A Rule-based Recommendation Approach for Business Process Modeling

Diana Sola[1,2], Christian Meilicke[2], Han van der Aa[2], Heiner Stuckenschmidt[2]

[1] Intelligent Robotic Process Automation, SAP SE, Walldorf, Germany
[2] Data and Web Science Group, University of Mannheim, Mannheim, Germany
{diana,christian,han,heiner}@informatik.uni-mannheim.de

Abstract. Business process modeling is a crucial, yet time-consuming and knowledge-intensive task. This is particularly the case when modeling a domain-specific process, which often requires the use of highly specialized terminology in a consistent manner. To alleviate these issues, the process modeling task can be supported by techniques that suggest how a model under development can be expanded. In this work, we provide such suggestions through a rule-based activity recommendation approach, which suggests suitable activities to be included at a user-defined position in a process model. A benefit of our rule-based work over other approaches is that it accompanies recommendations with explanations, providing additional transparency and trustworthiness to users. Furthermore, through comprehensive evaluation experiments on a large set of real-world process models, we show that our rule-based approach outperforms other methods, including an embedding-based one.

Keywords: Process modeling, activity recommendation, rule learning

1 Introduction

Business processes structure the operations of organizations. They consist of sets of activities which jointly lead to an outcome that is valuable to an organization or its clients. *Process models* have become the de facto standard to capture information on such processes and are, therefore, present in virtually all facets of the business process management lifecycle. For this reason, documenting business operations using process models has become a quintessential activity for many organizations [9].

Although they are clearly important instruments for the execution, analysis, and improvement of business processes, actually creating process models is a time-consuming task that requires substantial expertise [11, 12]. This task is even more challenging when modeling a domain-specific process, which often requires the use of specialized and technical vocabulary in a consistent manner. To mitigate these issues, modeling can be supported through the provision of recommendations that suggest modelers on how they may expand a process model that they are working on [10]. One clear manner in which such support can be provided is through activity recommendation.

Given a business process model under development, *activity recommendation* sets out to suggest suitable activities to extend the model at a user-defined position. A repository of available business process models can serve as a basis for this task. Here, it is crucial that recommendations are provided in a context-aware manner, i.e., they should take the current content and state of a process model into account. To illustrate this, consider Figure 1, which shows a process model under development where a user has just inserted an unlabeled activity on the model's right-hand side. The activity recommendation task is then to determine a suitable activity for this position, i.e., to find an appropriate label for it. Since the process that has been modeled so far contains activities that are commonly associated with *order-to-cash* processes, a context-aware recommender system will provide recommendations that correspond to activities that occur at a comparable position in similar processes found in a model repository, such as 'Submit purchase order', 'Analyze quotation', and 'Purchase order sent'.

