Noname manuscript No.
(will be inserted by the editor)

Knowledge Graph Completion for Activity Recommendation in Business Process Modeling

Keyvan Amiri Elyasi · Diana Sola · Christian Meilicke · Han van der Aa · Heiner Stuckenschmidt

Received: date / Accepted: date

Abstract Activity recommendation is an approach to assist process modelers by recommending suitable activities to be inserted at a user-defined position. In this paper, we suggest approaching activity recommendation as a knowledge graph completion task. We convert business process models into knowledge graphs through various translation methods and apply embedding- and rule-based knowledge graph completion techniques to the translated models. Our experimental evaluation reveals that generic knowledge graph completion methods do not perform well on the given task. They lack the flexibility to capture complex regularities that can be learned using a rule-based approach specifically designed for activity recommendation.

Keywords activity recommendation · knowledge graph completion · rule learning · embeddings

1 Introduction

Business processes are an integral part of every organization. They refer to sequences of activities that are performed to achieve an outcome that is of interest to the organization itself or to its customers. Capturing information on such processes, *process models* are present in all phases of the business process management lifecycle [9]. Despite the relevance of process models for the documentation, analysis and improvement of business processes, creating them is a time-consuming and error-prone task that requires substantial expertise [12, 13]. Modeling business processes is even more challenging in the case of domain-specific processes, which often require the consistent use of specialized and technical vocabulary.

It is possible to support process modelers in their modeling task by providing recommendations on how the process model that they are developing can be expanded [11]. One way to provide such support is through *activity recommendation*. An activity recommendation system suggests suitable activities to extend a given business process model under development at a user-defined position. Such systems can use a repository of already available process models as a basis for their recommendations. Naturally, activity recommendation systems should be context-aware, i.e., they should take the current content and state of a process model into account when recommending a suitable label for the next activity.

Figure 1 shows context-aware recommendations for a process model under development. The user has just inserted an unlabeled activity at the model's bottom. The activity recommendation task is to suggest suitable activities for this position, i.e., to find an appropriate label for the so far unlabeled activity node. Since the process that has been modeled up to this point contains activities that are commonly associated with *order-to-cash* processes, a context-aware recommender system provides recommendations that correspond to activities that occur at a comparable position in similar processes (found in a given model repository), such as *Create sales order*, *Analyze purchase order*, and *Update sales order*.

Several specialized methods for solving the activity recommendation problem have been proposed [4, 6, 21, 34, 32, 33]. These methods range from ones based on symbolic representations, using rule learning to capture regularities across the process models in a large repository, to methods that use deep learning or large language models. What these methods have in com-

Keyvan Amiri Elyasi
Data and Web Science Group, University of Mannheim, Mannheim, Germany
E-mail: keyvan@informatik.uni-mannheim.de

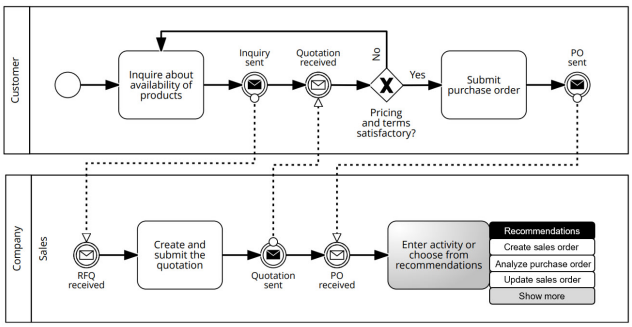


Fig. 1 A business process model under development

mon is that they have been specifically developed for the activity recommendation problem; generally stemming from researchers from the *business process management* community. Still, these methods struggle with the challenging nature of the problem, leaving considerable room for improvement in their recommendation accuracy.

Given that activity recommendation is thus far from solved, we propose to tackle it from a different angle. Specifically, we recognize that the activity recommendation problem can be rephrased as a *knowledge graph completion problem*.[1] Knowledge graph completion is a more general task than activity recommendation and has garnered a considerable amount of attention over the last years. A survey and an experimental evaluation that covers the most prominent methods is available in [27]. As an imperfect illustration of the attention for this task, we note that common knowledge graph completion methods such as TransE [2] and DistMult [44] have each attracted thousands of citations, several orders of magnitude more than methods for activity recommendation. Therefore, by rephrasing the problem of activity recommendation into this more general task, we are able to apply well-established methods stemming from a field with a long and extensive cumulative transition.

In this light, this paper makes three contributions:

1. We show how to rephrase the activity recommendation problem as a knowledge graph completion problem. To do this, we convert a given process model repository and process model under development into a large knowledge graph, which describes the relations and labels of the activities that appear in the models. The activity recommendation problem is then equal to a completion task for that graph.
2. We perform an experimental comparison of several knowledge graph completion methods on the activity recommendation problem. Our experiments cover both rule-based and embedding-based methods, which we also compare against methods specifically designed for activity recommendation.
3. By analyzing our experimental results we identify problems when applying general knowledge graph completion methods to our activity recommendation scenario. We propose a strategy to fix some of these problems and measure its impact on the recommendation quality.

Although our work focuses on the specific problem of activity recommendation for business process modeling, our results are also interesting from a general point of view. Most papers that propose knowledge graph completion methods evaluate these methods on the same evaluation datasets that have been used in the community for many years. Within this paper, we instead apply existing methods to a different dataset and an evaluation scenario that reflects a real-world downstream task and investigate if the methods are flexible enough to adapt to our task. The results indicate that the results do not fulfill expectations raised by the good results on the standard evaluation datasets.

The remainder of this paper is organized as follows. Section 2 presents an overview of related work in activity recommendation and knowledge graph completion. Section 3 provides a formal definition of the activity recommendation problem and different approaches for rephrasing it for knowledge graph completion. In Section 4, the results of our experimental study are discussed. Finally, the paper concludes with an outlook on future work in Section 6.

This journal paper is an improved and extended version of an AI4BPM workshop paper [35]. As we detected some minor inconsistencies in the experimental results, we repeated all experiments to verify and update the results. Moreover, we added three additional models (TuckER, HittER, and RotatE), which are known as top-performing knowledge graph embedding (KGE) models, while in our previous publication we used only two classic models. We also proposed and analyzed a new training method to enforce the models to focus on the label prediction task. Furthermore, we extended examples and explanations and discussed the results and the limitations of the approach in more detail (Section 5).

2 Related Work

Within this section we first give an overview on methods that have specifically been developed for activity recommendation. In the second part we are concerned with knowledge graph completion approaches and discuss, in

[1] This problem is also referred to as *link prediction*.

particular, the models that we use in our experimental evaluation.

2.1 Activity Recommendation Methods

The first methods designed for activity recommendation were mostly based on graph mining techniques [4, 6, 21]. Later Wang et al. developed an embedding-based approach called RLRecommender [42], which embeds activities and relations between them into a continuous low-dimensional space. As shown in [34], the performance of RLRecommender is comparably low, since the recommendations for an unlabeled activity are only based on one related activity in the process model. Therefore, the method generates different recommendations depending on the chosen related activity that is used to determine the recommendations for the unlabeled activity. In other words, RLRecommender lacks an aggregation method which combines all possible recommendations given the process model under development in one recommendation list.

In [34], the authors presented a rule-based activity recommendation approach. The proposed method learns rules that describe regularities in the use of labels in the given process model repository. For this purpose, the authors defined various rule types that capture different co-occurrence and structural patterns. Then the method applies the learned rules to the model under development making use of the full given context. In an extensive experimental study the rule-based method outperformed a variety of other approaches [19, 20, 42].

