 2001.

[61] Johannes Hoffart, Fabian M Suchanek, Klaus Berberich, Edwin Lewis-Kelham, Gerard De Melo, and Gerhard Weikum. Yago2: exploring and querying world knowledge in time, space, context, and many languages. In *Proceedings of the 20th international conference companion on World wide web*, pages 229–232, 2011.

[62] Aidan Hogan, Eva Blomqvist, Michael Cochez, Claudia d'Amato, Gerard De Melo, Claudio Gutierrez, Sabrina Kirrane, José Emilio Labra Gayo,

Roberto Navigli, Sebastian Neumaier, et al. Knowledge graphs. *ACM Computing Surveys (Csur)*, 54(4):1–37, 2021.

[63] Ari Holtzman, Jan Buys, Li Du, Maxwell Forbes, and Yejin Choi. The curious case of neural text degeneration. In *International Conference on Learning Representations*, 2019.

[64] Matthew Honnibal, Ines Montani, Sofie Van Landeghem, and Adriane Boyd. spaCy: Industrial-strength Natural Language Processing in Python, 2020.

[65] Thomas Hornung, Agnes Koschmider, and Georg Lausen. Recommendation based process modeling support: Method and user experience. In *Proceedings of the 27th International Conference on Conceptual Modeling*, pages 265–278, 2008.

[66] Thomas Hornung, Agnes Koschmider, and Andreas Oberweis. Rule-based autocompletion of business process models. In *CAiSE Forum*, volume 247, pages 222–232, 2007.

[67] Thomas Hornung, Agnes Koschmider, and Andreas Oberweis. A recommender system for business process models. In *17th Annual Workshop on Information Technologies & Systems (WITS)*, 2009.

[68] Ian Horrocks, Peter F Patel-Schneider, Harold Boley, Said Tabet, Benjamin Grosof, Mike Dean, et al. Swrl: A semantic web rule language combining owl and ruleml. *W3C Member submission*, 21(79):1–31, 2004.

[69] Constantin Houy, Peter Fettke, Peter Loos, Wil MP van der Aalst, and John Krogstie. Business process management in the large. *Business & Information Systems Engineering*, 3(6):385–388, 2011.

[70] Marta Indulska, Peter Green, Jan Recker, and Michael Rosemann. Business process modeling: Perceived benefits. In *Conceptual Modeling-ER 2009: 28th International Conference on Conceptual Modeling, Gramado, Brazil, November 9-12, 2009. Proceedings 28*, pages 458–471. Springer, 2009.

[71] Daphne Ippolito, Reno Kriz, Joao Sedoc, Maria Kustikova, and Chris Callison-Burch. Comparison of diverse decoding methods from conditional language models. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*. Association for Computational Linguistics, 2019.

[72] Dietmar Jannach and Simon Fischer. Recommendation-based modeling support for data mining processes. In *RecSys*, pages 337–340, 2014.

[73] Dietmar Jannach, Michael Jugovac, and Lukas Lerche. Supporting the design of machine learning workflows with a recommendation system. *ACM Transactions on Interactive Intelligent Systems (TiiS)*, 6(1):1–35, 2016.

[74] Kurt Jensen. Coloured petri nets. In *Petri nets: central models and their properties*, pages 248–299. Springer, 1987.

[75] Shaoxiong Ji, Shirui Pan, Erik Cambria, Pekka Marttinen, and S Yu Philip. A survey on knowledge graphs: Representation, acquisition, and applications. *IEEE transactions on neural networks and learning systems*, 33(2):494–514, 2021.

[76] Dimitris Karagiannis and Harald Kühn. Metamodelling platforms. In *E-Commerce and Web Technologies: Third International Conference, EC-Web 2002 Aix-en-Provence, France, September 2–6, 2002 Proceedings 3*, pages 182–182. Springer, 2002.

[77] Diksha Khurana, Aditya Koli, Kiran Khatter, and Sukhdev Singh. Natural language processing: State of the art, current trends and challenges. *Multimedia tools and applications*, 82(3):3713–3744, 2023.

[78] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In *ICLR*, 2015.

[79] Guillaume Klein, Yoon Kim, Yuntian Deng, Jean Senellart, and Alexander M Rush. Opennmt: Open-source toolkit for neural machine translation. In *Proceedings of ACL 2017, System Demonstrations*, pages 67–72, 2017.