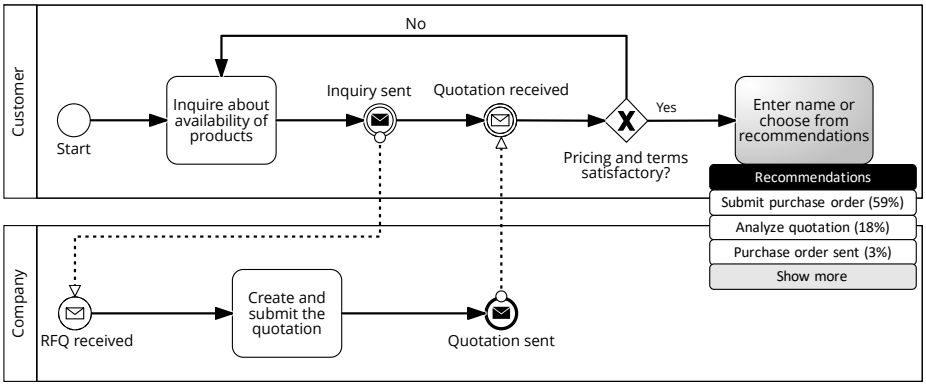

Fig. 1: A business process model under development

In this paper, we propose to tackle the activity recommendation task using a rule-based recommendation approach. Such approaches have their origin in the field of inductive logic programming [4] and have been shown to be competitive for use cases such as knowledge graph completion [17]. An additional benefit is that rule-based approaches offer an explanation for given recommendations, which helps to improve the transparency, trustworthiness, and satisfaction of recommendation systems [23]. Our proposed rule-based approach learns logical rules that describe how activities are used in a given repository of available process models, which we subsequently use to recommend appropriate activities at a given position. Our rule learner is based on the top-down search implemented in association rule mining systems [5,13], which we adapted to support a specific, process-oriented language designed for activity recommendation. An extensive experimental evaluation demonstrates that our rule-based approach outperforms both standard machine learning [14, 15] and embedding-based [22] techniques.

In the remainder, we first discuss various other works related to process model recommendation in Section 2, before formally defining the activity-recommen-

dation problem in Section 3. We present our approach for rule-based activity recommendation in Section 4 and discuss our extensive evaluation experiments in Section 5. Finally, we conclude the work in Section 6.

2 Related Work

There are several activity recommendation approaches that abstract business process models to directed graphs and use graph mining techniques to extract structural patterns from a process repository. The similarity between the extracted patterns and a process model under development can be calculated using different strategies, including common subgraph distance [3] and edit distance [3, 6, 16]. However, as stated by Wang et al. [22], graph mining methods reach their limits when applied to large complex datasets with thousands of processes. To overcome this, they present an embedding-based activity recommendation method, RLRecommender, which is able to handle large datasets and outperforms the graph-mining-based algorithms in the conducted experiments. Therefore, we include this method in our experimental studies.

Jannach et al. [14, 15] propose different recommendation techniques to provide modeling support for users in the specific area of data analysis workflows. The user support consists of recommending additional operations to insert into the machine learning workflow under development and is hence similar to activity recommendation. In this paper, we show that the recommendation strategies from Jannach et al. can also be used to recommend activities for more general process models than data analysis workflows, i.e., business process models, and evaluate them in our experiments.

Since activity recommendation is inherently based on the identification of patterns in a given process repository, the application of *association rule mining* techniques [1] also suggests itself for this purpose. Such techniques consider associations in terms of activity co-occurrence, but ignore the sequence of activities that can best be described with a multi-relational model. Contrary to association rule mining, relational rule mining can distinguish between different relations. One of the early systems is WARMR [5]. More recently, systems as AMIE [13] and AnyBURL [17] have been proposed to learn rules that describe the regularities in a given knowledge base. Before we started to develop our own method, we tried to apply these systems to the given problem. While WARMR can (in principle) learn the types of rules that we are interested in, it does not scale to the large process repositories that we are working with. AMIE and AnyBURL, on the other hand, have a restricted language bias which does not learn the types of rules that are important for recommendations in a process context.

Rule learning is also tightly linked to the field of Inductive Logic Programming (ILP). Prominent ILP approaches are, for example, FOIL [19] and Tilde [2]. While WARMR can also be regarded as an ILP approach, ILP techniques are usually based on a covering approach instead of mining all possibly relevant rules. 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. Another difference is the need for explicitly given negative examples, which are not available in the scenario we address. For these reasons we abstained from using ILP systems for the activity-recommendation problem.

Finally, our work itself represents a considerable extension and operationalization of ideas originally proposed in a doctoral consortium [21]. There, we conducted initial experiments using one simple rule type and only targeted a simple, context-agnostic recommendation setting.

3 Problem Definition

In this section, we discuss the transformation of business process models into business process graphs and formalize the activity-recommendation problem.