All approaches mentioned so far are limited to recommend only those labels that have previously been used in the models stored in the process model repository. In an extension of the rule-based approach proposed in [34], the authors developed in [32] a method that can recommend labels that have never been seen before. This approach combines components of the labels that are already known to create new labels. This results in some cases in the recommendation of labels that do not appear as a whole within the given repository. A similar method is presented in [15], where process element sequences are converted into text paragraphs. These textual data are then represented using sentence embeddings, which are learned text representations capturing semantic information as numerical vectors. In [33], the authors went a step further and presented an approach that is build on a language model. This approach is capable to recommend completely new labels where not even parts of these labels have been used in the process model repository previously.

Within respect to our research question we have to limit ourselves to approaches that predict only labels that have previously been used in the repository. We have to stick to this constraint as knowledge base completion methods are also limited to predict only those entities as candidates that are used in the given knowledge graph, i.e., that are elements from E. Thus we use only RLRecommender [42] and the rule-based method proposed in [34] as methods that have specifically been developed for the activity recommendation problem.

2.2 Knowledge Graph Completion Methods

We compare the selected methods for activity recommendation against some of the best and most prominent methods for solving the general problem of knowledge base completion. In particular, we focus on several knowledge graph embedding models and one rule-based system. To explain these methods, we first introduce the notion of a knowledge graph and describe the knowledge graph completion problem.

A knowledge graph $G = \{(e, r, e') \mid e, e' \in E \ \wedge \ r \in R\}$ is a set of triples. E denotes a set of entities. These entities can be persons, locations, organizations or, in our case, activities and their labels. R is a set of relations that might contain relations as $worksFor$ or $locatesIn$. From a logical point of view a relation is a binary predicate and a triple (e, r, e') is a atomic fact that expresses that e is in relation r to e'. Knowledge graphs are used to store our knowledge about a certain domain in simple formal representation.

Given that our knowledge about a certain domain is usually incomplete, we can assume that the knowledge graph itself is also incomplete. Knowledge graph completion deals with this problem in terms of completion tasks. A completion task (or completion query) asks to find a correct candidate for the question mark in an incomplete triple $(e, r, ?)$. The answer to such a query is a ranking of candidates. The higher the correct candidate is ranked the better. We introduce the metrics that are usually used to quantify the quality of a ranking in Section 2.3.

The field of knowledge graph completion has for a long time been dominated by knowledge graph embedding models. The standard evaluation protocol for knowledge graph completion has already been proposed in 2013 in the paper that also introduced the well-known model TransE [2]. TransE is a model that belongs to the family of translational models. Given a triple (e_1, r, e_2), in a (pure) translational model, the embedding of a relation r is used as a means to map (translate) the subject e_1 into the object e_2. Formally, we have $e_1^* + r^* = e_2^*$ if the embedding function is denoted by *. Another well-known translation model, that has proven to achieve very good results, is the model

RotatE [38]. Here the translation can be understood as a rotation in the complex embedding space. We included both TransE and RotatE in our experiments. Specifically, the RotatE model maps the entities and relations to the vector space and defines each relation as a rotation from the source entity to the target entity.

In adition to these models, we included two factorization models (DistMult [44] and TuckER [43]) and one hierarchical Transformer model (HittER [5]). These knowledge graph embedding (KGE) models are different in terms of how they combine entity and relation embeddings to capture the existence or absence of edges within the given graph. While DistMult uses a simple bilinear interaction between entities and relations [44], TuckER offers the advantages of parameter sharing across various relations and the decoupling of entity and relation embedding dimensions [43]. Finally, HittER employs a hierarchical Transformer model to capture the interaction between entity and relation embeddings.

As an alternative to the embedding-based models, we also consider AnyBURL [23,22], which is a rule learner specifically designed for knowledge graph completion problem. AnyBURL has been shown to perform on the same level as current state of the art KGE models [27]. Unlike embedding-based models, AnyBURL is a symbolic model that offers interpretability by providing the rules contributing to its predictions. In our experiments, we have incorporated AnyBURL as an additional general method for the knowledge graph completion task.

We have chosen the knowledge graph embedding models TransE, DistMult, TuckEr, HittER, RotateE for the following reasons: TransE and DistMult are probably the models that are used most often in different application settings. They can be understood as classical models. TuckEr, HittER and RotateE are younger models that have achieved very good results (see an overview in [27]). AnyBURL is a rule-based model that is also know to perform very well. All of these models have in common that they are generic knowledge graph completion models. None of these models has been specifically been designed for activity label recommendation. Within this paper we want to find out in how far these generic models can be used to solve label recommendation tasks. To understand how well these generic models perform, we compare them against the current state of the art approach for label recommendation described in [34]. This approach is a rule learning approach that is similar to AnyBURL. However, the supported rule types have specifically been designed for activity label recommendation. Additionally, we include RLRecommender [42], which is also a non-generic label recommendation approach that uses internally embeddings in rather specific way. Both approaches work directly on the given process models. To apply the generic models, we first need to translate the given process model into a knowledge graph. We explain this translation in Section 3.2.

2.3 Evaluation Criteria

The evaluation activity label recommendation should consider different aspects, which should be reflected in the criteria used to evaluate and compare recommendation techniques. To illustrate these, reconsider the example from Figure 1. The most important aspect is related to the quality of the predictions. In Figure 1, three possible candidates are shown to the user followed by the *Show more* option. These candidates are the top-3 candidates of a (in most cases) relatively long ranking of candidates. The predictive quality of an approach is usually quantified by the positions of the correct candidates. With respect to our use case position #1, #2 and #3 are displayed to the user, while a position lower in the ranking requires an additional interaction with the user interface to be displayed. Candidates that are ranked very low will not be displayed to the user at all.

In [2] the authors proposed the Hits@k measure to quantify the quality of predictions, where k is usually set to 1 or 10, resulting in Hits@1 and Hits@10 measures. Hits@10, for example, captures the fraction of *hits* in the top-10 recommendations, i.e., the fraction of cases where the label that was actually used in a process model is among the ten recommendations which are according to the employed method the most likely.

The Hits@10 metric distinguishes only between candidates ranked within the top-10 and those that are ranked lower. It does not differentiate between a correct candidate ranked at #2 and a correct candidate ranked at #9. To account for these differences, the MRR (mean rank reciprocal) has been proposed [39], which has become the most important metric in knowledge base completion and link prediction [27]. 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. The MRR is the mean of the reciprocal ranks of all generated recommendation lists. Within our experiments we consider a recommendation list of length 10 to compute the MRR, which is a close approximation of the MRR that is based on the full ranking. This approach is more realistic within our use case, as the list of recommendations shown to the user has to be limited. Within our example, one can imagine that first

the top-3 candidates are shown and by pressing *Show more* this list is enlarged to display the top-10.

However, the predictive quality of the recommended activities is only one aspect. Another aspect is related to the runtime of the algorithms that compute the recommendations. If we are concerned with a user interface depicted in Figure 1, the time required to make a prediction should be limited, to avoid waiting time for the user. Within this paper, we will not focus on an experimental analysis of runtimes. Instead, we refer to runtimes reported and discussed in [27]. If we look at the runtimes reported there, a trained model can make a single predictions within several milliseconds no matter if these models belong to the family of embedding-based models or rule-based models. However, there is a significant difference between both model families: Embedding-based models need to include the process model under development in the embedding. This means that each edit operation performed by the user requires to retrain or to update [16] the embeddings. While an update operation requires acceptable computational resources compared to a full training run, it is unclear in how far predictive quality remains stable.