[80] Takeshi Kojima, Shixiang Shane Gu, Machel Reid, Yutaka Matsuo, and Yusuke Iwasawa. Large language models are zero-shot reasoners. In *ICML Workshop KRLM*, 2022.

[81] Agnes Koschmider, Thomas Hornung, and Andreas Oberweis. Recommendation-based editor for business process modeling. *Data & Knowledge Engineering*, 70(6):483–503, 2011.

[82] Agnes Koschmider and Andreas Oberweis. Designing business processes with a recommendation-based editor. *Handbook on Business Process Management 1: Introduction, Methods, and Information Systems*, pages 299–312, 2010.

[83] Taku Kudo. Subword regularization: Improving neural network translation models with multiple subword candidates. In Iryna Gurevych and Yusuke

Miyao, editors, *ACL (1)*, pages 66–75. Association for Computational Linguistics, 2018.

[84] Taku Kudo and John Richardson. Sentencepiece: A simple and language independent subword tokenizer and detokenizer for neural text processing. *CoRR*, abs/1808.06226, 2018.

[85] Jonathan Lajus, Luis Galárraga, and Fabian Suchanek. Fast and exact rule mining with amie 3. In *The Semantic Web: 17th International Conference, ESWC 2020, Heraklion, Crete, Greece, May 31–June 4, 2020, Proceedings 17*, pages 36–52. Springer, 2020.

[86] M. M. Lankhorst, H. A. Proper, and H. Jonkers. The architecture of the archimate language. In *Enterprise, Business-Process and Information Systems Modeling*, pages 367–380. Springer, 2009.

[87] Jens Lehmann, Robert Isele, Max Jakob, Anja Jentzsch, Dimitris Kontokostas, Pablo N Mendes, Sebastian Hellmann, Mohamed Morsey, Patrick Van Kleef, Sören Auer, et al. Dbpedia–a large-scale, multilingual knowledge base extracted from wikipedia. *Semantic web*, 6(2):167–195, 2015.

[88] Henrik Leopold. *Natural Language in Business Process Models*. Springer, 2013.

[89] Henrik Leopold, Rami-Habib Eid-Sabbagh, Jan Mendling, Leonardo Guerreiro Azevedo, and Fernanda Araujo Baiao. Detection of naming convention violations in process models for different languages. *Decision Support Systems*, 56:310–325, 2013.

[90] Henrik Leopold, Jan Mendling, Hajo A Reijers, and Marcello La Rosa. Simplifying process model abstraction: Techniques for generating model names. *Information Systems*, 39:134–151, 2014.

[91] Henrik Leopold, Sergey Smirnov, and Jan Mendling. On the refactoring of activity labels in business process models. *Information Systems*, 37(5):443–459, 2012.

[92] Henrik Leopold, Han van der Aa, Jelmer Offenberg, and Hajo A Reijers. Using hidden markov models for the accurate linguistic analysis of process model activity labels. *Information Systems*, 83:30–39, 2019.

[93] Baoli Li and Liping Han. Distance weighted cosine similarity measure for text classification. In *International conference on intelligent data engineering and automated learning*, pages 611–618. Springer, 2013.

[94] Jing Li, Aixin Sun, Jianglei Han, and Chenliang Li. A survey on deep learning for named entity recognition. *IEEE Transactions on Knowledge and Data Engineering*, 34(1):50–70, 2020.

[95] Y. Li, B. Cao, L. Xu, J. Yin, S. Deng, Y. Yin, and Z. Wu. An efficient recommendation method for improving business process modeling. *IEEE Transactions on Industrial Informatics*, 10(1):502–513, 2014.

[96] Yankai Lin, Zhiyuan Liu, Maosong Sun, Yang Liu, and Xuan Zhu. Learning entity and relation embeddings for knowledge graph completion. In *Proceedings of the AAAI conference on artificial intelligence*, volume 29, 2015.

[97] Pengfei Liu, Weizhe Yuan, Jinlan Fu, Zhengbao Jiang, Hiroaki Hayashi, and Graham Neubig. Pre-train, prompt, and predict: A systematic survey of prompting methods in natural language processing. *ACM Computing Surveys*, 55(9):1–35, 2023.

[98] Xiao Liu, Fanjin Zhang, Zhenyu Hou, Li Mian, Zhaoyu Wang, Jing Zhang, and Jie Tang. Self-supervised learning: Generative or contrastive. *IEEE Transactions on Knowledge and Data Engineering*, 35(1):857–876, 2023.