Business process graphs. Various modeling notations, such as Petri nets and BPMN, are available to capture business processes. Since we do not want to limit our approach to any specific notation, we extend the abstract view from Dijkman et al. [7], in which a process model is represented as a directed attributed graph:

Definition 1 (Business process graph). *Let $\mathcal{L}$ be a set of labels and $\mathcal{R}$ 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.*

In order to treat a business process model as a business process graph, we map the model's contents to the graph representation using an abstraction procedure. This procedure has several degrees of freedom: We might, for example, drop (or keep) certain types of nodes and have to select the types of behavioral relations that we use and assign to edges (e.g., *directly follows*). In the following, we focus on the abstraction of Petri nets. For other modeling notations it is possible to develop similar abstraction procedures. For instance, BPMN models can first be translated into Petri nets [8] before applying the abstraction approach.

From Petri net to business process graph. Given a Petri net, we consider transitions, which correspond to activities, as nodes in a business process graph, while omitting 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. [22], who propose three abstraction strategies, based on different sets of behavioral relations. We refer to [22] 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 $\tau(e) = \{followedBy\}$. 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).

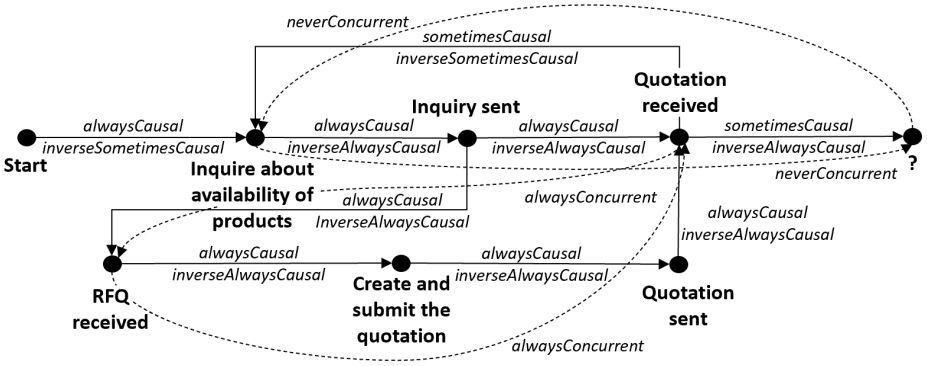

Fig. 2: A business process graph corresponding to the process model in Figure 1 using the relation types $\mathcal{R}^{causal+concurrent}$

- **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., $\tau(e) = \{alwaysCausal, inverseSometimesCausal\}$.
- **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 graph abstracts from some details, the overall structure and sequence of activities is preserved. Note that a $\mathcal{R}^{causal}$-graph can be obtained by omitting all dashed lines, while a $\mathcal{R}^{followedBy}$-graph corresponds to the $\mathcal{R}^{causal}$-graph in which all relation types are replaced by *followedBy*.

The activity-recommendation problem. With the concept of the abstraction of a business process model to a business process graph at hand, we can now proceed to the definition of the activity-recommendation problem. Given the current status of the process model under development, the activity-recommendation

problem is concerned with recommending suitable activities to extend the model at a user-defined position. Since the position of the activity that has to be recommended is given by the user (as the activity being added to a model), the activity-recommendation problem breaks down to finding a suitable label for the so far unlabeled activity node $\hat{n}$.

Definition 2 (Activity-recommendation problem). *Let $\mathcal{B}$ be a set of business process graphs and $\mathcal{L}_\mathcal{B}$ the set of activity labels that are used in $\mathcal{B}$. Let $B = (N, E, \lambda, \tau)$ be a given business process graph under development, where each node $n \in N$ except one node $\hat{n}$ is labeled, i.e., $\lambda(n)$ is given for all $n \in N \setminus \{\hat{n}\}$. The activity-recommendation problem is to find a suitable label $\lambda(\hat{n}) \in \mathcal{L}_\mathcal{B}$ for $\hat{n}$.*

In the next section, we propose our rule-based recommendation approach to address the activity-recommendation problem.

4 A Rule-Based Approach

Our rule-based recommendation approach consists of two main phases. First, *rule learning* derives rules that capture activity inter-relations from the process graphs in a repository. Second, *rule application* employs the learned rules to recommend the most suitable label for a target node in a process model under development, i.e., the recommended activity.