Finally, a user might also be interested in the reason or an explanation for a recommendation. Symbolic approaches are usually better suited to deliver these explanations in terms of the rules that fired [30]. While there are also approaches to explain embedding-based models, more computational effort is required and it can be doubted that these explanations are as appropriate as the direct symbolic explanations of a rule based approach [1]. We will not further discuss this issue within this paper, where we focus mainly on the predictive quality of the models that we analyze in our experiments.

3 Activity Recommendation through Knowledge Graph Completion

In this section, we formally define the activity recommendation problem and discuss different approaches for applying knowledge graph completion methods.

3.1 Problem Definition

Our paper focuses on recommending activity labels for imperative process models, which are widely utilized due to their formal execution semantics and understandability to both business and IT users. These models are labeled directed graphs which explicitly define control-flow relations between model nodes (i.e. the execution order of activities). Common notations for imperative process modeling include BPMNs, Petri nets, and EPCs [10]. In our work, we abstract from specific notations and their unique node and edge types, instead utilizing the generic concept of a Business Process Graph (based on Dijkman et al. [8]), comprised of labeled nodes and directed edges, formally defined as follows:

Definition 1 (Business process graph) Let $\mathcal{L}$ be a set of labels. A business process graph is a tuple (N, E, λ), where N is a set of nodes, $E \subseteq N \times N$ is a set of directed edges and $\lambda : N \to \mathcal{L}$ is a function that maps a node to a label.

Given a process model, the set of nodes N of a business process graph corresponds to a subset of the nodes of the model, e.g., the (non-silent) transitions of a Petri net, whereas the set of edges E reflects the directly-follows relation between these nodes, defined by the model's execution semantics. The mapping function λ follows straightforwardly from the process model as well.

Given a process model under development, the activity recommendation problem is concerned with recommending suitable activities to extend the model at a user-defined position. The position of the activity that has to be recommended is given by the activity that was last added to the model. Therefore, the activity recommendation problem breaks down to finding a suitable label for the last added, and so far unlabeled, activity node $\tilde{n}$.

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 $B = (N, E, \lambda)$ be a given business process graph under development, where each node $n \in N$ except one node $\tilde{n}$ is labeled, i.e., $\lambda(n)$ is given for all $n \in N \backslash \{\tilde{n}\}$. The activity recommendation problem is to find a suitable label $\lambda(\tilde{n}) \in \mathcal{L}_\mathcal{B}$ for $\tilde{n}$.

In the next section, we explore various approaches to construct a knowledge graph from given process models. This enables us to apply general knowledge graph embedding models and rule-based methods to activity recommendation problem.

3.2 Knowledge Graph Construction

A knowledge graph is a collection of triples *(head entity, relation, tail entity)*. While entities are represented as nodes in the graph, the relation between two entities is given as a labeled edge. The entities that will appear as nodes in the knowledge graph comprise both the nodes

and the activity labels from the given business process graphs.

To utilize methods initially developed for knowledge graph completion in the context of activity recommendation, it is essential to represent each business process graph $B \in \mathcal{B}$ from the given repository, as well as the process model under development, in the form of such triples. For instance, a partial knowledge graph derived from a graph $B = (N, E, \lambda)$ might comprise the nodes N and the activity labels $\mathcal{L}$ as entities. We have developed three different approaches to translate a business process graph into a set of triples. These approaches differ in the sets of entities and relations employed in the knowledge graph construction process.

The simplest approach uses the relations *hasLabel* and *followedBy*. These two relations are the basis for all approaches, as they capture the core information of a business process graph as knowledge graph, i.e., a triple $(n, \mathit{hasLabel}, \lambda(n))$ expresses that a node n has the label $\lambda(n)$, whereas an edge (m, n) becomes the triple $(m, \mathit{followedBy}, n)$.

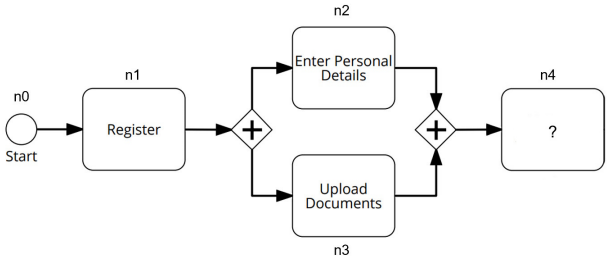
The other two approaches are alternative extensions which, in addition to the structural patterns captured by *followedBy*, consider co-occurrence patterns by using the relations *inSameProcess* and *inProcess*, respectively. This results in the following three translation approaches:

1. For each node $n \in N$, a triple $(n, \mathit{hasLabel}, \lambda(n))$ is added to the knowledge graph. Furthermore, we add for each edge $(m, n) \in E$ a triple $(m, \mathit{followedBy}, n)$.

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

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

To better understand the differences and commonalities between these translation approaches, we added Figure 2 where we have depicted the outcome of the different translation approaches in different colors. The black lines correspond to translation approach 1. By adding the blue lines to the black lines we get the results of translation approach 2b. Using the purple lines instead of the blue lines corresponds to translation approach 2b. Figure 2 illustrates also that translation 2a requires significantly more triples than the other approaches, as it needs to list all possible combination of activities within a process model.




```
n0  followedBy  n1
n1  followedBy  n2
n1  followedBy  n3
n2  followedBy  n4
n3  followedBy  n4
n0  hasLabel  "Start"
n1  hasLabel  "Register"
n2  hasLabel  "Enter Personal Details"
n3  hasLabel  "Upload Documents"
n0  inSameProcess  n1
n1  inSameProcess  n0
n0  inSameProcess  n2
n2  inSameProcess  n0
n0  inSameProcess  n3
n3  inSameProcess  n0
n0  inSameProcess  n4
n4  inSameProcess  n1
n1  inSameProcess  n2
n2  inSameProcess  n1
n1  inSameProcess  n3
n3  inSameProcess  n1
n1  inSameProcess  n4
n4  inSameProcess  n1
n2  inSameProcess  n3
n3  inSameProcess  n2
n2  inSameProcess  n4
n4  inSameProcess  n2
n3  inSameProcess  n4
n4  inSameProcess  n3
n0  inProcess  p1
n1  inProcess  p1
n2  inProcess  p1
n3  inProcess  p1
n4  inProcess  p1
```

Fig. 2 The upper part shows an example of a process model where the completion task (n_4, *hasLabel*, ?) has to be solved. The lower part shows the outcome of different translation approaches: black = 1, black+blue = 2a, black+purple = 2b

Different translation approaches are closely related, yet they exhibit subtle differences. In particular, two triples $\mathit{inProcess}(m, p)$ and $\mathit{inProcess}(n, p)$ in translation approach 2b imply $\mathit{inSameProcess}(m, n)$ triple in translation 2a and vice versa. However, there is a slight difference. In 2a two activities m and n are not in the same process, if the triple $\mathit{inSameProcess}(m, n)$ does not exist. In 2b they are not in the same process, be-

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

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

Given a knowledge graph under development (i.e., a knowledge graph in the validation or test set), we are interested in predicting tail entity in the completion task $(\tilde{n}, hasLabel, ?)$, where $\tilde{n}$ denotes the unlabeled node in the process model under development for which we want to suggest a suitable label. Once we applied one of the translation approaches, we can use a general knowledge graph completion technique to solve such a completion task.