[99] Yang Liu and Mirella Lapata. Text summarization with pretrained encoders. In *EMNLP-IJCNLP*, pages 3730–3740. Association for Computational Linguistics, 2019.

[100] Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. In *International Conference on Learning Representations*, 2018.

[101] Ruopeng Lu and Shazia Sadiq. A survey of comparative business process modeling approaches. In *Business Information Systems: 10th International Conference, BIS 2007, Poznan, Poland, April 25-27, 2007. Proceedings 10*, pages 82–94. Springer, 2007.

[102] Steffen Mazanek and Mark Minas. Business process models as a showcase for syntax-based assistance in diagram editors. In *MoDELS*, pages 322–336. Springer, 2009.

[103] Steffen Mazanek, Christian Rutetzki, and Mark Minas. Sketch-based diagram editors with user assistance based on graph transformation and graph drawing techniques. *Electronic Communications of the EASST*, 32, 2011.

[104] David McSherry. Explanation in recommender systems. *Artificial Intelligence Review*, 24:179–197, 2005.

[105] Christian Meilicke, Patrick Betz, and Heiner Stuckenschmidt. Why a naive way to combine symbolic and latent knowledge base completion works surprisingly well. In *3rd Conference on Automated Knowledge Base Construction*, 2021.

[106] Christian Meilicke, Melisachew Wudage Chekol, Daniel Ruffinelli, and Heiner Stuckenschmidt. Anytime bottom-up rule learning for knowledge graph completion. In *IJCAI*, pages 3137–3143, 2019.

[107] Jan Mendling. *Metrics for process models: empirical foundations of verification, error prediction, and guidelines for correctness*, volume 6. Springer Science & Business Media, 2008.

[108] Jan Mendling, Hajo A Reijers, and Jan Recker. Activity labeling in process modeling: Empirical insights and recommendations. *Information Systems*, 35(4):467–482, 2010.

[109] Jan Mendling, Hajo A. Reijers, and Jan Recker. Activity labeling in process modeling: Empirical insights and recommendations. *Inf. Syst.*, 35(4):467–482, 2010.

[110] Jan Mendling, Hajo A Reijers, and Wil MP van der Aalst. Seven process modeling guidelines (7pmg). *Information and software technology*, 52(2):127–136, 2010.

[111] Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. Efficient estimation of word representations in vector space. *Proceedings of Workshop at ICLR*, 2013.

[112] Bonan Min, Hayley Ross, Elior Sulem, Amir Pouran Ben Veyseh, Thien Huu Nguyen, Oscar Sainz, Eneko Agirre, Ilana Heintz, and Dan Roth. Recent advances in natural language processing via large pre-trained language models: A survey. *ACM Computing Surveys*, 2021.

[113] Philip Moore and Hai Van Pham. On context and the open world assumption. In *2015 IEEE 29th International Conference on Advanced Information Networking and Applications Workshops*, pages 387–392. IEEE, 2015.

[114] Isel Moreno-Montes de Oca and Monique Snoeck. Pragmatic guidelines for business process modeling. *Available at SSRN 2592983*, 2014.

[115] Michael zur Muehlen and Jan Recker. How much language is enough? theoretical and practical use of the business process modeling notation. In

Seminal Contributions to Information Systems Engineering, pages 429–443. Springer, 2013.

[116] Khalid Nassiri and Moulay Akhloufi. Transformer models used for text-based question answering systems. *Applied Intelligence*, 53(9):10602–10635, 2023.

[117] Maximilian Nickel, Volker Tresp, Hans-Peter Kriegel, et al. A three-way model for collective learning on multi-relational data. In *Icml*, volume 11, pages 3104482–3104584, 2011.

[118] OMG. Business Process Model and Notation (BPMN), Version 2.0.2, 2013.

[119] OMG. Case Management Model and Notation (CMMN), Version 1.1, 2016.

[120] OMG. Decision Model and Notation (DMN), Version 1.3, 2021.

[121] Simon Ott, Christian Meilicke, and Matthias Samwald. Safran: An interpretable, rule-based link prediction method outperforming embedding models. In *3rd Conference on Automated Knowledge Base Construction*, 2021.

[122] Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. Bleu: a method for automatic evaluation of machine translation. In *Proceedings of the 40th annual meeting of the Association for Computational Linguistics*, pages 311–318, 2002.

