On the Use of Knowledge Graph Completion Methods for Activity Recommendation in 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 essential for organisations. However, it is a time-consuming task that requires expert knowledge. In particular, this is the case when modeling domain-specific processes, which often involves the consistent use of technical terminology. Process modelers can be supported through the provision of recommendations on how the model under development can be expanded. Activity recommendation is one such support approach, in which suitable activities to be inserted at a user-defined position are recommended. Recently, it has been suggested to treat activity recommendation as a knowledge graph completion task and to apply methods from this discipline. In this paper, we investigate different approaches to apply embedding- and rule-based knowledge graph completion methods out of the box and evaluate them in an experimental study. Additionally, we compare them to two methods that have specifically been designed for activity recommendation.

Keywords: activity recommendation, knowledge graph completion, rule learning, embeddings

1 Introduction

Business processes are an integral part of every organisation. They refer to sets of activities, which are performed to achieve an outcome that is of interest to an organization or its customers. Capturing information on such processes, *process models* are present in all phases of the business process management lifecycle [5]. 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 [7,8]. 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 [6]. One possibility for such a support is *activity recommendation*. An activity recommendation system suggests suitable activities to extend a given

business process model under development at a user-defined position. A repository of already available process models can be used as a basis for this task. Activity recommendation systems should be context-aware, i.e., they should take the current content and state of a process model into account. 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*.

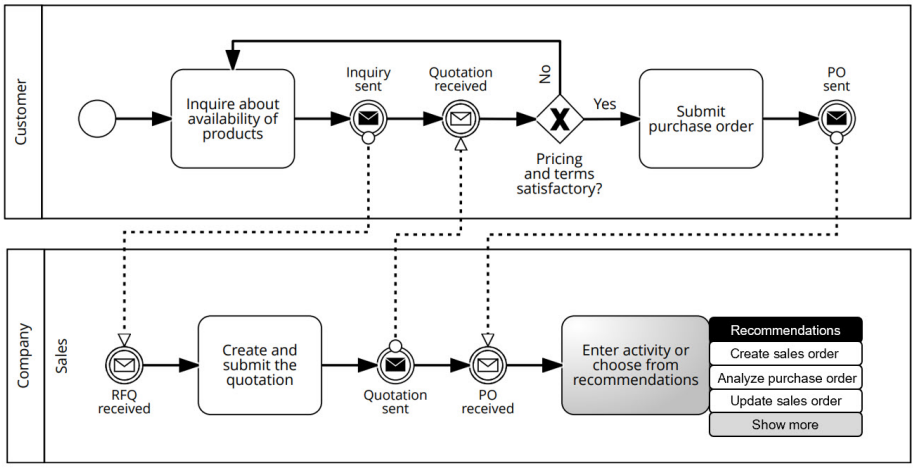


Fig. 1: A business process model under development

The activity-recommendation problem can be framed in terms of a *knowledge graph completion* task [21]. A knowledge graph captures data on a specific domain in the form of entities and their inter-relations. In our setting, we propose to turn a process model repository and a process model under development into a large knowledge graph, which thus captures knowledge on the relations between activities in the models, e.g., activities that commonly follow each other. Then, the recommendation of an appropriate activity is interpreted as a completion task in the graph, to which so-called knowledge graph completion approaches can be applied. This completion task has attracted lots of attention in the last decade with the majority of the approaches being based on the idea to embed the knowledge graph into a low dimensional space [22]. More recently, rule-based approaches have shown to be competitive alternatives [12].

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

While our main focus lies on the application of knowledge graph completion techniques to the field of activity recommendation for business process modeling, the results are also interesting from a general point of view. Most papers that propose knowledge graph completion techniques 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.

2 Related Work

Besides activity recommendation approaches that make use of graph mining techniques [3, 4, 11], our work primarily relates to embedding- and rule-based methods for activity recommendation.

