Exploiting Label Semantics for Rule-based Activity Recommendation in Business Process Modeling

Diana Sola[a,b,*], Han van der Aa[b], Christian Meilicke[b], Heiner Stuckenschmidt[b]

[a]*SAP SE, Dietmar-Hopp-Allee 16, 69190 Walldorf, Germany*
[b]*Data and Web Science Group, University of Mannheim, B6 26, 68159 Mannheim, Germany*

Abstract

Business process modeling is a crucial task in organizations. Yet, the creation of consistent and complete process models is challenging and necessitates the support of process modelers with their task. In previous work, we presented a rule-based activity-recommendation approach, which recommends appropriate labels for a new activity inserted by a modeler in a process model under development. While our method has shown to work well, it is limited by the fact that it only learns rules that describe the inter-relations between complete activity labels. In the case that the model's activities and the ones in the training repository are disjoint, the existing approach will thus not be able to provide any recommendations. In this paper, we overcome this restriction by additionally considering the natural language-based semantics of the process models. In particular, we propose a semantics-aware recommendation approach that extends the existing approach in both central phases, i.e., in the rule-learning phase and in the rule-application phase. We equip the rule learning with novel rule types, which capture action and business-object patterns in process models. For the rule application, we developed an optional similarity extension that allows rules to make recommendations even if the bodies of the rules are not exactly true for the given model. Through an evaluation on a large set of real-world process models, we demonstrate that the semantic extensions can improve the quality of recommendations.

Keywords: Process modeling, activity recommendation, rule learning, semantic analysis

1. Introduction

Business processes structure the operations of organizations. They consist of sets of activities that jointly lead to an outcome that is of value to an organization or its clients [1]. These processes are found in a wide variety of contexts, ranging from classical procure-to-pay and order-to-cash processes to treatment

*Corresponding author

Email addresses: diana@informatik.uni-mannheim.de (Diana Sola), han@informatik.uni-mannheim.de (Han van der Aa), christian@informatik.uni-mannheim.de (Christian Meilicke), heiner@informatik.uni-mannheim.de (Heiner Stuckenschmidt)

processes in healthcare organizations. *Process models* capture information on such processes in a graphical manner, e.g., in the form of BPMN (Business Process Model and Notation) models, EPCs (Event-driven Process Chains), or Petri nets. These models are widely used as a basis for the documentation, analysis, and improvement of business processes. Therefore, business process modeling has become an integral part of process management initiatives in organizations [2].

Yet, despite their relevance, creating process models is a time-consuming and error-prone task, which requires substantial expertise [3, 4]. These issues are amplified in large scale modeling projects, where the creation of models often relies on modelers with limited experience [5, 6], whereas the distributed nature of such settings makes it hard to ensure consistency and clarity in the established models [7]. Overall, these modeling difficulties and pitfalls can result in situations in which downstream analysis and managerial tasks are conducted based on incorrect, incomplete, or inconsistent models [8, 9].

These issues motivate the need for proper support for the process modeling task. Such support can manifest itself in the form of recommendations to modelers on how to expand a process model they are working on [10]. An established instantiation of this is referred to as *activity recommendation* [11–13], which involves the context-aware recommendation of suitable labels for a new activity placed by a modeler in a model under development. For example, an activity-recommendation system may suggest that a *send invoice* activity could be followed by an activity labeled *receive payment*.

To provide context-aware activity recommendations during process modeling, we previously proposed a *rule-based recommendation approach* [14], which learns logical rules that describe how activities are used in a given repository of available process models. These rules are subsequently used to recommend appropriate labels for a new activity in a process model under development. For instance, a rule could express how likely it is that a model that contains a *create purchase order* activity also contains *check purchase order*. Although this rule-based approach outperforms both standard machine learning [15, 16] and embedding-based [11] techniques, it is limited by the fact that it only learns rules for completely equivalent activity labels. Due to this rigidity, the existing approach is unable to generalize the information contained within activity labels, such as the probability that the actions *create* and *check* are used in the same process or that *send invoice* and *send bill* are highly similar to each other, so that they likely should be subject to the same rules. This lack of generalization may lead to sub-par recommendations in general, whereas for cases where a model's activities are disjoint from the ones in the training repository, the existing approach will not be able to provide any recommendations at all.

In this paper, we overcome these limitations by additionally considering the natural language-based semantics of the process models. In particular, we propose a *semantics-aware recommendation approach* that extends the existing approach [14] with 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. Besides the improvement of the rule-learning phase with new rule types, we also introduce a novel extension of the rule application that considers semantic similarity of actions and business objects. As such, our extensions aim to improve the quality of the provided recommendations, as well as the ability to provide recommendations for unseen activity labels. We demonstrate these points in an experimental evaluation using a large repository of publicly available process models. The results show that the semantic extensions indeed improve the existing approach. In addition, we investigate the impact of the different rule types in an ablation study, which reveals that especially structural and action patterns are useful for the recommendation of activities.

The remainder of the paper is organized as follows. Section 2 illustrates the advantages of considering natural language-based semantics for activity recommendation, before providing a formal definition of the activity-recommendation problem in Section 3. Section 4 presents our rule-based approach with the semantic-based extensions, followed by a discussion of an experimental evaluation in Section 5. Section 6 reflects on related work, before concluding in Section 7.

2. Motivation

In this section, we motivate the use of rules for activity recommendation, as well as the semantic extensions we propose in this paper. As a basis for this motivation, we use the exemplary process model under development depicted in Figure 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.

In our work, we use a repository of process models as a basis to learn recommendation rules from. So far, our existing work [14] learns rules that capture activity inter-relations in the given repository. In this