3.3 Training Specifics and Knowledge Graph Augmentation

We can directly apply any standard knowledge graph completion approach on the outcome of our translation approaches. However, we have to be aware that there are some specifics in our setting that might require us to modify the standard training procedure to achieve good results. While our graph contains two or three relations, depending on the translation approach, we are only interested in completions task as $(\tilde{n}, hasLabel, ?)$. This means that we are always concerned with the *hasLabel* relation and, moreover, we are only asking for possible objects (tails) given an activity $\tilde{n}$ as subject (head). This query direction is sometimes called tail-direction.

Training a knowledge graph embedding model usually involves exploring a hyperparameter search space to identify the optimal or, at the very least, a proficient hyperparameter configuration. This process typically relies on a validation set. Adhering to this conventional approach, we have generated a validation set that resembles the test set by exclusively containing *hasLabel* triples. Note that in most standard evaluation datasets, the validation set contains triples for each of the relations that appear in the training set.

The validation set is also used to stop the training process after several epochs to avoid overfitting to the training data. This is usually done by analysing the development of the mean reciprocal rank (MRR) against the validation set. If the MRR no longer improves (for several epochs), the training phase ends and the model that has been learned is used in the prediction phase. As we know that we are only interested in the tail-direction, we use the MRR in tail-direction as the stopping criteria.

If we take a look at the example in Figure 2, it becomes obvious that the *hasLabel* triples are only a small fraction of all triples. This means that the embeddings of the entities are only to a limited degree determined by the activity labels. In [17] the authors argued that a similar setting is quite often the case in downstream applications of knowledge base completion that the authors refer to as the *recommendation case*. They propose a specific training strategy that is based on the idea to generate additional negative examples for the target relation that has to be predicted. Unfortunately, we were not able to re-implement the proposed approach. Instead of that we simply increased the number of *hasLabel* triples by adding for each *hasLabel* triple a copy of that triple. That results within the training process into twice as much *hasLabel* updates based on the positive and negative examples derived from the training triple. In our experiments, we report on the results for the original dataset and the dataset that we created with this augmentation strategy. We refer to the latter as the augmented dataset.

4 Experimental Study

In this section, we report on the design and results of our experimental study. We investigate the performance of different approaches for applying existing knowledge graph completion methods that have not specifically been designed for activity recommendation. Additionally, we compare the approaches to the embedding-based activity recommendation technique RLRecommender [42] and to the rule-based method presented in [34]. By analyzing the resulting predictions in detail, we detect an important reason for negative results and apply a post-hoc filtering technique to resolve it.

4.1 Process model repository

The *Business Process Management Academic Initiative* (BPMAI) [24] and *SAP Signavio Academic Models* (SAP-SAM) [36] datasets are the most representative collections of process models publicly available. SAP-SAM and BPMAI, are collections of models that were created over the course of roughly a decade on a platform that researchers, teachers, and students could use to create business (process) models. As both datasets are created by humans, they provide the closest representation of typical modeling projects. Both have similar characteristics, with SAP-SAM being an essentially larger version of BPMAI[2].

We chose to use the BPMAI dataset for our experiments for two primary reasons. First, it is more widely employed in previous research [31,34] for the label prediction task. Within this paper we analyze for the first time in how far standard KGE approaches can be used for label prediction. Thus, we focus only on results for a process model repository that has already been used for evaluating the activity recommendation problem. Second, size of BPMAI dataset is more manageable, allowing us to conduct more in-depth experiments, including hyperparameter tuning for knowledge graph embedding (KGE) models, thus providing more insights than could be achieved by using the SAP-SAM dataset.[3]

Contrary to previous work, we only use the last revisions of the BPMN 2.0 models in the collection. In contrast to using all revisions, we thus represent the possible case that the recommendation methods sometimes only have few or even none domain-specific reference models for the suggestion of activities available. This makes the datasets more realistic and increases the test hardness. Moreover, out of all last revision BPMN 2.0 models, we use only those with 3 to 50 activities and English labels. This choice results in a process model repository consisting of 3,688 process models. On average, the processes involve 14.3 activities while the standard deviation equals 8.3.

4.2 Evaluation setup

We employed a 80%-10%-10% data split to separate the process model repository into train, validation and test splits. For the experiments, we create one recommendation task for every process model in the validation and test split.

Evaluation procedure. We want to evaluate the approaches on realistic recommendation tasks. Therefore, we use an evaluation procedure in which we simulate the current status of a process model under development from a given business process graph by specifying the amount of information that is available for the recommendation. The basic idea is to remove some of the nodes and all edges connected to these nodes from a given business process graph while treating the remaining graph as the intermediate result of a modeling process. The employed evaluation procedure is based on breadth-first search. In this procedure, we randomly select one activity, which is neither a source nor sink node, as the one to be predicted. During the evaluation, we therefore filter out processes which do not have a chain of at least three different activities when executed. Then we hide the label of the chosen activity and determine the shortest path s from a source node to the activity. After that we hide all other activities that are not on a path of length s starting from a source node while the remaining activities and edges between them serve as a context for the recommendation task.

Evaluated methods. Evaluated methods encompass the rule learner AnyBURL [23], and five KGE models (also referred to as *embedding-based methods*).

To apply knowledge graph embedding models, we use the PyTorch-based library libKGE [3]. The selected KGE models encompass TransE [2] and RotatE [38] from the translational family, along with two factorization models: DistMult [44], TuckER [43]. Additionally, we incorporate HittER [5], which utilizes the Transformer model.

For the KGE model HittER, we opt for the context independent Transformer model available in libKGE. This variant comprises a three-layer entity Transformer, excluding the six-layer context Transformer [5]. We refer to this model in the following as HittER*. We also tested other popular KGE models (ComplEx [41] and ConvE [7]) but they yielded comparatively poor results that we do not report here.

Additionally, the results of the rule-based activity recommendation method [34] and the results of the embedding-based technique RLRecommender [42] are included in our evaluation. Both techniques have been developed with a special focus on the activity recommendation problem. One of our main goals within this paper is to find out whether these specialized techniques perform worse or better compared to the general knowledge graph completion methods.

[2] While SAP-SAM dataset contains more than one million business (process) models, its subset BPMAI dataset, includes roughly 30 thousands models.

[3] Note that we performed the experiments dividedly on two computers: Intel® Xeon® CPU E5-2640 v3@40x2.40 GHz and Intel® Xeon® Silver 4114 CPU@40x2.20 GHz. The runtimes of the evaluated KGE models remain in a reasonable range of maximum 48 hours for the hyperparameter search given a particular translation approach.

T	A	Entities	Rel.	Train	Valid.	Test
1	- ×2	70,876	2	105,150 153,059	358	362
2a	- ×2	70,876	3	955,492 1,003,401	358	362
2b	- ×2	74,548	3	153,827 201,736	358	362

Table 1 Statistics for the six knowledge graphs used to evaluate knowledge graph embedding models. With "T" and "A" as the employed translation and augmentation approaches, "Entitities" as the number of entities in the knowledge graph, "Rel." as the number of relation types, and "Train", "Valid.", "Test" as the number of triples per split.

Datasets. We take each of the three translation approaches defined in Section 3.2 along with their augmented counterparts (see Section 3.3) to construct six different knowledge graphs for our experiments. Each of these knowledge graphs captures the triples obtained from process models in the training split (with or without augmentation), complemented with the triples describing the context of the process models under development from the validation and test splits.

Table 1 shows the statistics of these graphs, showing the number of entities, relation types, and number of triples in the training, validation, and test sets.

While the translation approaches 1 and 2a yield 70,876 entities, which is the sum of the total number of activity nodes (48,677) and activity labels (22,199), approach 2b also considers processes as entities, which adds the total number of processes in the evaluation (3,672) to the sum.

