Activity Recommendation for Business Process Modeling with Pre-trained Language Models

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

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

Abstract. Activity recommendation in business process modeling is concerned with suggesting suitable labels for a new activity inserted by a modeler in a process model under development. Recently, it has been proposed to represent process model repositories as knowledge graphs, which makes it possible to address the activity-recommendation problem as a knowledge graph completion task. However, existing recommendation approaches are entirely dependent on the knowledge contained in the model repository used for training. This makes them rigid in general and even inapplicable in situations where a process model consists of unseen activities, which were not part of the repository used for training. In this paper, we avoid these issues by recognizing that the semantics contained in process models can be used to instead pose the activity-recommendation problem as a set of textual sequence-to-sequence tasks. This enables the application of transfer-learning techniques from natural language processing, which allows for recommendations that go beyond the activities contained in an available repository. We operationalize this with an activity-recommendation approach that employs a pre-trained language model at its core, and uses the representations of process knowledge as structured graphs combined with the natural-language-based semantics of process models. In an experimental evaluation, we show that our approach considerably outperforms the state of the art in terms of semantic accuracy of the recommendations and that it is able to recommend and handle activity labels that go beyond the vocabulary of the model repository used during training.

Keywords: activity recommendation · process models · semantic process analysis · language models · sequence-to-sequence models

1 Introduction

All organizations, from enterprises to governmental institutions, to healthcare providers, perform processes to deliver services or products to their internal and external customers [10]. Each of these processes consists of a number of activities, which jointly turn an initial trigger into a desired outcome, such as *order-to-cash*, *purchase-to-pay*, or *ticket-to-resolution*. Process models are widely used artifacts to capture information on such processes, since they represent

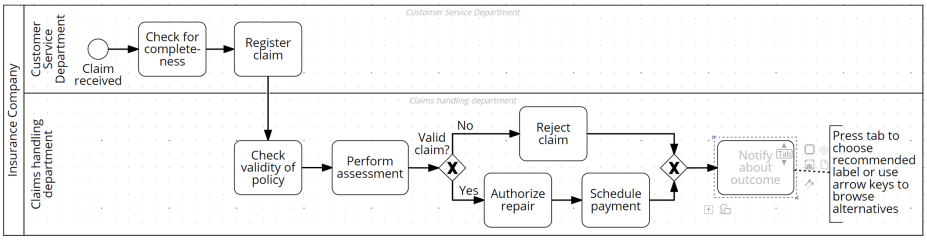


Fig. 1: A business process model under development

the semantics of a process in a structured manner, typically in the form of BPMN (Business Process Model and Notation) models or Petri nets. As graph-based process representations, they are a suitable basis for a variety of purposes, including process execution support, analysis, and improvement [8].

However, establishing process models is known to be a time-consuming and error-prone task [16, 17], in part due to its dependence on knowledge from domain experts, who are typically not familiar with process modeling itself [10,44]. Furthermore, in cross-departmental settings, ensuring clarity and consistency in established process models is even harder [42], yet also crucial to avoid that process execution and analysis are conducted based on incorrect, incomplete, or inconsistent models [1, 5]. Recognizing these issues, various methods have been proposed to support process modelers, which include methods that identify syntactic issues [12, 29] or provide modeling recommendations [15] in the form of suggestions on how to expand a process model under development.

Activity recommendation represents the most common instantiation of such recommendation support [9,47,52], which aims to suggest suitable labels for new activities placed by modelers. Figure 1 shows an instance of this. The BPMN model in the figure depicts a process that starts when a claim has been received, after which various activities are performed to handle the claim. This involves a decision point, indicated by an *XOR-split gateway* (diamond shape with an X), where a claim is either rejected, or its payment is authorized and scheduled. Following this decision, the model synchronizes the two branches using an *XOR-join gateway*. After this gateway, a new activity has been inserted, for which the activity-recommendation task is to suggest one or more suitable labels. As shown, a recommended label is "*Notify about outcome*". This label is fitting, because the preceding nodes indicate that the outcome of a claim has been determined, after which it is natural to inform the claimant.