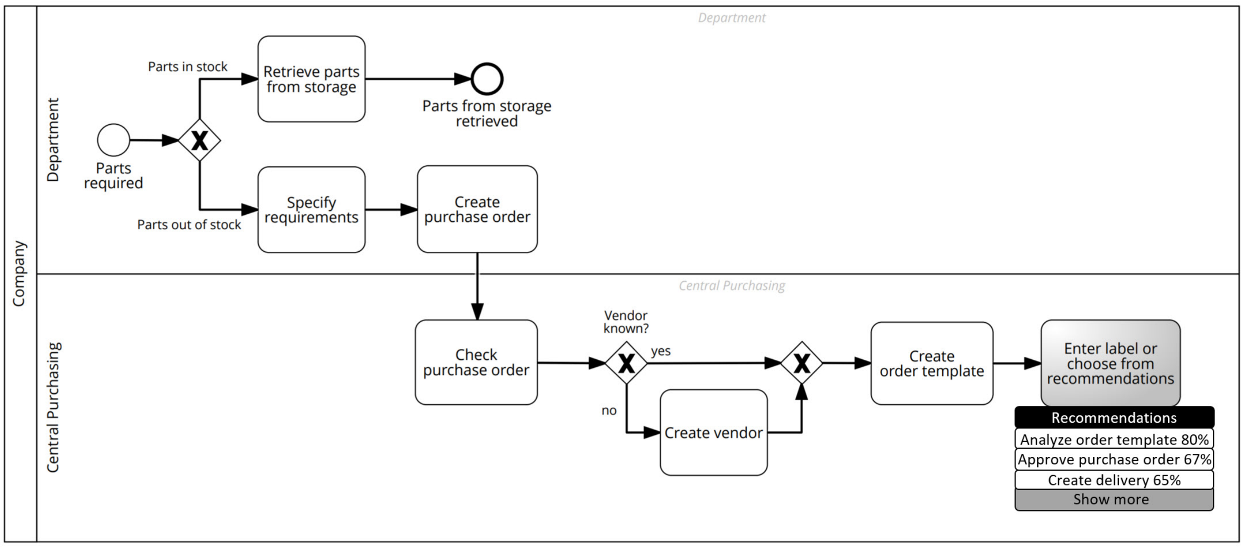

Figure 1: A process model under development

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 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-based 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 ** is followed by *analyze **', where * 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 1, given its preceding *create order template* activity. Similarly, a different action pattern might result in the depicted *approve purchase order* label, given the inclusion of the *create purchase order* activity in the model, 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 ** order template* and ** delivery*, where * can be replaced by any specific action. Given this pattern, we can then recommend a *create delivery* label for the task in Figure 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 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. Before we get to the details of our approach, the next section deals with the formalization of the underlying problem.

3. Problem Definition

In this section, we first provide essential definitions on the employed formalization of process models (Section 3.1), followed by the specification of the addressed activity-recommendation problem (Section 3.2).

3.1. Preliminaries

Our work is independent of any specific modeling notation that can be used to capture business processes, such as Petri nets or BPMN. Therefore, we employ the notion of *business process graphs* as a generic representation of process models to capture the behavioral relations that our recommendation approach uses as a basis.

Business process graphs. We use the following definition of business process graphs, adapted from Dijkman et al. [17], to represent a process model as a directed attributed graph:

Definition 1 (Business process graph). *Let $\mathcal{L}$ be the universe of all activity labels and $\mathcal{R}$ be a set of behavioral relation types. Let $\mathcal{P}(\mathcal{R})$ denote the power set of $\mathcal{R}$. A business process graph is a tuple (N, E, λ, τ), where N is a set of nodes, $E \subseteq N \times N$ is a set of directed edges, $\lambda : N \to \mathcal{L}$ is a function that maps a node to a label, and $\tau : E \to \mathcal{P}(\mathcal{R})$ is a function that maps an edge to a set of relation types.*

Note that, following this definition, a label can occur in multiple models, while the underlying nodes themselves always belong to exactly one model.

Behavioral abstraction using business process graphs. A business process model can be converted to a business process graph by using an abstraction procedure, where the model's content is mapped to the graph representation. 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 (e.g., *directly follows*). In the following, we use a transformation from Petri nets to business process graphs 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 notations. For instance, BPMN models can first be translated into Petri nets [18] before applying the abstraction approach.

Given a Petri net, we consider transitions, which correspond to activities, as nodes in a business process graph and omit its places. Then, for any pair of nodes m and n, we have to decide if we create a directed edge $e = (m, n) \in E$ and which relation types from a set $\mathcal{R}$ to assign to this directed edge. For this procedure, we follow Wang et al. [11], who propose three abstraction strategies, based on different sets of behavioral relations. We refer to [11] 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 $\tau(e) = \{\textit{followedBy}\}$. 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., $\tau(e) = \{\textit{alwaysCausal,inverseSometimesCausal}\}$.
- **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 instance, Figure 2 shows the business process graph obtained for the running example of Figure 1, based on $\mathcal{R}^{causal+concurrent}$. Although the business process graph abstracts from some details, the overall structure and sequence of activities is preserved with respect to the original model. Note that a $\mathcal{R}^{causal}$-graph can be obtained by omitting all dashed edges from Figure 2, while a $\mathcal{R}^{followedBy}$-graph corresponds to the $\mathcal{R}^{causal}$-graph in which all relation types are replaced by *followedBy*.

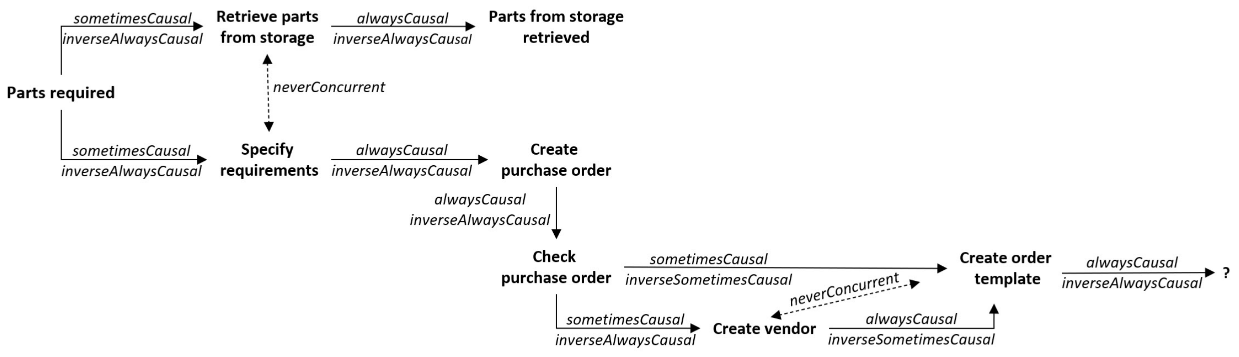

Figure 2: A business process graph corresponding to the process model in Figure 1 using the relation types $\mathcal{R}^{causal+concurrent}$

3.2. The activity-recommendation problem

Given a process model under development, captured as a business process graph B, the activity-recommendation problem is concerned with recommending a suitable label for a newly added activity $\hat{n}$:

Definition 2 (Activity-recommendation problem). *Let $\mathcal{B}$ be a process model repository, i.e., set of business process graphs, and let $B = (N, E, \lambda, \tau)$ be a business process graph under development with one unlabeled node $\hat{n}$, i.e., $\lambda(n)$ is given for all $n \in N \setminus \{\hat{n}\}$. Further, let $\mathcal{L}$ denote the universe of all activity labels. Then, the activity-recommendation problem is to find a suitable label $\lambda(\hat{n}) \in \mathcal{L}$ for $\hat{n}$.*

In contrast to our previous work [14], the choice of a suitable label is not limited to the activity labels used in the given repository $\mathcal{B}$ and can, rather, stem from any label in $\mathcal{L}$. Hence the consistent use of labels across the process models is now no longer anchored in the problem and has to be part of the solution instead. With this definition, the problem opens up and becomes more general.