[123] Romain Paulus, Caiming Xiong, and Richard Socher. A deep reinforced model for abstractive summarization. In *International Conference on Learning Representations*, 2018.

[124] Jeffrey Pennington, Richard Socher, and Christopher D. Manning. Glove: Global vectors for word representation. In *Empirical Methods in Natural Language Processing (EMNLP)*, pages 1532–1543, 2014.

[125] Carl Adam Petri. Kommunikation mit automaten. *Westfäl. Inst. f. Instrumentelle Mathematik an der Univ. Bonn*, 1962.

[126] Fabian Pittke, Henrik Leopold, and Jan Mendling. Automatic detection and resolution of lexical ambiguity in process models. *IEEE Transactions on Software Engineering*, 41(6):526–544, 2015.

[127] Artem Polyvyanyy. Process querying: Methods, techniques, and applications. In Artem Polyvyanyy, editor, *Process Querying Methods*, pages 511–524. Springer, 2022.

[128] J. Ross Quinlan. Learning logical definitions from relations. *Machine learning*, 5:239–266, 1990.

[129] Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J. Liu. Exploring the limits of transfer learning with a unified text-to-text transformer. *Journal of Machine Learning Research*, 21(140):1–67, 2020.

[130] Adrian Rebmann and Han Van der Aa. Extracting semantic process information from the natural language in event logs. In *CAiSE*. Springer, 2021.

[131] Jana-Rebecca Rehse, Peter Fettke, and Peter Loos. A graph-theoretic method for the inductive development of reference process models. *Software & Systems Modeling*, 16(3):833–873, 2017.

[132] Hajo A Reijers, Selma Limam, and Wil MP Van Der Aalst. Product-based workflow design. *Journal of management information systems*, 20(1):229–262, 2003.

[133] Petar Ristoski and Heiko Paulheim. Rdf2vec: Rdf graph embeddings for data mining. In *ISWC*, pages 498–514. Springer, 2016.

[134] Michael Rosemann. Potential pitfalls of process modeling: part a. *Business Process Management Journal*, 12(2):249–254, 2006.

[135] Michael Rosemann. Potential pitfalls of process modeling: part b. *Business Process Management Journal*, 12(3):377–384, 2006.

[136] Daniel Ruffinelli, Samuel Broscheit, and Rainer Gemulla. You CAN teach an old dog new tricks! On training knowledge graph embeddings. In *ICLR*. OpenReview.net, 2020.

[137] Stuart Russell and Peter Norvig. *Artificial Intelligence: A Modern Approach*. Prentice Hall, 3 edition, 2010.

[138] Apoorv Saxena, Adrian Kochsiek, and Rainer Gemulla. Sequence-to-sequence knowledge graph completion and question answering. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 2814–2828, 2022.

[139] Bernhard Schäfer, Han van der Aa, Henrik Leopold, and Heiner Stuckenschmidt. Sketch2bpmn: Automatic recognition of hand-drawn bpmn models. In *Advanced Information Systems Engineering*. Springer, 2021.

[140] August-Wilhelm Scheer. *ARIS—business process modeling*. Springer Science & Business Media, 2000.

[141] August-Wilhelm Scheer, Oliver Thomas, and Otmar Adam. *Process Modeling using Event-Driven Process Chains*, chapter 6, pages 119–145. John Wiley & Sons, Ltd, 2005.

[142] Timo Schick and Hinrich Schütze. It's not just size that matters: Small language models are also few-shot learners. In *NAACL-HLT*, pages 2339–2352, 2021.

[143] Michael Schlichtkrull, Thomas N. Kipf, Peter Bloem, Rianne van den Berg, Ivan Titov, and Max Welling. Modeling relational data with graph convolutional networks. In *ISWC*, pages 593–607. Springer, 2018.

[144] David Schumm, Frank Leymann, Zhilei Ma, Thorsten Scheibler, and Steve Strauch. Integrating compliance into business processes: Process fragments as reusable compliance controls. In *Proceedings of the Multikonferenz Wirtschaftsinformatik, MKWI'10, 23-25 February 2010, Göttingen, Germany*. Universitätsverlag Göttingen, 2010.

[145] Baoxu Shi and Tim Weninger. Open-world knowledge graph completion. In *Proceedings of the AAAI conference on artificial intelligence*, volume 32, 2018.