To provide such recommendations, approaches typically extract knowledge from a repository of existing process models. This allows them to mine relations between activities in the available models and use the learned patterns to provide recommendations in the form of labels contained in the repository at hand [9, 18, 46, 52]. However, such approaches are restricted to the knowledge contained in the model repository available for training, which is a strong limitation and results in two key issues. First, this makes these approaches inapplicable

in situations where a process model under development entirely consists of activities that were not included in the repository's models, since the extracted knowledge cannot be used to make a recommendation in these cases. Second, existing approaches can only recommend activity labels (or, at best, combinations of label parts) that are already present in the available model repository, which leads to poor recommendations for process models that strongly differ from those in the repository.

To overcome these issues, an activity-recommendation approach should extend its recommendation capabilities to models and activities beyond those contained in the available training data. To achieve this, we propose to capture an instance of the activity-recommendation problem as a set of textual sequence-to-sequence tasks, which enables the application of transfer-learning techniques from natural language processing (NLP). Transfer learning, where a model is first pre-trained on a data-rich task to develop general-purpose abilities and then fine-tuned on a downstream task, has emerged as a powerful technique in NLP [43]. By applying such techniques to the activity-recommendation problem, we can use the general-purpose knowledge of pre-trained language models as an additional source to the problem-specific knowledge contained in a process model repository, thus enabling us to provide relevant recommendations in more settings.

We operationalize this by introducing BPART5 – a **B**usiness **P**rocess **A**ctivity **R**ecommendation approach using the pre-trained language model **T5** [43]. Using the—to this date—largest publicly available collection of process models, we evaluate the performance of BPART5 and compare it to a state-of-the-art rule-based approach for activity recommendation. The results reveal that BPART5 outperforms the rule-based approach in generating relevant recommendations in terms of semantic accuracy. Furthermore, we show that it is able to leverage the knowledge contained in the pre-trained language model to generate recommendations that go beyond the vocabulary of the model repository used for training. Specifically, BPART5 recommended numerous activities that were not present in the training data and was also able to provide better recommendations for process models consisting of unseen activities.

2 Background and Related Work

The semantics of a process model follow from the combination of two aspects [50]: the *formal semantics* of a modeling notation, which dictate how a modeled process should be executed (e.g., capturing the execution flow, including choices and concurrency), and the *label semantics* of individual model elements, which capture the meaning of the individual parts of a process model through natural language text. This dual nature opens up various opportunities for the integration of semantic technologies in process modeling and analysis, as, e.g., outlined in the overviews by Fellmann et al. [13, 14]. Research directions in this context range from the development of an ontology for business process representation [2], the automated construction of process knowledge graphs [3], and general-purpose

business process representation learning [41], to problem-specific applications of semantic technologies, such as for process model matching [30], process model similarity [11], and the focal point of our work: activity recommendation.

In the following, we consider how both aspects of process model semantics are considered by existing activity-recommendation approaches, focusing on methods exploiting formal model semantics in Section 2.1 and those that additionally incorporate label semantics in Section 2.2.

2.1 Activity recommendation based on formal semantics

Several activity-recommendation approaches use the formal semantics of process models to abstract them to directed graphs, followed by the application of graph-mining techniques to extract structural patterns from process models. Such approaches use, for example, common subgraph distance [6] or edit distance [6, 9, 32] to determine the similarity of extracted patterns from a given model repository and a process model under development. However, activity-recommendation approaches using graph-mining techniques reach their limits when applied to large repositories consisting of thousands of process models [52].

Another way of handling the formal semantics of a process models is the use of embeddings or rules. Wang et al. developed a representation-learning-based recommendation approach named RLRecommender [52], which embeds activities and relations between them. While their approach bases the recommendations of activities on one previous activity in the process model under development only, the rule-based approach from our earlier work [46] considers the entire process model under development when generating activity recommendations, which leads to better recommendations results. Based on problem-specific rule templates, our rule-based approach learns logical rules that capture activity inter-relations from the process models in a repository and applies the learned patterns to the model under development to generate activity recommendations.

Moreover, we show in an experimental study [45] that standard rule- and embedding-based knowledge graph completion methods can be applied to the activity-recommendation problem out of the box, but are not flexible enough to completely adapt to it. Compared to RLRecommender [52] and our rule-based approach [46], both specifically designed for activity recommendation, the knowledge graph completion methods performed comparably poor.

2.2 Activity recommendation based on label semantics

Several works go beyond the consideration of formal semantics, by also taking label semantics for activity recommendation into account.