4. Activity-recommendation Approach

As shown in Figure 3, our rule-based activity-recommendation approach consists of the two main phases: rule learning and rule application. The *rule-learning phase* derives pattern-based inter-relations about activity labels, actions, and business objects from the process models in a provided repository, capturing them in the form of logical rules. These rules are then used in the *rule-application phase* to recommend suitable labels for a new activity in a process model under development. 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 recommendations at each modeling step.

4.1. Rule Learning

In the rule-learning phase, we generate logical rules that capture regularities in the use of labels within a given repository of business process graphs $\mathcal{B}$. Our approach employs a set of rule patterns, i.e., templates,

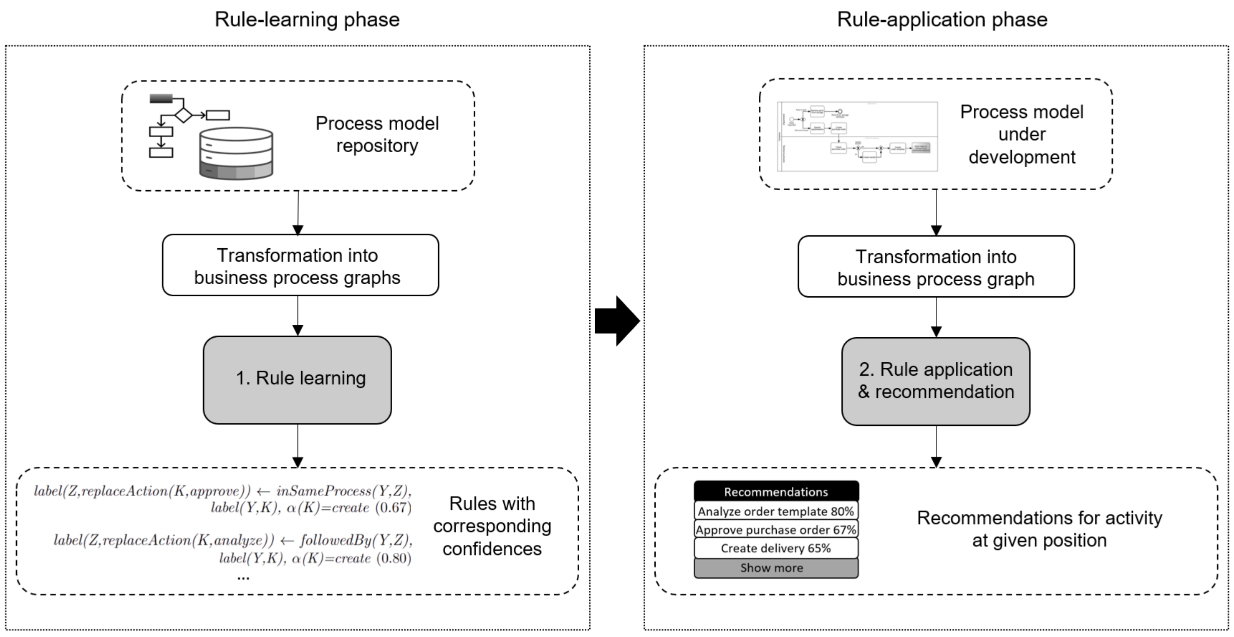
Rule-learning phase
Rule-application phase
Process model repository
Transformation into business process graphs
1. Rule learning
label(Z,replaceAction(K,approve)) ← inSameProcess(Y,Z), label(Y,K), α(K)=create (0.67)
label(Z,replaceAction(K,analyze)) ← followedBy(Y,Z), label(Y,K), α(K)=create (0.80)
...
Rules with corresponding confidences
Process model under development
Transformation into business process graph
2. Rule application & recommendation
Recommendations
Analyze order template 80%
Approve purchase order 67%
Create delivery 65%
Show more
Recommendations for activity at given position

Figure 3: Overview of the rule-based activity-recommendation approach

from which we derive a set of rules that hold in 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.

Our approach includes both rigid and semantic rule templates. The *rigid rule templates*, covered in Section 4.1.1, describe activity inter-relations in terms of complete activity labels. These rigid rules correspond to the once we employed in our earlier work [14]. They are complemented by novel *semantic rule templates*, described in Section 4.1.2, which capture regularities in the use of specific actions and business objects throughout a model repository. After introducing both kinds of templates, Section 4.1.3 describes how instantiations of these templates are learned from the model repository $\mathcal{B}$.

4.1.1. Rigid Rule Templates

This section describes the rigid rule templates, i.e., templates that capture regularities between entire activity labels, which our approach currently covers.

Rule predicates. For the definition of the rule templates, we first need to describe the given business process graphs in terms of logical formulas. For this, we translate each business process graph $B = (N, E, \lambda, \tau) \in \mathcal{B}$ as follows:

- For each node $n \in N$ we add a formula *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 $r \in \tau(e)$ we add a formula $r(m, n)$ 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 formulas *inSameProcess*(m, n) and *inSameProcess*(n, m) to express that m and n appear in the same business process graph.

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 graphs in the repository $\mathcal{B}$.

Templates and instantiations. When defining our templates, we use h, j, k and $l \in \mathcal{L}$ 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 $\mathcal{L}$. The variables W, X, Y and Z in the templates stand for concrete activity nodes. A rule is *grounded* if it is an instantiation of a rule template, where all variables are replaced by concrete values.

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) $\leftarrow$ *followedBy(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.