[146] Thiago Silveira, Min Zhang, Xiao Lin, Yiqun Liu, and Shaoping Ma. How good your recommender system is? a survey on evaluations in recommendation. *International Journal of Machine Learning and Cybernetics*, 10:813–831, 2019.

[147] Amit Singhal. Introducing the knowledge graph: Things, not strings. Available at `https://www.blog.google/products/search/introducing-knowledge-graph-things-not/` (2023/08/02).

[148] Rashmi Sinha and Kirsten Swearingen. The role of transparency in recommender systems. In *CHI'02 extended abstracts on Human factors in computing systems*, pages 830–831, 2002.

[149] Sergey Smirnov, Matthias Weidlich, Jan Mendling, and Mathias Weske. Action patterns in business process model repositories. *Computers in Industry*, 63(2):98–111, 2012.

[150] Diana Sola, Christian Meilicke, Han van der Aa, and Heiner Stuckenschmidt. On the use of knowledge graph completion methods for activity recommendation in business process modeling. In *International Conference on Business Process Management*, pages 5–17. Springer, 2021.

[151] Diana Sola, Christian Meilicke, Han van der Aa, and Heiner Stuckenschmidt. A rule-based recommendation approach for business process modeling. In *International Conference on Advanced Information Systems Engineering*, pages 328–343. Springer, 2021.

[152] Diana Sola, Han van der Aa, Christian Meilicke, and Heiner Stuckenschmidt. Exploiting label semantics for rule-based activity recommendation in business process modeling. *Information Systems*, 108:102049, 2022.

[153] Diana Sola, Han van der Aa, Christian Meilicke, and Heiner Stuckenschmidt. Activity recommendation for business process modeling with pretrained language models. In *European Semantic Web Conference*, pages 316–334. Springer, 2023.

[154] Diana Sola, Christian Warmuth, Bernhard Schäfer, Peyman Badakhshan, Jana-Rebecca Rehse, and Timotheus Kampik. Sap signavio academic models: A large process model dataset. In Marco Montali, Arik Senderovich, and Matthias Weidlich, editors, *Process Mining Workshops*, pages 453–465, Cham, 2023. Springer Nature Switzerland.

[155] Hyun-Je Song and Seong-Bae Park. Enriching translation-based knowledge graph embeddings through continual learning. *IEEE Access*, 6:60489–60497, 2018.

[156] Felix Stahlberg. Neural machine translation: A review. *Journal of Artificial Intelligence Research*, 69:343–418, 2020.

[157] Fabian M Suchanek, Jonathan Lajus, Armand Boschin, and Gerhard Weikum. Knowledge representation and rule mining in entity-centric knowledge bases. *Reasoning Web. Explainable Artificial Intelligence: 15th International Summer School 2019, Bolzano, Italy, September 20–24, 2019, Tutorial Lectures*, pages 110–152, 2019.

[158] Kaili Sun, Xudong Luo, and Michael Y Luo. A survey of pretrained language models. In *International Conference on Knowledge Science, Engineering and Management*, pages 442–456. Springer, 2022.

[159] Ilya Sutskever, Oriol Vinyals, and Quoc V Le. Sequence to sequence learning with neural networks. *Advances in neural information processing systems*, 27, 2014.

[160] Ilya Sutskever, Oriol Vinyals, and Quoc V. Le. Sequence to sequence learning with neural networks. In *NIPS*, 2014.

[161] Tom Thaler, Jürgen Walter, Peyman Ardalani, Peter Fettke, and Peter Loos. The need for process model corpora. In *FMI 2014*, page 14, 2014.

[162] Oliver Thomas and Michael Fellmann. Semantic process modeling - design and implementation of an ontology-based representation of business processes. *Business & Information Systems Engineering*, 1(6):438–451, 2009.

[163] Nava Tintarev and Judith Masthoff. A survey of explanations in recommender systems. In *2007 IEEE 23rd international conference on data engineering workshop*, pages 801–810. IEEE, 2007.

[164] Jasmin Türker, Michael Völske, and Thomas S Heinze. Bpmn in the wild: A reprise. In *ZEUS*, pages 68–75, 2022.

[165] Han Van der Aa, Josep Carmona Vargas, Henrik Leopold, Jan Mendling, and Lluís Padró. Challenges and opportunities of applying natural language processing in business process management. In *COLING 2018: The 27th International Conference on Computational Linguistics: Proceedings of the Conference: August 20-26, 2018 Santa Fe, New Mexico, USA*, pages 2791–2801. Association for Computational Linguistics, 2018.