Wang et al. developed the embedding-based recommendation approach RLRecommender [21], which embeds activities and relations between them into a continuous low-dimensional space. However, as we already proved in prior work [19], the performance of RLRecommender is comparably low, since the recommendations for an unlabeled activity are based on one related activity in the process model only. 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 of an aggregation method which combines all possible recommendations given the process model under development in one recommendation list.

In prior work [19], we presented our rule-based activity recommendation approach. First, the method learns rules that describe regularities in the use of labels in the given process repository. For this purpose, we have defined various rule types that capture different co-occurrence and structural patterns. Then, it 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 [9, 10, 21].

Beyond the scope of activity recommendation, our work relates to knowledge graph embedding (KGE) models and rule-based systems that can be used for knowledge graph completion. Two prominent KGE models are TransE [1] and DistMult [23]. While TransE models relations between entities in the knowledge graph as translations operating on the embeddings of the entities, DistMult uses a simple bilinear interaction between entities and relations. An alternative to the KGE models is given by AnyBURL [12], which is a rule learner designed for knowledge graph completion and has been shown to perform on the same level as current state of the art KGE models [15].

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 work is independent from a particular modeling notation. Therefore, we define the activity-recommendation problem based on the notion of business process graphs. Such a graph represents an abstracted version of a business process model in a particular modeling notation, such as a Petri net or BPMN model. Specifically, we employ the following definition:

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 $n \in N \setminus \{\tilde{n}\}$. 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 discuss different approaches to construct a knowledge graph from given process models such that we can apply knowledge graph embedding models and rule-based methods.

3.2 Knowledge Graph Construction

A knowledge graph is a collection of triples *(head entity, relation, tail entity)*. While the entities are represented as nodes in the graph, the relation between two entities is given as a labeled edge. To apply methods that have originally been developed for knowledge graph completion to the activity-recommendation problem, we need to describe each business process graph $B \in \mathcal{B}$ of the given repository and the process model under development in terms of such triples. The

partial knowledge graph resulting from a graph $B = (N, E, \lambda)$ can, for example, comprise the nodes N and the activity labels $\mathcal{L}$ as entities. We developed three different approaches to translate a business process graph into a set of triples, which vary in the used sets of entities and relations.

The simplest approach uses the relations *hasLabel* and *followedBy*. These two relations are the basis for all approaches, as they capture the core information of a business process 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 $n \in N$ we add a triple $(n, \mathit{hasLabel}, \lambda(n))$. 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 $(n, \mathit{inProcess}, \pi(B))$.

The different translation approaches are closely related. In particular, in 2b the two triples $\mathit{inProcess}(m, p)$ and $\mathit{inProcess}(n, p)$ imply $\mathit{inSameProcess}(m, n)$ in 2a and vice versa. However, there is a slight difference. In 2a two activities m and n are not in the same process, if the triple $\mathit{inSameProcess}(m, n)$ does not exist. In 2b they are not in the same process, because we have $\mathit{inProcess}(m, p)$ and $\mathit{inProcess}(n, p')$ with $p \neq p'$. Detecting that two activities m and n are in the same process with approach 1 is a bit more complicated, as it requires 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 an obtained knowledge graph, we are interested in predictions of the completion task $(\tilde{n}, \mathit{hasLabel}, ?)$, where $\tilde{n}$ denotes the unlabeled node in the process model under development for which we want to suggest a suitable label. For this task, existing completion approaches can be employed, as shown next.

4 Experimental Study

In this section, we report on the design and results of our study in which we investigate the performance of different approaches for applying existing knowledge graph completion methods that have not specifically been designed for activity recommendation. Additionally, we compare the approaches to the embedding-based activity recommendation technique RLRecommender [21] and to the rule-based method from our earlier work [19].