In an extension of our rule-based approach [47], we generalize the information contained within activities of a repository. Through an additional analysis of the natural-language-based semantics of activities, actions and business-object patterns in the use of activity labels are leveraged to recommend also combinations of actions and business objects used in the repository.

Goldstein et al. [18] developed an approach that leverages semantic similarity of sequences in process models using the pre-trained language model Universal Sentence Encoder (USE) [7] and evaluated it both in experiments and in a user study [19]. Their approach compares an input sequence to the sequences in the training repository, recommending the label that followed the most similar training sequence. While their approach represents a first step towards the use of transfer-learning techniques from NLP for activity recommendation, it is limited to the use of a pre-trained language model without fine-tuning and recommendations of activities that exist in the given model repository for training.

3 Problem Statement

Our work is independent of a specific process modeling notation, which we achieve by representing process models as directed attributed graphs:

Definition 1 (Process model). *Let $\mathcal{L}$ be the universe of node labels and $\mathcal{T}$ be a set of node types. A process model is a tuple $M = (N, A, E, \tau, \lambda)$, where*
- *N is a set of nodes,*
- *$A \subseteq N$ is a set of activity nodes,*
- *$E \subseteq N \times N$ is a set of directed edges, such that all nodes of N are connected,*
- *$\tau : N \to \mathcal{T}$ is a function that maps a node to a type, and*
- *$\lambda : N \to \mathcal{L}$ is a function that maps a node to a label.*

This definition explicitly captures the set of activity nodes A, since these nodes are core to activity recommendation. Depending on the modeling notation, this set may contain nodes of multiple types, i.e., there exists a subset of activity types, $\mathcal{T}_A \subseteq \mathcal{T}$, such that $A = \{n \in N \mid \tau(n) \in \mathcal{T}_A\}$. For example, for BPMN, set A includes, among others, *tasks* (e.g., *reject claim*) and *events* (e.g., *claim received*), whereas *gateways* (e.g., XOR splits) are not included in A. Note that the edges in E can capture a partial order between nodes in N, to allow for concurrency and alternative executions paths in a process, such as the two choices following the XOR split in Figure 1. We use $\bullet n = \{m \in N \mid (m, n) \in E\}$ to denote the *pre-set* of a node $n \in N$, i.e., the set of all nodes preceding n in the model.

Activity recommendation targets a situation in which a process model under development contains exactly one activity node that has not yet received a label[3], such as seen in Figure 1. We refer to such a model as an *incomplete process model*:

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

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

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

4 The BPART5 Approach

This section presents our proposed BPART5 approach for activity recommendation, which uses the pre-trained sequence-to-sequence language model T5 at its core. Since T5 requires totally ordered, textual sequences as input, whereas process model nodes can be partially ordered, Section 4.1 describes how we lift activity recommendation to the format of sequence-to-sequence tasks. In Section 4.2, we describe how we use this procedure to fine-tune T5 for activity recommendation based on process knowledge encoded in a process model repository. Finally, Section 4.3 describes how we use our fine-tuned T5 model to solve instances of the activity-recommendation problem, for which we first solve multiple sequence-to-sequence tasks, whose results we then aggregate in order to return one or more label recommendations.

4.1 Sequence-to-sequence tasks for activity recommendation

Sequence-to-sequence tasks are concerned with finding a model that maps a sequence of inputs $(x_1, \ldots, x_T)$ to a sequence of outputs $(y_1, \ldots, y_{T'})$, where the output length T' is unknown a priori and may differ from the input length T [49]. A classic example of a sequence-to-sequence problem from the NLP field is machine translation, where the *input sequence* is given by text in a source language and the *output sequence* is the translated text in a target language.

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

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

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

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

4.2 Fine-tuning T5 for activity recommendation

For our approach, we use the sequence-to-sequence language model T5, which is based on the transformer architecture introduced by Vaswani et al. [51]. T5 is

pre-trained on a set of unsupervised and supervised tasks, where each task is converted into a text-to-text format. We fine-tune T5 for activity recommendation by extracting a large number of sequence-to-sequence tasks from the models in an available process model repository $\mathcal{M}$. Specifically, for each model $M \in \mathcal{M}$, we extract all possible sequences of a certain length l that end in an activity node, i.e., $(n_1, \ldots, n_l)$ with $n_l \in A$. Afterwards, we apply verbalization on this node sequence to get the textual input sequence, as described in Section 4.1, whereas the output sequence corresponds to the label of n_l.