4.1 Rule Learning

Given a repository of business process graphs $\mathcal{B}$, we want to learn rules which capture regularities that appear in the use of labels within $\mathcal{B}$. For that purpose, we first need to determine the constants and predicates to be used to describe $\mathcal{B}$ in terms of logical formulas. We translate each $B = (N, E, \lambda, \tau) \in \mathcal{B}$ as follows.

1. 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).
2. For each node $n \in N$ we use a predicate λ_n that corresponds to the label $\lambda(n)$ of n and add a formula $\lambda_n(n)$, e.g., *quotationReceived(n)*.
3. 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 graph.

Given a set of $\mathcal{R}^X$, this means that we use $|\mathcal{R}^X| + 1$ binary predicates ($+1$ for *inSameProcess*) and $|\mathcal{L}_\mathcal{B}|$ unary predicates to describe the structure of the process graphs in the repository $\mathcal{B}$. Since a label can occur in multiple models, certain unary predicates will be used to describe the labels of nodes that appear in different process models, while the underlying nodes themselves always belong to exactly one model. Given these predicates, we next define *rule templates* and use them to capture activity regularities.

Rule templates. In the following we use h, j, k and l to refer to placeholders for unary predicates that correspond to the label of an activity, e.g., $l = \textit{submitPurchaseOrder}$. In this work we are interested in a special form of *horn rules*. Particularly, we are interested in rules that have the form $l(Z) \leftarrow \ldots$, which are rules that capture the regularities of activity Z being labeled with l.

To find such rules, our approach employs a set of rule patterns, i.e., templates, for which we generate all possible instantiations that hold in the repository $\mathcal{B}$. In particular, we define the following rule templates for a setting using directly-follows abstraction, i.e., with *followedBy* as the only relation type in $\mathcal{R}^{followedBy}$:

$$l(Z) \leftarrow \textit{inSameProcess}(Y, Z), k(Y) \tag{1}$$
$$l(Z) \leftarrow \textit{followedBy}(Y, Z), k(Y) \tag{2}$$
$$l(Z) \leftarrow \textit{inSameProcess}(X, Y), \textit{inSameProcess}(Y, Z), j(X), k(Y) \tag{3}$$
$$l(Z) \leftarrow \textit{inSameProcess}(X, Y), \textit{followedBy}(Y, Z), j(X), k(Y) \tag{4}$$
$$l(Z) \leftarrow \textit{followedBy}(X, Y), \textit{followedBy}(Y, Z), j(X), k(Y) \tag{5}$$
$$l(Z) \leftarrow \textit{followedBy}(W, X), \textit{followedBy}(X, Y), \textit{followedBy}(Y, Z), h(W), j(X), k(Y) \tag{6}$$

To operationalize the templates for the $\mathcal{R}^{causal}$ setting, we replace each occurrence of *followedBy* in templates (2), (4), (5) and (6) by each of the four types of causal relations in $\mathcal{R}^{causal}$. This results in various combinations, due to repeated occurrences of *followedBy* in certain templates. Specifically, we require 90 templates for the $\mathcal{R}^{causal}$ setting, primarily due to $4 \times 4 = 16$ different versions of (5) and $4 \times 4 \times 4 = 64$ of template (6). For brevity, we shall refer to the versions derived from one of the templates (1)-(6) as a *template group* in the remainder. To additionally incorporate the 3 types of concurrent relations in the $\mathcal{R}^{causal+concurrent}$setting, we introduce a further template group (7), which contains three templates that are similar to template (2), but in which the *followedBy* relation in (2) is replaced by one of the 3 *concurrent* relations.

Note that our approach is extendable, since the rule templates can be modified or complemented with additional ones. However, in light of the aforementioned combinatorial increase, 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.