4.1 Dataset

For the experiments we employ the model collection of the Business Process Management Academic Initiative [13]. Unlike in our previous work [18, 19], we only use the last revisions of the BPMN 2.0 models in the collection. In contrast to using all revisions, we thus represent the possible case that the recommendation methods sometimes only have few or even none domain-specific reference models for the suggestion of activities available. Moreover, the runtimes of the evaluated KGE models remain in a reasonable range of maximum 48 hours for the hyperparameter search given a particular translation approach[3]. Out of all last revision BPMN 2.0 models, we use those with 3 to 50 activities and English labels. This choice results in a dataset 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 separate the dataset into train, validation and test splits. More precisely, we train each method on 80 % of the process models while we use 10 % of the models for validation and 10 % for evaluation.

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 source nor sink node, as the one to

[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.

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. For the experiments, we create one recommendation task for every process model in the validation and test split.

Evaluated methods. We combine each of the three translation approaches defined in Section 3.2 with three existing knowledge graph completion methods. For this, we employ the TransE [1] and DistMult [23] KGE models (also referred to as *embedding-based methods*), as well as the rule learner AnyBURL [12][4]. For the application of the KGE models we use the PyTorch-based framework libKGE [2]. While these approaches are rather old, they have proven to yield good results when trained in an appropriate way [16]. Additionally, we include the results of our previously proposed rule-based activity recommendation method [19] and of the embedding-based technique RLRecommender by Wang et al. [21], which have both been developed with a special focus on activity recommendation.

Dataset statistics. Table 1 shows the statistics of the dataset for the embedding-based methods, i.e., the number of entities and relations as well as the number of triples in the training, validation, and test sets for each translation approach. 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.

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

Table 1: Overview of the dataset statistics for the embedding-based methods

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

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

needs to be part of the learned embeddings, this means that the contexts of the process models in the validation and test splits must be included in the training set. As such, the training set additionally contains all context triples of the validation and test process models. This is not necessary for rule-based approaches such as AnyBURL, since they naturally allow for the consideration of the context, which thus does not have to be included in the training set.

Hyperparameters. For the KGE models, we basically employed the large hyperparameter space presented by Ruffinelli et al. [16]. However, we additionally considered the embedding sizes 32 and 64. While our previously proposed rule-based activity recommendation method [19] has no hyperparameters, we have set the following hyperparameters for AnyBURL: We increased the length of cyclic rules to 5 and the length of acyclic rules to 2 (a higher parameter is not supported by AnyBURL)[5]. 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, both AnyBURL and the rule-based activity recommendation method from our previous work do not make any use of the validation set.

Evaluation metrics. We use two different metrics for quantifying the relevance of provided recommendations. First, we report on the *hit rate* Hits@10, which captures the fraction of *hits* in the top 10 recommendations, i.e., the fraction of cases where the activity label that was actually used in a process model is among the ten recommendations which are according to the employed method the most likely. Second, we report on the Mean Reciprocal Rank (MRR), which additionally takes the position of a hit in the recommendation list into account. 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. For the MRR we take the mean of the reciprocal ranks of all generated recommendation lists. Note that we 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.

4.3 Results

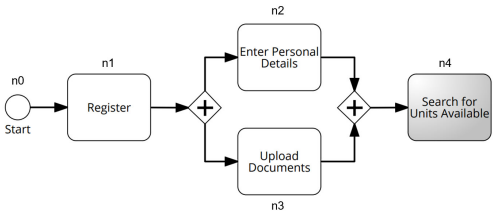
The results of our experiments are shown in Table 2. With Hits@10 numbers that are at least 10 % worse than the rule-based activity recommendation approach and low MRR numbers, no combination of tested methods and translation approaches works well. The two specialized methods, in particular the rule-based method, that are shown in the last two lines of the table work significantly better.