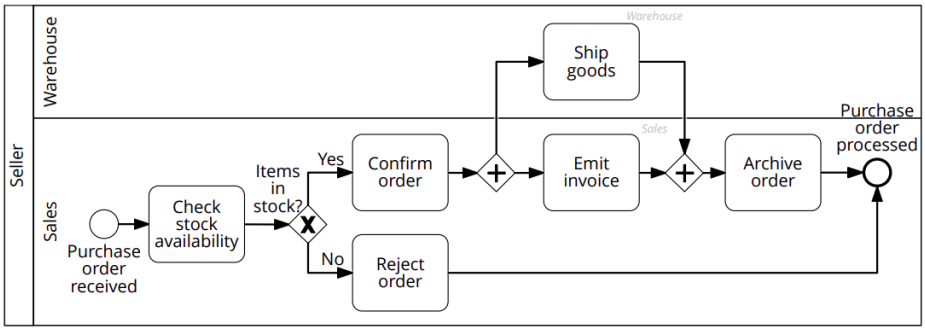


Fig. 2: An order-to-cash process model

As an example, consider the exemplary training process model depicted in Figure 2. Setting $l = 4$, the model contains nine sequences of length four that end in an activity node. After verbalization, these result in the following textual *(input,output)* sequences we can use to fine-tune T5:

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

4.3 Generating label recommendations

Given an incomplete process model M_I with an unlabeled activity node $\hat{n}$, for which we want to provide label recommendations, we first extract all sequences of length l that end in $\hat{n}$. We then verbalize all these sequences and feed the resulting input sequences as sequence-to-sequence tasks into our fine-tuned T5

model. For instance, for the example of Figure 1, this results in the two input sequences described earlier when using $l = 4$, which are:

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

Output sequence generation. We solve the individual sequence-to-sequence tasks by feeding each input sequence into our fine-tuned T5 model, generating 10 alternative output sequences, i.e., 10 possible labels, per input. To do this, we use *beam search* [20] as a decoding method, with beam width $w = 10$. The beam search algorithm uses conditional probabilities to track the w most likely output sequences at each generation step.

A downside of the beam search algorithm is that it can lead to output sequences that repeat words or even short sequences, i.e., *n-grams*. Following activity labeling convention [36, 37, 42], we favor the suggestion of short labels that do not contain any recurring terms. For example, rather than suggesting labels such as "*check passport and check visa*", our approach would suggest the non-repetitive alternative: "*check passport and visa*". To achieve this, we apply n-gram penalties [26, 40] during beam search. Specifically, we penalize the repetition of n-grams of any size (including single words) by setting the probability of next words that are already included in the output sequence to zero.

Tables 1a and 1b show the alternative output sequences (and probabilities) that the fine-tuned T5 model generates for input sequences I_1 and I_2.

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

(a) Output sequences and probabilities based on I_1

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

(b) Output sequences and probabilities based on I_2

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

(c) Final list of label recommendations

Table 1: From two lists of T5-generated output sequences to one list of label recommendations using maximum strategy

Result aggregation. Finally, we aggregate the different lists of output sequences, obtained by using beam search to solve individual sequence-to-sequence tasks, in order to end up with a single list of recommended activity labels. To do this, we aggregate the contents of the lists using a *maximum strategy*, which is commonly used by rule-based methods to rank proposed entities according to the different confidence values of the rules that suggested them [35, 38, 46].

To apply the maximum strategy, we establish an aggregated recommendation list, sorted according to the maximal probability score that a recommended

label received. For instance, the *notify about outcome* label receives a score of 0.64, from the output sequences generated for I_1, although the label also appears in I_2's list, yet with a score of 0.42. If two recommendations have the same maximum probability, we sort them based on their second-highest probability, if available. Analogously, if two recommendations share maximum and second-highest probability, we continue until we find a probability that makes a difference. In the end, BPART5 thus provides a list of ten label recommendations for the unlabeled node $\hat{n}$ that are the most probable candidates, according to the sequences contained in the model under development, the fine-tuned T5 model, and the maximum aggregation strategy.

The final list obtained for the running example is shown in Table 1c. Notably, the top five recommendations represent alternative manners to inform an applicant, e.g., in the form of *notify about outcome*, *send notification*, or *send email to customer*, which indeed appears to be the appropriate process step given the current state of the process model under development.