When instantiating the rule templates, we replace h, j, k, and l by all possible label predicates created from $\mathcal{L}_\mathcal{B}$. This means that we, for example, have $|\mathcal{L}_\mathcal{B}| * (|\mathcal{L}_\mathcal{B}| - 1) \approx |\mathcal{L}_\mathcal{B}|^2$ different instantiations of templates (1) and (2).

Rule interpretation. Each of the defined templates captures a certain type of probabilistic regularity about activity inter-relations in process models. The probability of a rule that instantiates template (1) 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 (2)-rule tells us how probable it is that an activity k is directly followed by an activity l.

Here, it is important to consider that certain rules inherently relate to each other. For instance, any instantiation of a (2)-rule, always results in a corresponding (1)-rule as well. This is the case, because $\textit{inSameProcess}(X, Y) \leftarrow \textit{followedBy}(X, Y)$ is always true, i.e., if X can be followed by Y, then X and Y

are naturally also part of the same process model. Since the inverse is not true, i.e., $followedBy(X, Y) \leftarrow inSameProcess(X, Y)$ does not have to hold, we say that a (2)-rule is more *special* than a (1)-rule. Similar inter-relations also exist for the other rule templates. Whenever a rule r is more special than a rule r', rule r tends to make fewer and more specific predictions compared to rule r', which will be reflected in the ablation study in Section 5.

Rule confidence. For each *concrete rule*, i.e., an instantiation of one of the rule templates, we compute its *confidence* as a measure of its quality. For this, we follow the definition given in [13]. According to this definition, 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, our employed dataset leads to two rules with the same body related to activities that succeed a 'Quotation received' activity:

$r_1 =$ *submitPurchaseOrder(Z)* $\leftarrow$ *inverseAlwaysCausal(Y,Z), quotationReceived(Y)*
$r_2 =$ *analyzeQuotation(Z)* $\leftarrow$ *inverseAlwaysCausal(Y,Z), quotationReceived(Y)*

The body of both rules is the same and it holds 15 times over $\mathcal{B}$. This means that the pattern described by the body appears in 15 process models from $\mathcal{B}$. Considering that the head is additionally true, these numbers go down to 10 and 5, respectively. Thus, we have $support(r_1) = 10$, $support(r_2) = 5$, $confidence(r_1) = 10/15 = 0.667$, and $confidence(r_2) = 5/15 = 0.333$.

4.2 Rule Application

Given an unfinished business process graph B with its unlabeled node $\hat{n}$, we use the rules learned from $\mathcal{B}$ 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 resulting body is true. An example for a specific rule that instantiates template (4) in the $\mathcal{R}^{causal}$ setting is given by $(*)$. It is also a rule that leads to the top-ranked recommendation of the running example shown in Figure 1, where $\hat{n}$ is the rightmost node.

$$submitPurchaseOrder(\hat{n}) \leftarrow inSameProcess(X, Y), inverseAlwaysCausal(Y, \hat{n}), \quad (*)$$
$$createAndSubmitTheQuotation(X), quotationReceived(Y)$$

If we compare the rule to Figure 1, we can see that the body of this partially instantiated rule is indeed true, as we can map X and Y to nodes that have the respective labels. Thus, the rule recommends *submitPurchaseOrder* as label for $\hat{n}$. This recommendation is weighted via the confidence of the rule, which is $9/9 = 1$ with respect to the dataset used in our experiments. We do the same with all rules and collect the recommendations of the rules where the body was true with respect to the given unfinished model B.

Confidence aggregation. 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 [20]. After applying an aggregation method, we obtain a set of recommendations, each with a confidence score.

Then, we remove all recommendations from the set that refer to a label that is already used in B, since it is generally undesirable to have multiple activities with the same label in a single model. Since at most one of the recommendations in the set of recommended activities will be chosen, we normalize the confidence scores of the recommendations such that their sum equals 1. This changes the score for the label *submitPurchaseOrder* from 1 to 0.59.