[166] Han van der Aa, Adrian Rebmann, and Henrik Leopold. Natural language-based detection of semantic execution anomalies in event logs. *Information Systems*, 102:101824, 2021.

[167] Wil Van Der Aalst and Kees Max Van Hee. *Workflow management: models, methods, and systems*. MIT press, 2004.

[168] W.M.P. van der Aalst and A.H.M. ter Hofstede. Yawl: yet another workflow language. *Information Systems*, 30(4):245–275, 2005.

[169] Boudewijn van Dongen. Bpi challenge 2020: Domestic declarations, 2020.

[170] Jussi Vanhatalo, Hagen Völzer, and Frank Leymann. Faster and more focused control-flow analysis for business process models through sese decomposition. In *Service-Oriented Computing–ICSOC 2007: Fifth International Conference, Vienna, Austria, September 17-20, 2007. Proceedings 5*, pages 43–55. Springer, 2007.

[171] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 30, 2017.

[172] Jesse Vig. A multiscale visualization of attention in the transformer model. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics: System Demonstrations*, pages 37–42, Florence, Italy, July 2019. Association for Computational Linguistics.

[173] Jan Vom Brocke, Theresa Schmiedel, Jan Recker, Peter Trkman, Willem Mertens, and Stijn Viaene. Ten principles of good business process management. *Business process management journal*, 20(4):530–548, 2014.

[174] Denny Vrandečić and Markus Krötzsch. Wikidata: a free collaborative knowledgebase. *Communications of the ACM*, 57(10):78–85, 2014.

[175] Alexandra Vultureanu-Albişi and Costin Bădică. Recommender systems: An explainable ai perspective. In *2021 International Conference on INnovations in Intelligent SysTems and Applications (INISTA)*, pages 1–6. IEEE, 2021.

[176] Haifeng Wang, Jiwei Li, Hua Wu, Eduard Hovy, and Yu Sun. Pre-trained language models and their applications. *Engineering*, 2022.

[177] Huaqing Wang, Lijie Wen, Li Lin, and Jianmin Wang. RLRecommender: A representation-learning-based recommendation method for business process modeling. In *ICSOC*, pages 478–486. Springer, 2018.

[178] Nan Wang, Shanwu Sun, and Dantong OuYang. Business process modeling abstraction based on semi-supervised clustering analysis. *Business & Information Systems Engineering*, 60(6):525–542, 2018.

[179] Qiang Wang, Bei Li, Tong Xiao, Jingbo Zhu, Changliang Li, Derek F Wong, and Lidia S Chao. Learning deep transformer models for machine translation. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 1810–1822, 2019.

[180] Quan Wang, Zhendong Mao, Bin Wang, and Li Guo. Knowledge graph embedding: A survey of approaches and applications. *IEEE Transactions on Knowledge and Data Engineering*, 29(12):2724–2743, 2017.

[181] Zhen Wang, Jianwen Zhang, Jianlin Feng, and Zheng Chen. Knowledge graph embedding by translating on hyperplanes. In *Proceedings of the AAAI conference on artificial intelligence*, volume 28, 2014.

[182] Mayur Wankhade, Annavarapu Chandra Sekhara Rao, and Chaitanya Kulkarni. A survey on sentiment analysis methods, applications, and challenges. *Artificial Intelligence Review*, 55(7):5731–5780, 2022.

[183] Mathias Weske. *Business Process Management: Concepts, Languages, Architectures*. Springer, 2012.

[184] Mathias Weske, Gero Decker, Marlon Dumas, Marcello La Rosa, Jan Mendling, and Hajo A. Reijers. Model Collection of the Business Process Management Academic Initiative, 2020.

[185] Alfred North Whitehead and Bertrand Russell. *Principia mathematica to* 56*, volume 2. Cambridge University Press, 1997.

[186] Karol Wieloch, Agata Filipowska, and Monika Kaczmarek. Autocompletion for business process modelling. In *Business Information Systems Workshops: BIS 2011 International Workshops and BPSC International Conference, Poznań, Poland, June 15-17, 2011. Revised Papers 14*, pages 30–40. Springer, 2011.

[187] Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Rémi Louf, Morgan Funtowicz, and Jamie Brew. Huggingface's transformers: State-of-the-art natural language processing. *CoRR*, abs/1910.03771, 2019.