Relations *followedBy* and *hasLabel* are the basis for all approaches. In the training set, there exist 57,241 instances of the *followedBy* relation across all datasets. Additionally, there are 47,909 instances of the *hasLabel* relation in the datasets that are not augmented. The count of the this relation is doubled in augmented datasets. Moreover, translation approaches 2a, 2b consider co-occurrence patterns. This is done by either incorporating 850,342 instances of the *inSameProcess* relation in the former translation approach or 48,677 instances of the *inProcess* relation in the latter one.

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

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

It is noteworthy that the augmentation approach outlined in Section 3.3 has no impact on the outcomes of AnyBURL. While for an embedding-based method two identical triples result into two updates within the training phase, two identical triples collapse into the same logical formula from the perspective of AnyBURL. Consequently, our experimentation encompasses a total of 33 distinct combinations, incorporating various translation approaches, both with and without augmentation, and an array of knowledge graph completion methods.

Hyperparameters. For the KGE models, we generally employed the large hyperparameter space presented by Ruffinelli et al. [28]. However, we additionally considered the embedding sizes 32 and 64 for all five KGE models as well as the embedding size of 16 for TuckER, HittER*, and RotatE. The rationale behind expanding the hyperparameter search space is the limited diversity of relation types in our knowledge graph which makes larger KGE models prone to overfitting. Following the hyperparameter optimization algorithm presented in [28], we conducted a quasi-random hyperparameter search, followed by a Bayesian optimization phase.

While the rule-based activity recommendation technique proposed in [34] operates without hyperparameters, AnyBURL is configured with two parameters. Specifically, the length of cyclic rules is set to 5, and the length of acyclic rules is set to 2 (as AnyBURL does not support a higher parameter value)[4]. Aside from these changes we used AnyBURL in its default setting. Also note that rule-based approaches usually do not fine-tune their hyperparameters against the validation set. Thus, AnyBURL and the specialized rule-based activity recommendation method from [34] do not make any use of the validation set.

[4] These parameter settings are specified by `MAX_LENGTH_CYCLIC = 5` and `MAX_LENGTH_ACYCLIC = 2`.

4.3 Results

The results of our experiments are shown in Table 2. Before we look at the results of the general knowledge graph completion methods, we first conside the results of the specialized activity recommendations methods (RLRecommender and the state-of-the-art rule-based approach). RLRecommender, which has been published in 2018, has an MRR of 23.8% and a Hits@10 score of 35.1%. The latter means that in roughly one third of all recommendations the correct one is among the top-10 list of recommendation. These results have clearly been topped in the rule-based approach proposed in 2021 (see [34]) with an MRR of 41.4% and a Hits@10 score of 47.5%, which means that in nearly half of all recommendation tasks a correct recommendation is among the top-10 list. This is a clear improvement and divides the space of possible results into three areas: i) outcomes inferior to RLRecommender, ii) outcomes at least comparable to RLRecommender but falling short of the state-of-the-art rule-based approach (highlighted in bold in Table 2), and iii) outcomes as accurate as or potentially outperforming the state-of-the-art rule-based approach (highlighted in bold and underlined in Table 2).

When looking at the results obtained by general knowledge graph completion methods, we observe that the Hits@10 numbers are at least 10 percentage points worse than those achieved by the rule-based activity recommendation approach and also the MRR scores are worse. This means that no combination of tested methods and translation approaches works well for the activity recommendation problem. While some of the KGE models demonstrate Hits@10 scores that are better than those achieved by RLRecommender, it is evident that both specialized methods, particularly the rule-based approach, significantly outperform general knowledge graph completion methods.

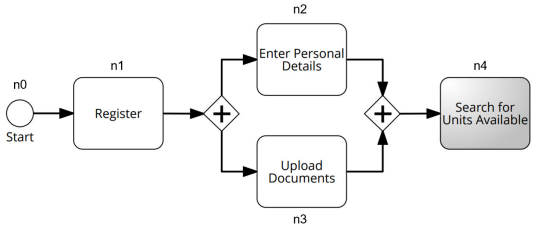
In terms of Hits@10, TuckER stands out as the most proficient model. AnyBURL and DistMult achieve similar yet slightly worse results. These top three models yield results at least on par with RLRecommender. Notably, AnyBURL and DistMult exhibit robust stability, consistently delivering comparable outcomes across various translation approaches. In contrast, TuckER's performance experiences a decline specifically when employing translation approach 2a. Surprisingly, providing more information through *inSameProcess* or *inProcess* relations does not always translate into higher Hits@10 figures for TuckER. This trend extends beyond TuckER, encompassing all knowledge graph completion methods, with the exception of DistMult. In the case of DistMult, the incorporation of supplementary information through *inSameProcess* or *inProcess* relations consistently leads to enhanced performance, observed through improvements in both Hits@10 and MRR metrics.

While some of the general knowledge graph completion methods exhibit comparable Hits@10 results to the RLRecommender, they perform surprisingly bad with respect to the MRR metric. DistMult emerges as the most successful model, followed by AnyBURL and TuckER. However, none of these three models manage to surpass RLRecommender, and evidently the performance gap between these models and the Rule-based approach is quite significant.

Examination of the Predictions. To spot the reason for these unexpected results, we looked at some concrete cases. Figure 3 shows a process model in the validation set, where the activity *Search for Units Available* has been randomly selected as the one to be predicted. The

T	A	Method	Hits@10	MRR
1	-	TransE	29.2 %	7.3 %
		DistMult	**35.1 %**	14.5 %
		TuckER	**37.8 %**	14.2 %
		HittER*	27.9 %	10.1 %
		RotatE	30.4 %	13.2 %
		AnyBURL	**36.1 %**	14.8 %
	×2	TransE	29.3 %	7.2 %
		DistMult	**36.5 %**	15.0 %
		TuckER	33.4 %	14.8 %
		HittER*	31.2 %	12.1 %
		RotatE	29.6 %	12.4 %
2a	-	TransE	11.0 %	2.6 %
		DistMult	**35.4 %**	21.1 %
		TuckER	34.2 %	11.4 %
		HittER*	14.3 %	4.1 %
		RotatE	25.1 %	8.1 %
		AnyBURL	**37.2 %**	15.2 %
	×2	TransE	22.1 %	5.5 %
		DistMult	33.4 %	21.7 %
		TuckER	30.1 %	10.2 %
		HittER*	22.6 %	6.2 %
		RotatE	23.2 %	7.0 %
2b	-	TransE	30.9 %	6.2 %
		DistMult	**35.4 %**	15.2 %
		TuckER	**38.4 %**	14.3 %
		HittER*	15.7 %	5.9 %
		RotatE	20.4 %	6.4 %
		AnyBURL	**35.8 %**	14.5 %
	×2	TransE	31.5 %	6.4 %
		DistMult	**35.1 %**	14.6 %
		TuckER	**37.3 %**	15.0 %
		HittER*	25.4 %	10.2 %
		RotatE	23.5 %	10.0 %
RLRecommender [42]			**35.1 %**	**23.8 %**
Rule-based Approach [34]			**47.5 %**	**41.4 %**

Table 2 Experimental results



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

Fig. 3 An example of a process model where the completion task (n_4, *hasLabel*, ?) has to be solved, given that *Search For Units Available* is the correct answer, and the top-10 recommendations of TransE in combination with translation approach 1.

use of translation approach 1 combined with TransE yields the top 10 recommendations shown on the right.