Result explanation. One of the advantages of our approach is that the rules that serve as a basis for recommendations can also be used to explain provided recommendations. With respect to the given top recommendation, such an explanation can be phrased like: *Since the previous activity is 'Quotation received' and the process also includes a 'Create and submit the quotation' activity, there is a rather strong indication (normalized score of 0.59) that the activity should be labeled 'Submit purchase order'.* Such an explanation might raise the confidence of the user in the given recommendation and might make it easier for 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 supported this recommendation. Hence, the user can have a look at similar processes, which might further help her with the current modeling task.

5 Experimental Evaluation

In this section we report about our experimental studies[3] in which we assess the quality of the recommendations provided by our rule-based approach and compare our work against learning-based approaches [14,15] and the embedding-based RLRecommender technique [22]. Additionally, we present an ablation study that provides insights into the types of rules that are most important for our recommendation approach.

5.1 Dataset

To conduct the experiments, we employ the model collection of the Business Process Management Academic Initiative [18], which has also been used in the evaluation of RLRecommender [22]. The collection contains almost 30,000 process models in different process modeling languages. The models of the collection

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

are available in different revisions, which is useful for our purposes, since we consider each revision as a separate process model and thus ensure that most of the activities appear repeatedly across different processes in the repository.[4]

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

5.2 Evaluation Setup

We conducted our evaluation using 10-fold cross validation, i.e., training an approach on 90% of the dataset and evaluating it on the remainder. We report the mean results obtained over 10 runs (i.e., repetitions) of this cross validation.

Evaluation scenarios. We create one recommendation task for every business process graph in the evaluation set by following different strategies, which vary in terms of the amount of information that is available for the recommendation method, i.e., different ways to simulate the current status of a business process model under development. The basic idea is to remove some of the nodes and all edges connected to these nodes from a given process model. The remaining graph is treated as the intermediate result of a construction process. We choose a node from this graph as the one that has to be predicted and hide its label.

- **given-*k*.** In the given-k scenario, we pick a path of length $k+1$ which is a longest path from a source node (node with no incoming edges) to the activity at position $k+1$ and aim to predict the label of this activity. Especially for low values of k, the given-k evaluation method allows us to compare different methods in the '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 to a recommendation approach.
- **hide-last-two.** The opposite to this is the 'hide-last-two' evaluation method, which maintains a near 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 for which a label shall be predicted, while taking all other (non-hidden) activities into account.
- **breadth-first search (BFS).** Finally we have implemented a BFS-based evaluation method, 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.

Evaluation metrics. We quantify the relevance of provided recommendations using two metrics. First, we report on the *hit rate* H@10, which captures the

[4] Note that with the decision to keep these different revisions, we follow the setup used to evaluate RLRecommender.

fraction of *hits* in the top 10 recommendations, i.e., the fraction of cases where the activity that was actually used in a process model is among the 10 recommendations with the highest confidence score. Second, we report on the Mean Reciprocal Rank (MRR), which also takes the position of a hit in the recommendation list into account. For a given recommendation, the MRR has the value 0 if the actually chosen activity is not in the list and $1/p$ otherwise, where p denotes the position of the hit in the list. More precisely, we also consider a recommendation list of length 10 to compute MRR, which is a close approximation of the MRR that is based on the full ranking. However, this is more realistic, as the list of recommendations shown to the user has to be limited as well. In the ablation study we additionally report the average number of generated recommendations (i.e., the length of the recommendation list) per recommendation task as $\varnothing|\text{Rec}|$.

Approach configurations. We evaluate different configurations of our rule-based recommendation approach by varying two aspects. First, we vary the behavioral relations taken into accounts, i.e., we consider configurations that use the *followedBy*, *causal*, and *causal plus concurrent* relations. Second, we vary the aggregation method over the *max* and *noisy-or* methods described in Section 4.2. Combining these two aspects results in six different configurations, which we denote, for example, as Rules $followedBy^{max}$.