Examination of the Predictions. To spot the reason for these unexpected results, we looked at some concrete cases. Figure 2 shows a process model in the validation set, where the activity *Search for Units Available* has been randomly

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

Translation approach	Method	Hits@10	MRR
1	TransE	34.0 %	8.4 %
	DistMult	35.9 %	15.0 %
	AnyBURL	36,1 %	14,8 %
2a	TransE	20.7 %	4.2 %
	DistMult	34.3 %	19.1 %
	AnyBURL	37.2 %	15.2 %
2b	TransE	33.4 %	8.0 %
	DistMult	36.5 %	15.6 %
	AnyBURL	35.8 %	14.5 %
RLRecommender [21]		35.1 %	23.8 %
Rule-based Approach [19]		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. 2: An example of a process model in the validation set where the completion task (n_4, *hasLabel*, ?) has to be solved and the top-10 recommendations of TransE in combination with translation approach 1

selected as the one to be predicted. The 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. Other items of the recommendation list include some nodes of the given process model, i.e., n3 and n4, as well as activities like *Register* or *Enter Personal Details* that have already been used in the process model. Clearly, the recommendation of nodes is not useful since we are interested in the prediction of activities. Also, it is likely that activities that have already been inserted into the process model are not added a second time. Therefore, we decided to do a post-processing in which we filter out other recommendations than labels, i.e., nodes and processes, as well as activities that are already present in the given process model. In the example of Figure 2, this means that the correct prediction moves to position four of the recommendation list which is an MRR improvement from 0.125 to 0.25.

Results with Post-processing. The results of the experiments with post-processing are 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.

Translation approach	Method	Hits@10	MRR
1	TransE	41.7 % (+ 7.7 %)	24.9 % (+ 21.1 %)
	DistMult	38.7 % (+ 2.8 %)	29.8 % (+ 14.8 %)
	AnyBURL	36.9 % (+ 0.1 %)	24.4% (+ 9.6 %)
2a	TransE	37.0 % (+ 16.3 %)	20.8 % (+ 16.6 %)
	DistMult	37.3 % (+ 3.0 %)	29.2 % (+ 10.1 %)
	AnyBURL	38.6 % (+ 1.4 %)	24.2 % (+ 9.0 %)
2b	TransE	42.5 % (+ 9.1 %)	29.3 % (+ 21.2 %)
	DistMult	39.8 % (+ 3.3 %)	31.4 % (+ 15.8 %)
	AnyBURL	36.4 % (+ 0.5 %)	24.4 % (+ 9.8 %)
RLRecommender [21]		35.1 %	23.8 %
Rule-based Approach [19]		47.5 %	41.4 %

Table 3: Experimental results with post-processing

The KGE model TransE profits most from the post-processing. However, the TransE results strongly depend on the translation approach. This effect is smaller when using DistMult or AnyBURL. For the embedding-based methods TransE and DistMult, translation approach 2b is the best, which shows that the additional explicit information given by the triples with the relation *inProcess* is useful. In contrast, approach 2a works comparably poor in combination with the embedding-based methods. One reason for this could be that in approach 2a the nodes are strongly interconnected via the relation *inSameProcess*. Thus, the interconnection of the nodes via *inSameProcess* is more prominent than via the relation *followedBy*. This can be disadvantageous since the co-occurrence patterns depicted by *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 $(m, \mathit{inSameProcess}, n)$ to express that two nodes m and n are in a process p, approach 2b needs the two triples $(m, \mathit{inProcess}, p)$ and $(n, \mathit{inProcess}, p)$. This has a direct impact on the regularities that can be captured by AnyBURL. A rule as $\mathit{hasLabel}(X, \mathit{register}) \leftarrow \mathit{inSameProcess}(X, Y), \mathit{hasLabel}(Y, \mathit{upload\ documents})$ 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.

If we now compare the results of the standard knowledge graph completion methods with post-processing to the results of the specialized methods RLRecommender and the rule-based activity recommendation approach, we observe that the gap between standard and specialized methods has become less significant after post-processing. As described in Section 2, RLRecommender [21] is based on a rather specific approach to use embeddings in which only one related activity in the process model is used for the recommendation of an activity, thus its results are slightly worse compared to the results of our approaches for using TransE and DistMult. This holds in particular for translation approach 2b where both TransE and DistMult are better in Hits@10 and MRR.