The correct activity *Search for Units Available* is at position eight of the recommendation list, which means that the MRR would be $1/8 = 0.125$, if this was the only completion task of the whole evaluation. Surprisingly, some of the items in the recommendation list are not labels but activity nodes of the given process model, i.e., n3 and n4, as well as activity labels like *Register* or *Enter Personal Details* that have already been used in the process model. Clearly, the recommendation of activity nodes is not useful since we are interested in the prediction of activity labels. Also, it is likely that activity labels that have already been inserted into the process model are not added a second time. Therefore, we decided to do a post-processing in which we filter out other recommendations than labels, i.e., activity nodes and processes, as well as activity labels that are already present in the given process model. In the example of Figure 3, this means that the correct prediction moves from position eight to position four of the recommendation list. This corresponds to an MRR improvement from 0.125 to 0.25.

Results with Post-processing. We applied this post-processing to all our experimental results. This resulted in significant improvements shown in Table 3. The percentages in brackets indicate the improvement of the associated Hits@10 or MRR numbers in comparison to the results without post-processing. It seems general knowledge graph completion methods are not capable to distinguish between different types of entities. This has already been observed in [18], where the authors argue that especially knowledge graph embedding methods often violate constraints on the domain and range of relations. In our case this corresponds to a missing distinction between activity nodes and activity labels. Moreover, none of the approaches, including the rule learner AnyBURL, was able to learn that a label that has already been used in the process under development will not be used for another activity node in the same process.

If we now compare the results of the general knowledge graph completion methods with post-processing to the results of the specialized methods (i.e., RLRecommender and the rule-based activity recommendation), we observe that the gap between general and specialized methods has become less significant after post-processing. Except for HittER*, every general knowledge graph completion technique demonstrates results that are, at the very least, comparable to those achieved by RLRecommender.

As described in Section 2, RLRecommender [42] is based on a rather specific approach to use embeddings in which only one related activity in the process model is used for the recommendation of an activity. In contrast, knowledge graph embedding (KGE) models do not face such constraints and, as a result, have the potential to surpass RLRecommender in performance. This is particularly evident in the case of translation approach 2b, where models such as TuckER, DistMult and TransE exhibit superior performance in Hits@10 and MRR.

The post-processing step notably enhances the performance of the TransE model, leading to its superior results in the Hits@10 metric. However, the TransE results strongly depend on the translation approach. Similar observation applies to TuckER, which excels in terms of MRR. On the contrary, both DistMult and AnyBURL showcase greater stability, consistently delivering comparable results across various translation approaches.

5 Discussion and practical implications

In this section we will analyse performance of generic knowledge graph completion methods in more detail. Our goal is to explore why these methods do not reach state-of-the-art performance and discuss the practical implications of our findings.

Based on the results provided in Table 3, translation approach 2b proves to be the most effective for the KGE models TransE, TuckER and DistMult. This outcome underscores the value of the additional explicit information presented in triples featuring the *inProcess* relation. In contrast, approach 2a works comparably

Translation approach	Augmentation	Method	Hits@10	MRR
1	-	TransE	**41.7 %** (+ 7.7 %)	**24.9 %** (+ 21.1 %)
		DistMult	**37.8 %** (+ 2.7 %)	**29.1 %** (+ 14.6 %)
		TuckER	**39.2 %** (+ 1.4 %)	**28.6 %** (+ 14.4 %)
		HittER*	31.2 % (+ 3.3 %)	20.5 % (+ 10.4 %)
		RotatE	**36.2 %** (+ 5.8 %)	**24.1 %** (+ 10.9 %)
		AnyBURL	**36.9** % (+ 0.1 %)	**24.4%** (+ 9.6 %)
	×2	TransE	34.5 % (+ 5.2 %)	20.8 % (+ 13.6 %)
		DistMult	**39.5 %** (+ 3.0 %)	**30.1 %** (+ 15.1 %)
		TuckER	**35.1 %** (+ 1.7 %)	**23.8 %** (+ 9.0 %)
		HittER*	33.1 % (+ 1.9 %)	21.1 % (+ 9.0 %)
		RotatE	31.2 % (+ 1.6 %)	21.1 % (+ 8.7 %)
2a	-	TransE	**37.3 %** (+ 26.3 %)	18.2 % (+ 15.6 %)
		DistMult	**35.9 %** (+ 0.5 %)	**28.6 %** (+ 7.5 %)
		TuckER	**36.3 %** (+ 2.1 %)	**25.8 %** (+ 14.4 %)
		HittER*	17.6 % (+ 3.3 %)	7.8 % (+ 3.7 %)
		RotatE	34.8 % (+ 9.7 %)	17.3 % (+ 9.2 %)
		AnyBURL	**38.6 %** (+ 1.4 %)	**24.2 %** (+ 9.0 %)
	×2	TransE	31.5 % (+ 9.4 %)	15.1 % (+ 9.6 %)
		DistMult	**35.6 %** (+ 2.2 %)	**27.3 %** (+ 5.6 %)
		TuckER	32.8 % (+ 2.7 %)	21.9 % (+ 11.7 %)
		HittER*	25.7 % (+ 3.1 %)	10.3 % (+ 4.1 %)
		RotatE	30.9 % (+ 7.7 %)	15.9 % (+ 8.9 %)
2b	-	TransE	**42.5 %** (+ 9.1 %)	**29.3 %** (+ 21.2 %)
		DistMult	**36.7 %** (+ 1.3 %)	**29.8 %** (+ 14.6 %)
		TuckER	**40.9 %** (+ 2.5 %)	**32.1 %** (+ 17.8 %)
		HittER*	18.0 % (+ 2.3 %)	10.4 % (+ 4.5 %)
		RotatE	28.2 % (+ 7.8 %)	13.5 % (+ 7.1 %)
		AnyBURL	**36.4 %** (+ 0.5 %)	**24.4 %** (+ 9.8 %)
	×2	TransE	**41.4 %** (+ 9.9 %)	**27.8 %** (+ 21.4 %)
		DistMult	**38.1 %** (+ 3.0 %)	**28.5 %** (+ 13.9 %)
		TuckER	**41.4 %** (+ 4.1 %)	**28.9 %** (+ 13.9 %)
		HittER*	27.1 % (+ 1.7 %)	16.0 % (+ 5.8 %)
		RotatE	26.5 % (+ 3.0 %)	16.1 % (+ 6.1 %)
RLRecommender			**35.1 %**	**23.8 %**
Rule-based Approach			**47.5 %**	**41.4 %**

Table 3 Experimental results with post-processing

poor in combination with the embedding-based methods. One reason for this could be that in approach 2a the nodes are strongly interconnected via the relation *inSameProcess*. Thus, the interconnection of the nodes via *inSameProcess* is more prominent than via the relation *followedBy*. This can be disadvantageous since the co-occurrence patterns depicted by *inSameProcess* are often less relevant than the structural patterns captured by *followedBy*, which avoid recommending activities that have high co-occurrence statistics, but are not relevant at the current model position.

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

$$hasLabel(X, register) \leftarrow inSameProcess(X, Y), \\ hasLabel(Y, upload\ documents)$$

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

Overall, the rule-based method [34], which has been specifically designed for activity recommendation, surpasses the performance of the best general knowledge graph completion methods, exhibiting a minimum improvement of 5 % in Hits@10 and 9 % in MRR. These significant differences illustrate that a general knowledge graph completion method cannot compete with an approach which has specifically been designed for activity recommendation in business process modeling. This remains the case despite our thorough exploration of various translation approaches, augmentation of the

knowledge graph, and the development of a problem specific filtering as a post-processing step to increase the quality of the results.