Baselines and other Approaches. We compare the performance of our rule-based approach against different baseline techniques and alternative approaches, derived from various existing works:

- CoOccur [14]: This technique is based on the conditional probabilities of the simultaneous occurrence of two activities in a process. Hence, this strategy recommends activities that co-occurred most often with the so far inserted activities of the unfinished process.
- kNN [14]: kNN is a weighted k-nearest-neighbors-based technique. It represents each process model as a vector, capturing whether or not an activity is present in a model. Then, kNN recommends activities that have not yet been included in a model, but appear in similar models in the repository.
- Link-Ctx [15]: Unlike the prior techniques, the link-based Link-Ctx technique takes the order of activities in process models into account. Specially, it considers which activities frequently occur directly after each other.
- Chain-Ctx [15]: The chain-based Chain-Ctx method generalizes Link-Ctx by considering longer chains of activities, i.e., it also considers recurring patterns consisting of three or more activities in order to provide recommendations that take a larger amount of contextual information into account.
- Hybrid-Ctx [15]: The Hybrid-Ctx technique combines a contextualized kNN strategy, kNN-Ctx, with Link-Ctx. Compared to kNN, kNN-Ctx increases the weight of the neighbour processes that contain activities, which are also included in the context of the process under development. Hybrid-Ctx is a weighting strategy which gives more weight to the kNN-Ctx technique for larger processes under development, while Link-Ctx receives a higher weight for smaller ones.

– RLRec [22]: The RLRecommender approach embeds activities and behavioral relations into a continuous low-dimensional space. The embedded vectors and their distances are then used to provide activity recommendations.

The first two methods CoOccur and kNN can be understood as simple baselines. The other methods are more sophisticated techniques that have especially been designed to perform well in the given (or a highly similar) activity recommendation scenario. Similar to the configurations of our approach, we assess the performance of RLRec for configurations that consider different behavioral relations, i.e., RLRec *followedBy*, RLRec *causal*, and RLRec *causal+conc*. The other approaches cannot distinguish between different relations, which means that we apply them in the $\mathcal{R}^{followedBy}$ setting only.

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

5.3 Evaluation Results

The results of our experiments are shown in Table 1. With exception of the specific 'cold-start' evaluation scenario given-1 our rule-based activity recommendation approach outperforms all other methods.

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

Table 1: Experimental results of the methods in different evaluation scenarios

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

equal results because RLRecommender bases its recommendations on one edge $(m, \hat{n})$ between the activity $\hat{n}$ that has to be labeled and another activity m in the given unfinished process only. In other words, the method does not collect and aggregate the predictions from all edges that are connected with $\hat{n}$. This also leads to the comparably low H@10 and MRR numbers.

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

The average time required to provide a recommendation is generally below 0.65s when running on an Intel® Xeon® E5-2623 v3@16x3.00 GHz CPU computer with 256G RAM. The LOO scenario using max-aggregation is an exception to this, which may require 1.1s on average to provide recommendations.

5.4 Ablation study

We investigate the importance of the individual rule templates (and groups) through an ablation study.[5] In particular, we evaluate the performance of our method when only learning and applying rules from one template group, e.g., a configuration R-(1) only considers rules related to the *inSameProcess* predicate used in template (1). Note that we apply the $\mathcal{R}^{followedBy}$ setting for the given-5 evaluation scenario while we adopt the $\mathcal{R}^{causal+concurrent}$ setting for the BFS and hide-last-two scenarios. Further, we employ the *max* aggregation for all settings.

The results in Table 2 reveal that the templates that include at least one

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

Table 2: Results of the ablation study

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

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

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

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

6 Conclusion

In this paper, we presented a rule-based approach for activity recommendation in process models. We demonstrated different configurations of our approach, highlighting its extendable nature. Our extensive experiments showed that it outperforms a variety of other approaches [14,15] including an embedding-based method [22]. In contrast to these approaches, our rule-based method is, furthermore, able to provide explanations alongside the given recommendations.

In future work, we aim to refine our rule-based method such that it is able to learn and apply rules to variations of previously seen labels or even to completely unseen labels. This requires other types of rule templates and the use of matching techniques that allow us to compare unseen labels with ones in the repository.