The rule-based method [19], which has specifically been designed for activity recommendation, is still at least 5 % better in Hits@10 and 10 % better in MRR. These significant differences illustrate that a standard knowledge graph completion method cannot compete with an approach which has specifically been designed for activity recommendation in business process modeling. This holds even though we tried out different translation approaches and developed a problem-specific filtering as post-processing step to increase the quality of the results.

It seems that the specific regularities which are important for making a good activity recommendation seem to influence the resulting embedding space only to a limited degree. These specifics are reflected in the types of rules supported by the rule-based recommendation method [19] that are also more expressive compared to the general rule types supported by AnyBURL. We can conclude that standard methods for knowledge graph completion are not flexible enough to adapt to the given problem resulting in a relatively low prediction quality.

5 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 still worked better than the application of various standard knowledge graph completion methods, which revealed their lack of flexibility to adapt to the given problem.

In the future, alternative techniques working on (RDF) knowledge graphs, e.g., Graph Convolutional Networks [17] or RDF2Vec [14], could be tested. Until now we have only considered co-occurrence and structural patterns, which suffers from the sparseness of the knowledge graphs. We would also like to take textual patterns into account in future, for example, by applying KG-Bert [24].

Another future direction 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. [20] for evolving knowledge graphs and translation-based embeddings.

References

1. Bordes, A., Usunier, N., García-Durán, A., Weston, J., Yakhnenko, O.: Translating embeddings for modeling multi-relational data. In: NIPS. pp. 2787–2795 (2013)
2. 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)

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. 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)
5. Dumas, M., La Rosa, M., Mendling, J., Reijers, H.A.: Fundamentals of Business Process Management. Springer, Berlin (2013)
6. Fellmann, M., Zarvic, N., Metzger, D., Koschmider, A.: Requirements catalog for business process modeling recommender systems. In: WI. pp. 393–407 (2015)
7. 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)
8. Friedrich, F., Mendling, J., Puhlmann, F.: Process model generation from natural language text. In: CAiSE. pp. 482–496. Springer (2011)
9. Jannach, D., Fischer, S.: Recommendation-based modeling support for data mining processes. In: RecSys. pp. 337–340 (2014)
10. Jannach, D., Jugovac, M., Lerche, L.: Supporting the design of machine learning workflows with a recommendation system. ACM TiiS **6**(1), 1–35 (2016)
11. 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)
12. 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)
13. Model collection of the BPM Academic Initiative, http://bpmai.org/
14. Ristoski, P., Paulheim, H.: Rdf2vec: Rdf graph embeddings for data mining. In: ISWC. pp. 498–514. Springer (2016)
15. 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)
16. Ruffinelli, D., Broscheit, S., Gemulla, R.: You CAN teach an old dog new tricks! On training knowledge graph embeddings. In: ICLR. OpenReview.net (2020)
17. 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)
18. Sola, D.: Towards a rule-based recommendation approach for business process modeling. In: ICSOC PhD Symposium. Springer (2020)
19. Sola, D., Meilicke, C., Van der Aa, H., Stuckenschmidt, H.: A rule-based recommendation approach for business process modeling. In: CAiSE. Springer (2021)
20. Song, H.J., Park, S.B.: Enriching translation-based knowledge graph embeddings through continual learning. IEEE Access **6**, 60489–60497 (2018)
21. 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)
22. Wang, Q., Mao, Z., Wang, B., Guo, L.: Knowledge graph embedding: A survey of approaches and applications. IEEE Transactions on Knowledge and Data Engineering **29**(12), 2724–2743 (2017)
23. 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)
24. Yao, L., Mao, C., Luo, Y.: KG-BERT: BERT for knowledge graph completion. CoRR **abs/1909.03193** (2019)