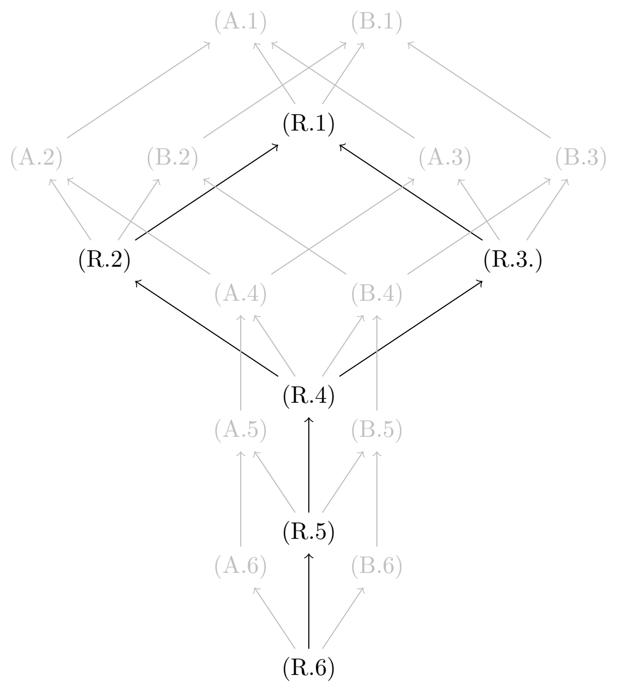

Figure 4: Template specificity lattice

Rule specificity. Certain rules inherently relate to each other. For instance, any grounding $Y = a_1$ and $Z = a_2$ 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. The black-colored part of Figure 4 shows a complete specificity lattice of the R templates (for now we ignore the grey-coloured parts of the figure). Most of the arcs in the specificity lattice can be explained by the simple rule shown above or the fact 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).*

4.1.2. Semantic Rule Templates

In addition to activity inter-relations, we want to consider natural language-based semantics in the form of action and business-object patterns. Corresponding to established work on the semantic analysis of activity labels in process models [19], 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 graphs $\mathcal{B}$, 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., [20, 21]. 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 predicates. For the semantic rule templates, we add the following logical formulas to the translation of a business process graph $B = (N, E, \lambda, \tau) \in \mathcal{B}$ in addition to the ones previously specified in Section 4.1.1, 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., β(*create order template*)=*order template* or β(*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.

Templates and instantiations. We define four additional template sets, which adapt the R templates defined in Section 4.1.1 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 $label(Z,replaceAction(K,a')) \leftarrow inSameProcess(Y,Z),\ label(Y,K),\ \alpha(K)=a$

A.2 $label(Z,replaceAction(K,a')) \leftarrow followedBy(Y,Z),\ label(Y,K),\ \alpha(K)=a$

A.3 $label(Z,replaceAction(K,a')) \leftarrow inSameProcess(X,Y),\ inSameProcess(Y,Z),\ label(X,j),\ label(Y,K),\ \alpha(K)=a$

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

A.5 $label(Z,replaceAction(K,a')) \leftarrow followedBy(X,Y),\ followedBy(Y,Z),\ label(X,j),\ label(Y,K),\ \alpha(K)=a$

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

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 exemplary 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:

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

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:

$$\textit{label(Z,replaceActionAndFlip(K,check))} \leftarrow \textit{followedBy(Y,Z), label(Y,K), } \alpha\textit{(K)=create}$$

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 *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 *replace-BusinessObject* 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 4.1.1 for the rigid rule templates.

Template specificity. Turning back to the specificity lattice in Figure 4, we see how 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 4 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 a 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 be 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 4, since the specificity of these templates is equivalent to their non-flipped A and B counterparts.

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. 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.

4.1.3. 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 labels, actions, or business objects from the repository. 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, actions and business objects 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.

Rule instantiation. 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*, α or β 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. For the semantic rule templates, which involve the functions *replaceAction*, *replaceActionAndFlip*, *replaceBusinessObject* or *replaceBusinessObjectAndFlip*, we additionally limit the instantiations of the rule templates to those activities Y and Z that have separable labels (cf., Section 4.1.2) 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.

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. [22], which states that the *support* of a horn rule *head* $\leftarrow$ *body* shall be computed by counting all 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 the rule can be understood as the probability that the rule makes a correct prediction within the given repository of business process graphs $\mathcal{B}$. For instance, the following two rules are related to activities that are in the same process with a *parts required* activity:

$r_1 =$ *label(Z,complete purchase order)* $\leftarrow$ *inSameProcess(Y,Z), label(Y,parts required)*

$r_2 =$ *label(Z,check purchase order)* $\leftarrow$ *inSameProcess(Y,Z), label(Y,parts required)*

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 *complete 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 $\textit{confidence}(r_2) = 5/15 = 0.333$.

Default rules. Finally, to ensure that our approach is always able to provide a sufficient number of recommendations, we also learn ten default rules [23], 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.

4.2. Rule Application

Given an unfinished business process graph B with its unlabeled node $\hat{n}$, we use the rules learned from $\mathcal{B}$, as described in Section 4.1, and apply them on $\hat{n}$, while taking the current state of the process graph B into account. 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 one of the recommendations for the model under development depicted in Figure 1, where $\hat{n}$ is the rightmost node.

$$\textit{label}(\hat{n},\textit{approve purchase order}) \leftarrow \textit{inSameProcess}(X,Y),\ \textit{alwaysCausal}(Y,\hat{n}), \qquad (*)$$
$$\textit{label}(X,\textit{ check purchase order}),\ \textit{label}(Y,\textit{ create order template})$$

If we compare this rule to Figure 1, 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.

Recommendations. Once the body of a rule is true, an activity recommendation is given by its head. The recommendation of a default rule or a rule that instantiates the rigid rule templates R.1-R.6 is directly given by the second argument of the *label*-predicate. Rule $(*)$, for example, recommends *approve purchase order* as the label for $\hat{n}$. To derive the recommendation of a rule that instantiates one of the semantic rule templates, we have to take an additional step and evaluate the function in the second argument of the head's *label* predicate. As an example, consider the rule given by $(2*)$, which instantiates rule template A.2:

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

Comparing this rule to the process model in Figure 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*:

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

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 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 unfinished 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 confidence 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 high-dimensional vector representations of words, so-called *embeddings*, which capture semantic information of words so that similar ones are closer in the vector space. Word embeddings can be generated using algorithms like word2vec [24]. 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. During the rule-application phase, we gather the recommendations of all rules where the body is true with respect to the unfinished model B, as well as those stemming from the semantically similar rules, 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. With the *max*-aggregation method, we assign the maximum confidence of the applicable rules to the recommendation, while 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 [25]. After applying an aggregation method, we obtain a set of recommendations for the recommendation task at hand, each with its own confidence score.