Acknowledgement We would like to thank Dietmar Jannach and Michael Jugovac for providing their code and kind help.

References

1. Agrawal, R., Imieliński, T., Swami, A.: Mining association rules between sets of items in large databases. In: SIGMOD. pp. 207–216. ACM (1993)

2. Blockeel, H., De Raedt, L.: Top-down induction of first-order logical decision trees. Artificial intelligence **101**(1-2), 285–297 (1998)
3. Cao, B., Yin, J., Deng, S., Wang, D., Wu, Z.: Graph-based workflow recommendation: on improving business process modeling. In: CIKM. pp. 1527–1531. ACM (2012)
4. De Raedt, L.: Logical and Relational Learning. Springer (2008)
5. Dehaspe, L., De Raedt, L.: Mining association rules in multiple relations. In: International Conference on ILP. pp. 125–132. Springer (1997)
6. Deng, S., Wang, D., Li, Y., Cao, B., Yin, J., Wu, Z., Zhou, M.: A recommendation system to facilitate business process modeling. IEEE Trans. Cybern. **47**(6), 1380–1394 (2017)
7. Dijkman, R.M., Dumas, M., García-Bañuelos, L.: Graph matching algorithms for business process model similarity search. In: BPM. vol. 5701, pp. 48–63. Springer (2009)
8. Dijkman, R.M., Dumas, M., Ouyang, C.: Semantics and analysis of business process models in BPMN. Inf. Softw. Technol. **50**(12), 1281–1294 (2008)
9. Dumas, M., La Rosa, M., Mendling, J., Reijers, H.A.: Fundamentals of Business Process Management. Springer, Berlin (2013)
10. Fellmann, M., Zarvic, N., Metzger, D., Koschmider, A.: Requirements catalog for business process modeling recommender systems. In: WI. pp. 393–407 (2015)
11. Frederiks, P.J., Van der Weide, T.P.: Information modeling: The process and the required competencies of its participants. DKE **58**(1), 4–20 (2006)
12. Friedrich, F., Mendling, J., Puhlmann, F.: Process model generation from natural language text. In: CAiSE. pp. 482–496. Springer (2011)
13. Galárraga, L.A., Teflioudi, C., Hose, K., Suchanek, F.: AMIE: Association rule mining under incomplete evidence in ontological knowledge bases. In: WWW. pp. 413–422 (2013)
14. Jannach, D., Fischer, S.: Recommendation-based modeling support for data mining processes. In: RecSys. pp. 337–340 (2014)
15. Jannach, D., Jugovac, M., Lerche, L.: Supporting the design of machine learning workflows with a recommendation system. ACM TiiS **6**(1), 1–35 (2016)
16. Li, Y., Cao, B., Xu, L., Yin, J., Deng, S., Yin, Y., Wu, Z.: An efficient recommendation method for improving business process modeling. IEEE Transactions on Industrial Informatics **10**(1), 502–513 (2014)
17. Meilicke, C., Chekol, M.W., Ruffinelli, D., Stuckenschmidt, H.: Anytime bottom-up rule learning for knowledge graph completion. In: IJCAI. pp. 3137–3143. AAAI Press (2019)
18. Model collection of the Business Process Management Academic Initiative, http://bpmai.org/
19. Quinlan, J.R.: Learning logical definitions from relations. Machine learning **5**(3), 239–266 (1990)
20. Russell, S., Norvig, P.: Artificial Intelligence: A Modern Approach. Prentice Hall, 3 edn. (2010)
21. Sola, D.: Towards a rule-based recommendation approach for business process modeling. In: ICSOC PhD Symposium. Springer (2020)
22. Wang, H., Wen, L., Lin, L., Wang, J.: RLRecommender: A representation-learning-based recommendation method for business process modeling. In: ICSOC. pp. 478–486. Springer (2018)
23. Zhang, Y., Chen, X.: Explainable recommendation: A survey and new perspectives. arXiv preprint arXiv:1804.11192 (2018)