5 Experimental Study

In our experimental study[4], we evaluate the performance of BPART5 and compare it to the state-of-the-art approach from our earlier work [47]. We first describe the used dataset (Section 5.1), the experimental setup (Section 5.2) and the employed metrics (Section 5.3). Finally, we present the results of our experiments in Section 5.4.

5.1 Dataset

For our experimental evaluation, we employ the SAP Signavio Academic Models (SAP-SAM) [24] dataset. SAP-SAM is the—to this date—largest publicly available collection of business process models, which consists of over one million process models in different modeling notations and languages.

SAP-SAM contains models of varying complexity and quality. For our experiments, we aim to use a subset of the dataset where certain process modeling standards, as proposed in established work [36], are met. Following the recommendations for usage of the dataset provided by its publishers [48], we thus filter the dataset as follows. We select all BPMN 2.0 models in English with three to 30 nodes (including gateways), where each activity label is composed of at least three non-empty characters. In addition, we exclude default vendor-provided example models included in SAP-SAM[5]. Note that for filtering and pre-processing the models of SAP-SAM we apply label cleaning, in which we turn non-alphanumeric characters into whitespace, handle special cases as line

[4] We provide the source code of the employed implementation under this link: https://github.com/disola/bpart5.

[5] SAP-SAM contains a high number of vendor-provided example models. The publishers of the dataset recommend sorting them out as they negatively affect the diversity of the dataset.

breaks, change all letters into lowercase and delete unnecessary whitespace. This results in a filtered dataset, which consists of 77,239 process models containing an average of 14.7 nodes (median: 13) and a total of 241,283 unique node labels with an average length of 26.5 characters (median: 24).

5.2 Experimental setup

Evaluated approaches. In our experiments, we choose a sequence length l=4 for our BPART5 approach, i.e., we extract sequences of length four that end in node $\hat{n}$. This choice follows findings from prior research [18], which showed that considering three previous nodes for activity recommendation works well across different datasets.

We compare BPART5 to the rule-based approach from our earlier work [47], which has been shown to outperform several other recommendation approaches [22, 23, 45, 46, 52]. We also wanted to include the approach from Goldstein et al. [18], which is based on semantic similarity and the universal sentence encoder, but the source code of their approach is not available online, and also the authors did not provide us their source code after requesting it.

Implementation. Our implementation of BPART5 and the metrics uses the Huggingface library [53]. For tokenizing sequences, we used the *fast* T5 tokenizer backed by HuggingFace's tokenizer library, which is based on Unigram [27] in conjunction with SentencePiece [28]. We fine-tuned T5-Small[6] employing the Adam algorithm [25] with weight decay fix as introduced in [33] and constant learning rate 0.0003. Moreover, we set the batch size to 128 and trained the model until the validation loss did not improve for 20,000 steps. The experiments were carried out using two Nvidia RTX A6000 GPUs.

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

Evaluation procedure. The testing split consists of process models, which the modelers have considered finished. However, we want to evaluate the ability of approaches to generate activity recommendations for process models under development, i.e., for incomplete process models. Given a finished process model, we thus simulate different stages of the model construction, which is a common practice for the evaluation of activity-recommendation approaches [18, 47]. We use a breadth-first search inspired simulation technique, where we first select an activity node $\hat{n}$ from a finished process model as the one for which we want to recommend labels, and hide the label of $\hat{n}$. Then, we determine the shortest sequence from a source node (a node without preset) to $\hat{n}$ and denote the length of this sequence by s. Subsequently, we remove all nodes that are not contained in

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

a sequence of length s starting from a source node and retain all other nodes and edges between them. The remaining process model is treated as the intermediate result of a model construction, and the task to recommend labels for the selected node $\hat{n}$ based on the remaining process model represents one evaluation case. For each process model in the testing split, we generate several such evaluation cases by carrying out this procedure for each activity node, where the shortest sequence to a source node has the minimum length three. This leads to a total of 36.143 evaluation cases, i.e., activity-recommendation tasks.