Recommendation transparency. One of the advantages of our approach is that the rules that serve as a basis for recommendations allow to better understand them. For example, 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: Since the previous activity is *create order template* and the process also includes a *check purchase order* activity, there is a rather strong indication (confidence score of 0.67) that the activity should be labeled *approve purchase order*. The explanation for the top-ranked recommendation derived from rule (2$*$) could be: In most cases (confidence score of 0.80) the action *create* is followed by the action *analyze*. Such an explanation might raise the confidence of the user in the given recommendation and might make it easier for her 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 have a look at similar processes, which might further help her 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 blackbox, for which it is harder to pinpoint and omit specific relations that were learned.

5. Experimental Evaluation

In this section, we present an extensive experimental study that we conducted to evaluate our semantics-aware activity-recommendation approach. Before we get to the experiments, we introduce the used dataset (Section 5.1) and the evaluation setup (Section 5.2). In the first part of the experiments (Section 5.3), we compare our method to other activity-recommendation approaches in two scenarios with different evaluation procedures. The second part (Section 5.4) represents an ablation study, in which we investigate the impact of the individual rule templates, as well as the added value of the semantic-aware patterns. Finally, we present a study in which we examine the extension with similarity-based recommendations (Section 5.5). We conclude the experimental evaluation with a discussion on the limitations of our approach (Section 5.6).

5.1. Dataset

To conduct our evaluation, we used models from the Business Process Model Academic Initiative (BPMAI) [26] collection. From this collection, we selected all Petri nets and BPMN 2.0 models that are in English and contain between 3 and 50 activities. Given that the BPMAI collection contains multiple versions (i.e., revisions) of 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 these 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 can contain 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.

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 5.2 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.

5.2. Evaluation Setup

Implementation and environment. We implemented our activity-recommendation approach as a Python prototype.[1] To operationalize the semantic rule templates, the implementation uses the label-parsing approach by Rebmann and Van der Aa [21] 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 [27]. 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 4.1.2, 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 as 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-d vectors generated from a large corpus of written text and is thus fully sufficient for our use.

All experiments are conducted on an Intel® Xeon® E5-2623 v3@16x3.00 GHz CPU computer with 256G RAM.

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. Aside from evaluating our work on two different datasets (with and without model revisions), we also assess the accuracy in various modeling situations, as is common practice for

[1]For proprietary reasons, requests for the source code of the implementation should be submitted to diana.sola@sap.com.

activity-recommendation approaches [16]. Therefore, we use three evaluation procedures, reflecting varying states of process models under development, resulting in a variety of recommendation tasks.

The evaluation procedures work in two steps. First, given a business process graph 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, we alter the state of the process model under development by removing some of the other nodes and their associated edges from the graph, according to one of the evaluation procedures. The remaining graph 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.

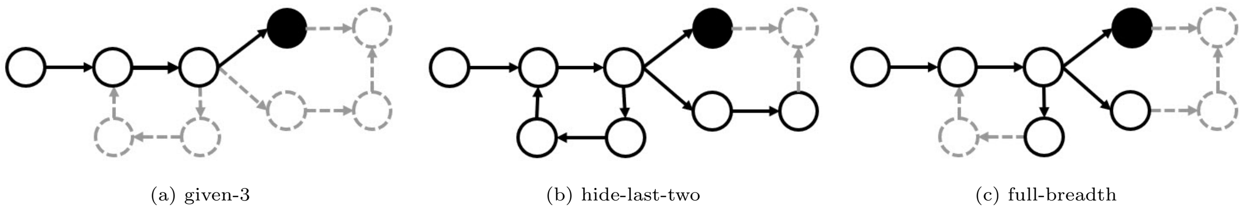

Figure 5: Illustration of the different evaluation procedures

- **given-3.** In the given-3 procedure, we pick a path of length 4 which is a longest path from a source node (node with no incoming edges) to the activity at position 4 and aim to predict the label of this activity. The given-3 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.
- **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.

Overall, as also seen in Figure 5, the given-3 procedure thus provides the least information as a basis for recommendation, whereas hide-last-two maintains the given process graph almost completely. Finally, the amount of information given by the full-breadth procedure usually lies in between them.

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

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.

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 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. 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 rank 9 and 10 (0.01). Higher positions are therefore weighted more heavily by the MRR.

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 recommendation extension is used in the last part of the experiments in Section 5.5. Note that we assess the added value of the semantic rule templates in Section 5.4 as part of an ablation study, whereas the similarity-based extension is assessed in Section 5.5. 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 4.2 introduced *max*- and *noisy-or*-aggregation with each of the in Section 3 presented abstraction strategies, i.e., *followedBy, causal* and *causal+concurrent.* This leads to six different configurations, which we denote, for instance, by RULES $followedBy^{max}$. 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 $\mathcal{R}^{followedBy}$ setting for the given-3 scenario.

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

- MOSTFREQ [15]: This method always recommends the ten activities most frequently used in the available process repository.

- CoOccur [15]: This technique is based on the conditional probabilities of the simultaneous occurrence of pairs of activities 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 [15]: 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.

The following contextualized methods (Ctx) consider the longest path to the unlabeled activity in the process model under development as the current modeling context for the recommendation.

- CoOccur-Ctx [16]: 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 [16]: kNN-Ctx is a contextualized version of kNN. Compared to kNN, kNN-Ctx increases the weight of the neighbour processes that contain activities, which are also included in the current modeling context of the process under development.
- Link-Ctx [16]: 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 [16]: 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 [16]: The Hybrid-Ctx technique combines the contextualized kNN strategy, kNN-Ctx, 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.

While the first three methods, MostFreq, CoOccur and kNN, can be understood as simple baselines, the other five are more sophisticated techniques that have been specifically designed to perform well in highly similar activity-recommendation scenarios.

Note that the above methods can be appropriately applied on the business process graph representations of the process models we also use as input for our approach, since they only consider which pairs of activities occur in the same process model and which directly follow each other. The graphs can losslessly capture this information in the *inSameProcess* and *followedBy* relations. However, the other methods are unable to incorporate the causal and concurrency relations that we additionally consider in our rule-based approach, which is why we only apply them in the $\mathcal{R}^{followedBy}$ setting.