[188] Bishan Yang, Wen tau Yih, Xiaodong He, Jianfeng Gao, and Li Deng. Embedding entities and relations for learning and inference in knowledge bases. In *ICLR (Poster)*, 2015.

[189] Liang Yao, Chengsheng Mao, and Yuan Luo. KG-BERT: BERT for knowledge graph completion. *CoRR*, abs/1909.03193, 2019.

[190] Yongfeng Zhang, Xu Chen, et al. Explainable recommendation: A survey and new perspectives. *Foundations and Trends® in Information Retrieval*, 14(1):1–101, 2020.

Appendix A

SAP-SAM Details

The following mappings from element types to element type groups have been used for the statistics in Table 4.1:

Table A.1: Element types and their groups

Element type	Element type group	Element type	Element type group
Task	Activities	Intermediate message event catching	Events
Collapsed subprocess	Activities	Intermediate timer event	Events
Subprocess	Activities	Intermediate escalation event	Events
Collapsed event subprocess	Activities	Intermediate conditional event	Events
Event subprocess	Activities	Intermediate link event catching	Events
Exclusive databased gateway	Gateways	Intermediate error event	Events
Event-based gateway	Gateways	Intermediate cancel event	Events
Parallel gateway	Gateways	Intermediate compensation event catching	Events
Inclusive gateway	Gateways	Intermediate compensation event catching	Events
Complex gateway	Gateways	Intermediate signal event catching	Events
Pool	Swimlanes	Intermediate multiple event catching	Events
Collapsed pool	Swimlanes	Intermediate parallel multiple event catching	Events
Lane	Swimlanes	Intermediate event	Events
Vertical pool	Swimlanes	Intermediate message event throwing	Events
Collapsed vertical pool	Swimlanes	Intermediate escalation event throwing	Events
Vertical lane	Swimlanes	Intermediate link event throwing	Events
Process participant	Swimlanes	Intermediate compensation event throwing	Events
Group	Artifacts	Intermediate signal event throwing	Events
Text annotation	Artifacts	Intermediate multiple event throwing	Events
IT system	Artifacts	End none event	Events
Data object	Data Elements	End message event	Events
Data store	Data Elements	End escalation event	Events
Message	Data Elements	End error event	Events
Start none event	Events	End cancel event	Events
Start message event	Events	End compensation event	Events
Start timer event	Events	End signal event	Events
Start escalation event	Events	End multiple event	Events
Start conditional event	Events	End terminate event	Events
Start error event	Events	Sequence flow	Connecting objects
Start compensation event	Events	Association undirected	Connecting objects
Start signal event	Events	Association unidirectional	Connecting objects
Start multiple event	Events	Association bidirectional	Connecting objects
Start parallel multiple event	Events	Message flow	Connecting objects

The following list contains all names of the example processes provided by the SAP-SAI system:

- Lieferung-zu-Bezahlung
- Bestellung-zu-Lieferung
- BANF-zu-Bestellung
- Wertschöpfungskette: Beschaffung
- Ebene 1 - Prozesslandkarte ACME AG
- Ebene 2 - Prozessbereich: Auftragsdurchführung
- Ebene 2 - Prozessbereich: Produktentwicklung
- Ebene 2 - Prozessbereich: Personalwesen
- Teile beschaffen
- Wareneingang
- Menge und Qualität überprüfen
- Arbeitsmittel beschaffen
- Bewerbungseingang
- Bewerber prüfen
- Mitarbeiter Onboarding
- Purchase Order-to-Delivery
- Delivery-to-Payment
- Purchase Requisition-to-Purchase Order
- Value chain: Procurement
- Level 1 - Value Chain ACME AG
- Level 2 - Process Area: Product Development
- Level 2 - Process Area: Order Processing
- Level 2 - Process Area: Human Resources
- Procure parts
- Check quantity and quality
- Receipt of Goods
- Receipt of Application
- Verify applicant
- Employee Onboarding
- Procurement of Work Equipment
- Niveau 1 - Chaine de valeur d'ACME AG
- Niveau 2 - Processus des Ressources Humaines
- Niveau 2 - Processus de développement produit
- Niveau 2 - Processus de gestion des commandes
- Donner l'équipement de travail
- Vérifier le candidat
- Installation d'un employé
- Réception d'une candidature
- Contrôler la quantité et la qualité
- Commande de pièces
- Réception de biens