The specific regularities crucial for effective activity recommendations appear to exert only a limited influence on the resulting embedding space. These specifics are reflected in the types of rules supported by the rule-based recommendation method [34] that are also more expressive compared to the general rule types supported by AnyBURL. We can conclude that general methods for knowledge graph completion are not flexible enough to adapt to the given problem resulting in a relatively low prediction quality.

The limited predictive accuracy of knowledge graph embedding (KGE) models stems from their focus solely on semantic evidence, disregarding local evidence such as subgraphs around query triples. These models learn triples independently as they learn an embedding for each entity and relation. Therefore, they may overlook sequential information inherent in triples [46]. In our knowledge graph, *followedBy* relation is limited to activity nodes. Consequently, it is impossible for KGE models to capture similar sequential relations between activity label nodes because information about such sequential relationships is only preserved along the paths between activity label nodes. Therefore, crucial information about the sequential order of activity labels may be overlooked by KGE models. To address this, path-based knowledge graph completion methods like NBFNet [48], A^*Net [47], and RED-GNN [46] can be applied to activity recommendation problem in future research. These models predict relations between entities by employing message-passing neural networks [14] over all paths between them. Like rule-based methods such as AnyBURL, they provide interpretability by identifying the most influential paths for predicting the label of newly added activity node in the process model.

KGE models also face the challenge of overfitting, evident in differences between loss function values for training, validation, and test triples. To understand this limitation, we compared our study's datasets with common benchmarks for knowledge graph completion. Our datasets have notably fewer relation types and are sparser. For example, while FB15K-237 [40] and Yago3-10 [25] contain 237 and 37 relation types, ours only have 2 or 3. Similarly, the ratio of triples to entities is much lower in our datasets compared to benchmark datasets.

Due to the sparse nature of our datasets and their limited relation types, general Knowledge Graph Embedding (KGE) models exhibit excessive flexibility in embedding entities in the vector space. Consequently, entity embeddings are susceptible to initialization and randomness, and more importantly are prone to overfitting. Notably, overfitting issue is pronounced for more flexible models like HittER* and RotatE. In contrast, a simpler model like TransE tends to yield superior results. Additionally, we found that KGE models with smaller embedding sizes often perform better.

Leveraging larger training datasets, such as the SAP-SAM dataset [36], can potentially reduce overfitting risks when training general KGE models for activity recommendation tasks. Furthermore, investigating alternative translation methods, including inverse relationships like *precededBy* and behavioural relationships such as *indirectFollowedBy*, holds significance for future research. These additional relations could augment knowledge graph density and its informativeness, consequently diminishing the likelihood of overfitting.

We showed that generic knowledge graph completion methods do not match the performance of specialized rule-based approaches as outlined in [34]. However, our findings have broader implications. They suggest greater potential for adapting knowledge graph completion methods in the business process domain, particularly in process modeling. Moreover, our results indicate that knowledge graph embedding (KGE) models may underperform on sparse knowledge graphs with few relation types. This underscores the need for further research on KGE models tailored to different characteristics of knowledge graphs beyond those found in benchmark datasets.

6 Conclusion and Future Work

In this paper, we presented different approaches to use knowledge graph completion methods for activity recommendation in business process modeling. A problem-specific filtering as post-processing step improved the quality of the predictions. However, the rule-based activity recommendation method [34] still worked better than the application of various general knowledge graph completion methods, which revealed their lack of flexibility to adapt to the given problem. In summary, we conclude that the use of generic knowledge graph completion methods is not a good choice for solving the activity recommendation problem. Our empirical results indicate the use of the problem-specific rule-based approach proposed in [34] is currently the best solution for any application, e.g., activity recommendation in a process model editor (see Figure 1), that requires to present ranked recommendations to a user.

A key limitation of the general knowledge graph completion methods employed in this study is their lack of explicit encoding for local sub-graphs between entity pairs. Consequently, these models may struggle

to capture intricate regularities crucial for determining the label of the unlabeled activity node. A potential remedy to tackle this challenge involves the use of the Neural Bellman-Ford Networks in which representation of a pair of entities is defined as the generalized sum of all the path representations connecting them [48]. Similar path-based knowledge graph completion approaches like A^*Net [47], and RED-GNN [46] can be applied to the activity recommendation problem, offering several advantages. Firstly, these approaches can learn not only from semantic evidence but also from local evidence provided by relational path between entities [47]. Secondly, they identify influential paths between entities for predicting labels, enhancing explainability. Thirdly, they feature a lower number of model parameters compared to KGE models, potentially reducing risk of overfitting and improving predictive accuracy [48]. Lastly, path-based methods can generalize to unseen entities as path semantics are determined solely by relations rather than entities [47]. To implement path-based approaches, the knowledge graph representation of process model repository needs augmentation with inverse relations (e.g., *precededBy*) and identity relations (i.e., self-loops for all entities). Besides path-based approaches, alternative techniques working on (RDF) knowledge graphs, e.g., Graph Convolutional Networks [29] or RDF2Vec [26], could be tested in the future.

Another direction for future research is related to the current state of the process model under development that should be taken into account as a context for the recommendation of an activity. When putting the context into the training set, as done for the embedding-based methods, the KGE models have to be retrained after every activity that has been added to the model under development. This is very impractical for the application of such methods for real-time recommendations, as the training takes too much time. In future work, we would like to explore ways of avoiding the need for complete retraining when the process model under development has been extended, as it has, for example, been done by Song et al. [37] for evolving knowledge graphs and translation-based embeddings.

Last but not least, it is important to acknowledge that our current exploration has focused exclusively on co-occurrence and structural patterns, a choice negatively influenced by the inherent sparseness of knowledge graphs. In future research, we intend to incorporate textual patterns, potentially leveraging approaches such as KG-Bert [45].

Reproducibility: Our source code and the configuration files for different knowledge graph completion methods are publicly accessible at `https://github.com/keyvan-amiri/KGE-ActivityRecommendation`.

References

1. Betz, P., Meilicke, C., Stuckenschmidt, H.: Adversarial explanations for knowledge graph embeddings. In: IJCAI. vol. 2022, pp. 2820–2826 (2022)
2. Bordes, A., Usunier, N., García-Durán, A., Weston, J., Yakhnenko, O.: Translating embeddings for modeling multi-relational data. In: NIPS. pp. 2787–2795 (2013)
3. Broscheit, S., Ruffinelli, D., Kochsiek, A., Betz, P., Gemulla, R.: LibKGE - A knowledge graph embedding library for reproducible research. In: EMNLP: System Demonstrations. pp. 165–174 (2020)
4. 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)
5. Chen, S., Liu, X., Gao, J., Jiao, J., Zhang, R., Ji, Y.: HittER: Hierarchical Transformers for Knowledge Graph Embeddings (Oct 2021), arXiv:2008.12813 [cs]
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. Dettmers, T., Minervini, P., Stenetorp, P., Riedel, S.: Convolutional 2d knowledge graph embeddings. In: Proceedings of the AAAI conference on artificial intelligence. vol. 32 (2018)
8. 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)
9. Dumas, M., La Rosa, M., Mendling, J., Reijers, H.A.: Fundamentals of Business Process Management. Springer, Berlin (2013)
10. Fahland, D., Lübke, D., Mendling, J., Reijers, H., Weber, B., Weidlich, M., Zugal, S.: Declarative versus imperative process modeling languages: The issue of understandability. In: Halpin, T., Krogstie, J., Nurcan, S., Proper, E., Schmidt, R., Soffer, P., Ukor, R. (eds.) Enterprise, Business-Process and Information Systems Modeling. pp. 353–366. Springer (2009)
11. Fellmann, M., Zarvic, N., Metzger, D., Koschmider, A.: Requirements catalog for business process modeling recommender systems. In: WI. pp. 393–407 (2015)
12. 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)
13. Friedrich, F., Mendling, J., Puhlmann, F.: Process model generation from natural language text. In: CAiSE. pp. 482–496. Springer (2011)
14. Gilmer, J., Schoenholz, S.S., Riley, P.F., Vinyals, O., Dahl, G.E.: Neural message passing for quantum chemistry. In: Precup, D., Teh, Y.W. (eds.) Proceedings of the 34th International Conference on Machine Learning. Proceedings of Machine Learning Research, vol. 70, pp. 1263–1272. PMLR (2017)
15. Goldstein, M., González-Álvarez, C.: Augmenting modelers with semantic autocompletion of processes. In: Polyvyanyy, A., Wynn, M.T., Van Looy, A., Reichert, M. (eds.) Business Process Management Forum. pp. 20–36. Springer International Publishing (2021)
16. Hamaguchi, T., Oiwa, H., Shimbo, M., Matsumoto, Y.: Knowledge transfer for out-of-knowledge-base entities: A graph neural network approach. arXiv preprint arXiv:1706.05674 (2017)