5.3 Metrics

We assess the performance of the activity-recommendation approaches using four different metrics, namely Hits@k, BLEU@k, METEOR@k and Cos@k:

Hits@k. First, we report on the standard *hit rate* Hits@k [21], which is frequently used in the context of activity recommendation [47, 52] as well as for other recommendation applications, where it is sufficient that the recommendation list contains one item that the user selects [21]. Hits@k captures the proportion of hits in the top-k recommendations. In other words, it is the proportion of evaluation cases, where the ground truth, i.e., the actually used activity in a process model, is one of the k recommendations.

BLEU@k. The BLEU [39] metric is typically used in machine translation, where a candidate translation is compared to one or more reference translations. In the context of activity recommendation, BLEU basically compares n-grams of the recommended activity with n-grams of the ground-truth activity and calculates a modified precision based on n-gram matches. Similarly to the standard hit rate Hits@k, we can define the BLEU@k hit rate as the maximum BLEU score of the top-k recommendations. This results in a single score for a recommendation list of length k instead of the k BLEU scores of each recommendation in the list.

METEOR@k. Just as BLEU, the METEOR [4] metric is also typically used to assess the quality of machine translations[7]. In our context, METEOR evaluates the quality of an activity recommendation based on unigram matches with the ground-truth activity. In addition to exact matches, it also considers semantic similarity in the form of stemmed matches and wordnet-based synonym matches. Analogously to BLEU@k, the meteor hit rate METEOR@k is given by the maximum METEOR score of the first k recommendations.

Cos@k. The cosine similarity [31] requires representations of the activity recommendation and the ground-truth activity as embeddings, enabling the calculation of the similarity of the two activities in the form of the cosine similarity of their embeddings. Cos@k is the maximum cosine similarity score of the top-k

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

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

Table 2: Values of BLEU, METEOR and cosine similarity for different recommendations given the actual used activity *Notify about outcome*

recommendations. In our evaluation, we use the universal sentence encoder [7] to generate the embeddings of activities, which allows Cos@k to consider the semantic similarity of the recommendations and the actual used activity.

Metric relevance. In a recent user study, Goldstein et al. [19] showed that the employed metrics strongly correlate with experts' ratings of activity recommendations. Thus, they can be confidently used to measure the quality of activity recommendations.

Employing four metrics allows us to gain different kinds of insights. The standard hit rate, Hits@k, is a strict metric in the sense that a hit is realized only if a recommendation and the ground truth are an exact match. If, for example, a recommendation is given by *Notify about outcome* while *Inform about outcome* is used in the test process model, then the recommendation would not count as a hit and the recommendation approach would be considered unsuccessful in this case. However, given its similarity and the fact that there are several possible manners of describing an activity with a label, the recommendation would still be highly useful for the modeler. In this sense, the semantic hit rates BLEU@k, METEOR@k and Cos@k are more practice-oriented. By taking the similarity of recommendations to the ground truth into account, they measure the semantic accuracy of the recommendations. The values of the semantic hit rates are always bigger or equal to the Hits@k values. To illustrate the different levels of similarity that are measured by the three semantic hit rates, Table 2 shows the values of BLEU, METEOR and cosine similarity for four exemplary recommendations from the list in Table 1c, given that the actual used activity label is *Notify about outcome*.

5.4 Evaluation results

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

Overall results. The overall results of our experiments, in which we compare BPART5 to the rule-based recommendation approach from our earlier work [47], are shown in Table 3.[8] Considering a recommendation list of length k=10, the rule-based approach outperforms BPART5 by 11% in terms of the rigid hit rate

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

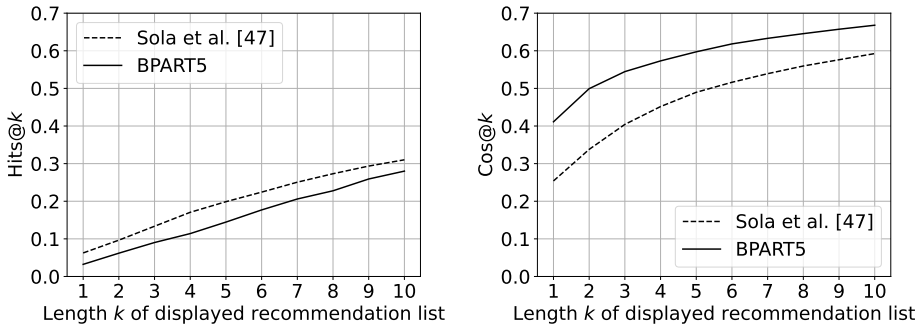
List size	Approach	Hits@k	BLEU@k	METEOR@k	cos@k
$k = 10$	Sola et al. [47]	**0.3102**	0.3358	0.4149	0.5925
	BPART5	0.2800	**0.3876**	**0.5154**	**0.6679**
$k = 1$	Sola et al. [47]	**0.0625**	0.0714	0.1049	0.2539
	BPART5	0.0322	**0.1179**	**0.2269**	**0.4112**