5.3. Evaluation Results

Table 1 shows the experimental results of the evaluated approaches and configurations on the whole dataset, i.e., in an evaluation situation where multiple similar 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.

	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
*k*NN	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
*k*NN-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 $followedBy^{max}$	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 $causal^{max}$	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 $causal{+}conc.^{max}$	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 1: Experimental results for the different approaches on the whole dataset. Best results per procedure are in bold.

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 baselines. As expected, the results of the simple baselines MostFreq, CoOccur and *k*NN are comparatively low. However, the *k*NN 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 *k*NN-Ctx work better than their non-contextualized counterparts. This shows that the current modeling context as considered by Jannach et al. [16] can be of importance, i.e., it can be useful to give greater consideration to certain parts of the process model under development. 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 proves 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 [16]: 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 *max*-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 to graphs, 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 $causal^{max}$ is slightly worse than the one of RULES $followedBy^{max}$ 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 graph, 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 graph and node o is removed for the partial graph, 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 used 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 baseline 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.

	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*^{max}	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*^{max}	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.*^{max}	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 2: Experimental results for the different approaches on the last-revision dataset. Best results per procedure are in bold.

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 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.

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≤10. Figure 6 shows these hit rates of our rule-based approach in the $\mathcal{R}^{causal+concurrent}$ setting with *max* aggregation when evaluated in the full-breadth procedure on the whole (left) and on the last-revision (right) dataset. Note that the

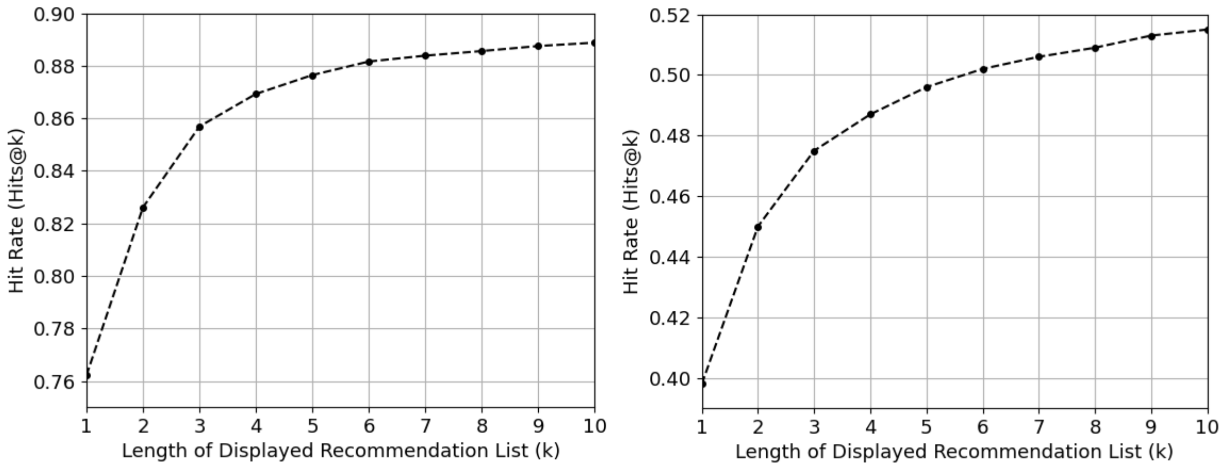

Figure 6: Different hits rates for Rules *causal+conc.*^{max} in the full-breadth evaluation procedure on the whole (left) and on the last-revision (right) dataset

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. If our approach would generate the ranking randomly, the curves would be a rather straight line from the Hits@1 value to the Hits@10 value. Instead, both curves in Figure 6 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 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 shows the length of the recommendation list (left) and the recall (right) depending on the chosen confidence score threshold when we use our approach in the $\mathcal{R}^{causal+concurrent}$ setting with *max* 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 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 that we investigated previously, the length of the recommendation list depends on the confidence score for the recall.

The left curve in Figure 7, 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 right graph in Figure 7 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

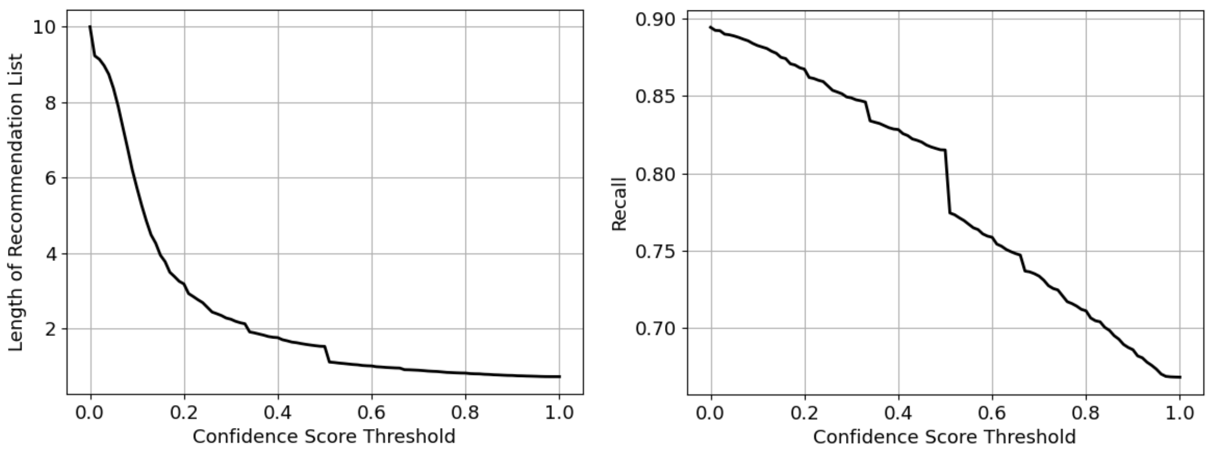

Figure 7: Length of recommendation list (left) and recall (right) for different confidence score thresholds when using Rules *causal+conc.*^{max} in the full-breadth evaluation procedure on the whole dataset

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 recall 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. The hide-last-two scenario using max-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.

5.4. 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 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 *max* aggregation for all procedures and the last-revision dataset.

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* predicate and the associated consideration of structural patterns improves the results significantly. In the next step, we add template

	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 3: Results of the ablation study with varying sets of rule templates

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* predicates 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* predicate 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.*^{max} are better than the ones of Rules *causal*^{max} in Table 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 [14] 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 max-aggregation requiring 1.56s is an exception to this. These numbers are to be compared with the runtimes reported in Section 5.3 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.