17. Hubert, N., Monnin, P., Brun, A., Monticolo, D.: New strategies for learning knowledge graph embeddings: The recommendation case. In: International Conference on Knowledge Engineering and Knowledge Management. pp. 66–80. Springer (2022)
18. Hubert, N., Monnin, P., Brun, A., Monticolo, D.: Sem@ k: Is my knowledge graph embedding model semantic-aware? arXiv preprint arXiv:2301.05601 (2023)
19. Jannach, D., Fischer, S.: Recommendation-based modeling support for data mining processes. In: RecSys. pp. 337–340 (2014)
20. Jannach, D., Jugovac, M., Lerche, L.: Supporting the design of machine learning workflows with a recommendation system. ACM TiiS **6**(1), 1–35 (2016)
21. 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)
22. Meilicke, C., Chekol, M.W., Betz, P., Fink, M., Stuckeschmidt, H.: Anytime bottom-up rule learning for large-scale knowledge graph completion. The VLDB Journal pp. 1–31 (2023)
23. 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)
24. Model collection of the BPM Academic Initiative, `http://bpmai.org/`
25. Rebele, T., Suchanek, F., Hoffart, J., Biega, J., Kuzey, E., Weikum, G.: YAGO: A multilingual knowledge base from wikipedia, wordnet, and geonames. In: Groth, P., Simperl, E., Gray, A., Sabou, M., Krötzsch, M., Lecue, F., Flöck, F., Gil, Y. (eds.) The Semantic Web – ISWC 2016. pp. 177–185. Springer International Publishing (2016)
26. Ristoski, P., Paulheim, H.: Rdf2vec: Rdf graph embeddings for data mining. In: ISWC. pp. 498–514. Springer (2016)
27. Rossi, A., Barbosa, D., Firmani, D., Matinata, A., Merialdo, P.: Knowledge graph embedding for link prediction: A comparative analysis. ACM Transactions on Knowledge Discovery from Data (TKDD) **15**(2), 1–49 (2021)
28. Ruffinelli, D., Broscheit, S., Gemulla, R.: You CAN teach an old dog new tricks! On training knowledge graph embeddings. In: ICLR. OpenReview.net (2020)
29. Schlichtkrull, M., Kipf, T.N., Bloem, P., van den Berg, R., Titov, I., Welling, M.: Modeling relational data with graph convolutional networks. In: ISWC. pp. 593–607. Springer (2018)
30. Schramm, S., Wehner, C., Schmid, U.: Comprehensible artificial intelligence on knowledge graphs: A survey. Journal of Web Semantics **79**, 100806 (2023)
31. Sola, D.: Towards a rule-based recommendation approach for business process modeling. In: ICSOC PhD Symposium. Springer (2020)
32. Sola, D., van der Aa, H., Meilicke, C., Stuckenschmidt, H.: Exploiting label semantics for rule-based activity recommendation in business process modeling. Information Systems **108**, 102049 (2022)
33. Sola, D., van der Aa, H., Meilicke, C., Stuckenschmidt, H.: Activity recommendation for business process modeling with pre-trained language models. In: European Semantic Web Conference. pp. 316–334. Springer (2023)
34. Sola, D., Meilicke, C., Van der Aa, H., Stuckenschmidt, H.: A rule-based recommendation approach for business process modeling. In: CAiSE. Springer (2021)
35. Sola, D., Meilicke, C., van der Aa, H., Stuckenschmidt, H.: On the use of knowledge graph completion methods for activity recommendation in business process modeling. In: Marrella, A., Weber, B. (eds.) Business Process Management Workshops. pp. 5–17. Springer International Publishing, Cham (2022)
36. Sola, D., Warmuth, C., Schäfer, B., Badakhshan, P., Rehse, J.R., Kampik, T.: Sap signavio academic models: a large process model dataset. In: International Conference on Process Mining. pp. 453–465. Springer (2022)
37. Song, H.J., Park, S.B.: Enriching translation-based knowledge graph embeddings through continual learning. IEEE Access **6**, 60489–60497 (2018)
38. Sun, Z., Deng, Z.H., Nie, J.Y., Tang, J.: RotatE: Knowledge Graph Embedding by Relational Rotation in Complex Space (Feb 2019), arXiv:1902.10197 [cs, stat]
39. Toutanova, K., Chen, D.: Observed versus latent features for knowledge base and text inference. In: Proceedings of the 3rd workshop on continuous vector space models and their compositionality. pp. 57–66 (2015)
40. Toutanova, K., Chen, D.: Observed versus latent features for knowledge base and text inference. In: Allauzen, A., Grefenstette, E., Hermann, K.M., Larochelle, H., Yih, S.W.t. (eds.) Proceedings of the 3rd Workshop on Continuous Vector Space Models and their Compositionality. pp. 57–66. Association for Computational Linguistics (2015)
41. Trouillon, T., Welbl, J., Riedel, S., Gaussier, É., Bouchard, G.: Complex embeddings for simple link prediction. In: International conference on machine learning. pp. 2071–2080. PMLR (2016)
42. 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)
43. Wang, Y., Broscheit, S., Gemulla, R.: A Relational Tucker Decomposition for Multi-Relational Link Prediction (Feb 2019), arXiv:1902.00898 [cs, stat]
44. Yang, B., tau Yih, W., He, X., Gao, J., Deng, L.: Embedding entities and relations for learning and inference in knowledge bases. In: ICLR (Poster) (2015)
45. Yao, L., Mao, C., Luo, Y.: KG-BERT: BERT for knowledge graph completion. CoRR **abs/1909.03193** (2019)
46. Zhang, Y., Yao, Q.: Knowledge graph reasoning with relational digraph. In: Proceedings of the ACM Web Conference 2022. pp. 912–924. WWW '22, Association for Computing Machinery (2022). https://doi.org/10.1145/3485447.3512008
47. Zhu, Z., Yuan, X., Galkin, M., Xhonneux, L.P., Zhang, M., Gazeau, M., Tang, J.: A*net: A scalable path-based reasoning approach for knowledge graphs. Advances in Neural Information Processing Systems **36**, 59323–59336 (2023)
48. Zhu, Z., Zhang, Z., Xhonneux, L.P., Tang, J.: Neural Bellman-Ford Networks: A General Graph Neural Network Framework for Link Prediction. In: Advances in Neural Information Processing Systems. vol. 34, pp. 29476–29490. Curran Associates, Inc. (2021)