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

Table 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. 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 *max* 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

	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 4: Results with and without similarity-based recommendations

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 4.2, does not achieve the desired result and needs to be improved in future work.

5.6. Limitations

Our rule-based approach is subject to some limitations. First, our approach 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.

A second limitation relates to the labeling style of the process models. While the rigid rule templates work independently from 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 models that can be leveraged for the recommendation approach. Rather, they offer the opportunity to leverage other patterns in the labels to make recommendations.

Another limitation is related to the extension with the similarity-based recommendations. In our experiments, this extension 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 can generate recommendations that are similar to the actually chosen activity and can thus be very useful for the modeler. Also, the Hits@100 results showed that the approach with similarity-based extension can recommend 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.

6. Related Work

Our work primarily relates to other approaches for activity recommendation in process models, as well as to research streams on activity label analysis and rule learning.

Activity recommendation. Various existing works provide activity recommendations while modeling a process. A common approach is the abstraction of business process models to directed graphs, followed by the use of graph mining techniques to extract structural patterns from a process repository. The similarity between the extracted patterns and a process model under development can then be calculated using different distance measures, including common subgraph distance [28] and edit distance [12, 13, 28]. However, graph mining methods reach their limits when applied to large datasets, e.g., containing thousands of process models. To overcome this, Wang et al. [11] developed an embedding-based activity recommendation method, called RLRecommender. While this method is able to handle large datasets and outperforms the graph mining methods in the conducted experiments, it only considers one preceding activity in the process model under development and its inter-relation with the unlabeled activity as a context for the recommendation. As we have shown in previous work [14, 29], the performance of RLRecommender is comparatively low for this reason.

The work by Jannach et al. [16], in contrast, focuses on considering the modeling context for the provision of recommendations. Originally developed to provide modeling support for users in the specific area of data analysis workflows, the recommendation techniques by Jannach et al. can also be used to recommend activities for business process models [14]. They present a variety of methods that are based on different concepts, i.e., k-nearest neighbors, co-occurrence and frequently linked elements. In a laboratory study, the developed recommendation tool is shown to help users increasing the efficiency of the modeling process. However, as shown through our experiments, our rule-based recommendation approach outperforms the various methods they propose for the task at hand.

In previous work [29], we presented different approaches to use embedding- and rule-based knowledge graph completion methods for activity recommendation. While we showed that standard knowledge graph completion methods can in principle be applied to the activity-recommendation problem, the experimental results revealed that they are not flexible enough to completely adapt to the problem.

Beyond the scope of activity recommendation, there are several other ways to support users with their process modeling task [30], for example, by suggesting process fragments rather than single activities or by detecting naming conflicts during the labeling of process elements. In contrast to methods that suggest a set of process elements or even a whole process model, activity recommendation systems can be used iteratively to assist the user modeling a business process in an interactive way.

Activity label analysis. Similarly to our approach, Smirnov et al. [31] introduce a formal framework for the identification of co-occurrence and behavioral action patterns in the activities of process models in a repository, based on association rule mining. While their works concentrates on action patterns, we combine them with patterns based on extracted business objects. The focus of their work thus results in the recommendations of actions rather than full activities. For the applicability of their action-pattern framework, the work also lacks a component that maps an activity label to an action. Existing techniques for the parsing of activity [32] and event labels [21] close this gap, by proposing methods that automatically extract components such as actions and business objects from labels, as used in our work. While here used to improve activity recommendations, recent work by Van der Aa et al. [33] also uses these patterns to detect behavioral anomalies in processes through common-sense analysis. This avenue may be explored in future work, to avoid recommending activity labels that do not make sense at a given position in a model.

Rule Learning. Activity recommendation is inherently based on the identification of patterns in a given model repository, which suggests the application of machine learning techniques. A process model can be seen as a specific type of labeled directed graphs. Since such a graph can be described via a set of first order formulas, symbolic methods such as association rule mining or multi-relational rule mining methods are applicable.

While association rule mining considers patters in terms of co-occurrence, multi-relational models can distinguish between different relations, which makes them ideally suited for describing the order of activities in addition to their co-occurrence. One of the early relational rule mining systems is WARMR [34], which can (in principle) learn the types of rules that we are interested in but does not scale to the large process repositories that we are working with. More recently, systems as AMIE [22] and AnyBURL [35] have been proposed to learn rules that describe the regularities in a given knowledge base. However, such systems have a restricted language bias tailored to knowledge base problems and are thus not able to adjust to rule types that are important in a process context. For example, we showed in [29] that AnyBURL performs on the activity-recommendation task clearly worse than the specialized approach we proposed in [14].

Rule mining can be understood as a field that is highly related to or overlaps the research field Inductive Logic Programming (ILP). Classic ILP techniques such as FOIL [36] and Tilde [37] are usually based on a covering approach. Instead of mining all possibly relevant rules, they focus on finding a small rule set that covers all examples in the training set once. However, in a prediction scenario we might often encounter situations where the prediction must be based on a rather weak rule that covers only few examples and would be redundant in the set of all rules. Thus, a rule mining approach that aims to detect all relevant rules fits better to the activity-recommendation task. Another difference is the need for explicitly given negative examples, which are not available in the scenario we address. For these reasons, we do not apply ILP systems to the activity-recommendation problem.

7. Conclusion

In this paper, we presented a rule-based approach to support process modelers with activity recommendations, which makes additional use of label 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 their semantic similarity. The semantic components in both the rule-learning and rule-application phases make our approach better applicable in cases where the process model under development contains unseen activity labels. Our extensive experiments showed that considering the natural language-based semantics of the process models is a meaningful addition and can improve the quality of the provided recommendations.

In future work, we would like to use methods from natural language processing to further improve our approach. In case that the process model under development and the available process models have small or even no overlap, where our rule-based approach might generate recommendations with rather low confidence scores, it could be useful to apply a pre-trained language model to generate recommendations. We could then benefit from the large vocabulary of these pre-trained models. Further improvements may be possible by incorporating information beyond activity labels and their inter-relations into the recommendation procedure, when available. For example, resources associated with activities could be used to establish rules that also consider which activities are typically performed by which kinds of actors. Similarly, process model annotations could be used to discern patterns related to ontological concepts or categories [38, 39], e.g., allowing a recommendation approach to learn what types of activities typically follow each other. Furthermore, we aim to improve the order of the provided recommendations, e.g., by finding other ways to aggregate the confidence scores of rules that lead to the same recommendation. This concerns in particular the integration of the recommendations stemming from our similarity-based extension. Moreover, we want to expand our approach such that it is able to also recommend larger process fragments, thus going beyond individual activity labels.

References

[1] M. Dumas, M. La Rosa, J. Mendling, H. A. Reijers, Fundamentals of Business Process Management, Springer, Berlin, 2013.

[2] I. Davies, P. Green, M. Rosemann, M. Indulska, S. Gallo, How do practitioners use conceptual modeling in practice?, Data & Knowledge Engineering 58 (3) (2006) 358–380.

[3] P. J. Frederiks, T. P. Van der Weide, Information modeling: The process and the required competencies of its participants, DKE 58 (1) (2006) 4–20.

[4] F. Friedrich, J. Mendling, F. Puhlmann, Process model generation from natural language text, in: CAiSE, Springer, 2011, pp. 482–496.

[5] M. Rosemann, Potential pitfalls of process modeling: part a, Business Process Management Journal.

[6] J. Mendling, Empirical studies in process model verification, in: Transactions on petri nets and other models of concurrency II, Springer, 2009, pp. 208–224.

[7] F. Pittke, H. Leopold, J. Mendling, Automatic detection and resolution of lexical ambiguity in process models, IEEE Transactions on Software Engineering 41 (6) (2015) 526–544.

[8] A. Abran, J. W. Moore, P. Bourque, R. Dupuis, L. Tripp, Software engineering body of knowledge, IEEE Computer Society, Angela Burgess.

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

[10] M. Fellmann, N. Zarvic, D. Metzger, A. Koschmider, Requirements catalog for business process modeling recommender systems, in: WI, 2015, pp. 393–407.

[11] H. Wang, L. Wen, L. Lin, J. Wang, RLRecommender: A representation-learning-based recommendation method for business process modeling, in: ICSOC, Springer, 2018, pp. 478–486.

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

[13] S. Deng, D. Wang, Y. Li, B. Cao, J. Yin, Z. Wu, M. Zhou, A recommendation system to facilitate business process modeling., IEEE Trans Cybern 47 (6) (2017) 1380–1394.

[14] D. Sola, C. Meilicke, H. Van der Aa, H. Stuckenschmidt, A rule-based recommendation approach for business process modeling, in: CAiSE, Springer, 2021.

[15] D. Jannach, S. Fischer, Recommendation-based modeling support for data mining processes, in: RecSys, 2014, pp. 337–340.

[16] D. Jannach, M. Jugovac, L. Lerche, Supporting the design of machine learning workflows with a recommendation system, ACM TiiS 6 (1) (2016) 1–35.

[17] R. M. Dijkman, M. Dumas, L. García-Bañuelos, Graph matching algorithms for business process model similarity search., in: BPM, Vol. 5701, Springer, 2009, pp. 48–63.

[18] R. M. Dijkman, M. Dumas, C. Ouyang, Semantics and analysis of business process models in BPMN, Inf. Softw. Technol. 50 (12) (2008) 1281–1294.

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

[20] H. Leopold, H. van der Aa, J. Offenberg, H. A. Reijers, Using hidden markov models for the accurate linguistic analysis of process model activity labels, Information Systems 83 (2019) 30–39.

[21] A. Rebmann, H. Van der Aa, Extracting semantic process information from the natural language in event logs, in: CAiSE, Springer, 2021.

[22] L. A. Galárraga, C. Teflioudi, K. Hose, F. Suchanek, AMIE: Association rule mining under incomplete evidence in ontological knowledge bases, in: WWW, 2013, pp. 413–422.

[23] J. Fürnkranz, D. Gamberger, N. Lavrac, Foundations of Rule Learning, Cognitive Technologies, Springer, 2012.

[24] T. Mikolov, K. Chen, G. Corrado, J. Dean, Efficient estimation of word representations in vector space, arXiv preprint arXiv:1301.3781.

[25] S. Russell, P. Norvig, Artificial Intelligence: A Modern Approach, 3rd Edition, Prentice Hall, 2010.

[26] Model collection of the Business Process Management Academic Initiative, http://bpmai.org/.

[27] M. Honnibal, I. Montani, S. Van Landeghem, A. Boyd, spaCy: Industrial-strength Natural Language Processing in Python (2020). doi:10.5281/zenodo.1212303.
URL https://doi.org/10.5281/zenodo.1212303

[28] B. Cao, J. Yin, S. Deng, D. Wang, Z. Wu, Graph-based workflow recommendation: on improving business process modeling., in: CIKM, ACM, 2012, pp. 1527–1531.

[29] D. Sola, C. Meilicke, H. Van der Aa, H. Stuckenschmidt, On the use of knowledge graph completion methods for activity recommendation in business process modeling, in: AI4BPM, Springer, 2021.

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

[31] S. Smirnov, M. Weidlich, J. Mendling, M. Weske, Action patterns in business process model repositories., Comput. Ind. 63 (2) (2012) 98–111.

[32] H. Leopold, S. Smirnov, J. Mendling, On the refactoring of activity labels in business process models, Information Systems 37 (5) (2012) 443–459.

[33] H. van der Aa, A. Rebmann, H. Leopold, Natural language-based detection of semantic execution anomalies in event logs, Information Systems 102.

[34] L. Dehaspe, L. De Raedt, Mining association rules in multiple relations, in: International Conference on ILP, Springer,

1997, pp. 125–132.

[35] C. Meilicke, M. W. Chekol, D. Ruffinelli, H. Stuckenschmidt, Anytime bottom-up rule learning for knowledge graph completion, in: IJCAI, AAAI Press, 2019, pp. 3137–3143.

[36] J. R. Quinlan, Learning logical definitions from relations, Machine learning 5 (3) (1990) 239–266.

[37] H. Blockeel, L. De Raedt, Top-down induction of first-order logical decision trees, Artificial intelligence 101 (1-2) (1998) 285–297.

[38] H. Leopold, C. Meilicke, M. Fellmann, F. Pittke, H. Stuckenschmidt, J. Mendling, Towards the automated annotation of process models, in: J. Zdravkovic, M. Kirikova, P. Johannesson (Eds.), Advanced Information Systems Engineering, Springer International Publishing, 2015.

[39] C. Di Francescomarino, P. Tonella, Supporting ontology-based semantic annotation of business processes with automated suggestions, International Journal of Information System Modeling and Design (IJISMD) 1 (2) (2010) 59–84.