Table 3: Results of the evaluated approaches

Hits@10. However, when considering the semantic hit rates, which recognize that activity recommendations that are semantically similar to the ground-truth activity are also useful for modelers, then BPART5 turns out to be superior. It outperforms the rule-based approach by 15%, 24%, and 12% in BLEU@10, METEOR@10, and Cos@10, respectively. Turning to the hit rates for k=1, i.e., the hit rates of the top recommendation of each list, it is equally apparent that the rule-based approach performs better in terms of the standard hit rate, whereas BPART5 achieves better results in terms of the semantic hit rates. Thus, the results indicate that the rule-based approach is more accurate in giving recommendations that correspond exactly to the ground truth. BPART5 is better in generating recommendations that are not an exact match but have a high semantic similarity to the ground truth, though, which means that BPART5 provides in general more relevant recommendations.

Regarding the ranking of suitable activities within a recommendation list, Figure 3 shows the courses of the standard hit rate Hits@k and the cosine hit rate Cos@k for recommendation lists of lengths k=1 to k=10 as examples. The curves of BLEU@k and METEOR@k, which are not depicted here, are comparable to the curves of Hits@k and Cos@k, respectively. Figure 3a shows that the lines from the Hits@1 to the Hits@10 values are rather straight. The likelihood of finding a—in terms of this metric—suitable recommendation thus increases linearly with each additional activity in the recommendation list. In the case of Cos@k (Figure 3b), the curves rise more steeply for smaller lengths of the recommendation list, which indicates that both approaches are able to rank recommendations that are semantically similar to the ground truth on the first positions of the recommendation list.

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



(a) Hit@k (b) Cos@k

Fig. 3: Results for different lengths of the displayed recommendation list

The difference in the unique numbers of generated label recommendations indicates that BPART5 achieves a higher diversity of recommended labels, while the rule-based approach is dependent on the knowledge in the process models used for training and thus more limited in its recommendations. From the percentages of newly generated labels, we can conclude that BPART5 is able to leverage the knowledge contained in the pre-trained language model and recommend activity labels that go beyond the vocabulary of the process models in the training set. On the one hand, BPART5 performs worse in terms of hit rate for this reason, on the other hand, this leads to less dependency on the given process models used for training and therefore a higher semantic accuracy of BPART5.

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

Approach	Hits@10	BLEU@10	METEOR@10	cos@10
Sola et al. [47]	0.0070	0.0079	0.0628	0.2892
BPART5	**0.0232**	**0.0963**	**0.2112**	**0.4452**

Table 4: Results on the subset of evaluation cases with only unseen labels

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

The results of the study are presented in Table 4. While the absolute numbers of the metrics on this subset are naturally low, due to the challenging nature

of the cases, the results show that BPART5 clearly outperforms the rule-based approach on the subset in terms of all metrics. Although the rule-based approach is restricted to the knowledge in the process model repository, it is able to generate a few useful recommendations in the form of default label recommendations. Specifically, it recommends the ten most often used activities of the repository whenever none of the rules it learned matches the process model under development, as is applicable to the cases at hand. Nevertheless, the difference between the results of both approaches is much larger than on the complete set of evaluation cases. This demonstrates that BPART5 is not only the better approach in terms of semantic accuracy in general, but also the approach of choice when it comes to recommendations for situations that differ considerably from the available training data.

6 Conclusion and Future Work

In this paper, we presented the BPART5 approach for activity-recommendation, which uses the formal and natural language semantics contained in process models to enable the application of pre-trained sequence-to-sequence language models. Our experiments showed that BPART5 outperforms a state-of-the-art rule-based approach in terms of semantic accuracy, which means that it provides in general more relevant recommendations. We also demonstrated that BPART5 is able to deal with input that differs considerably from what it has seen before, and can even generate label recommendations that go beyond the vocabulary of the model repository used for training. In this sense, it is the first activity-recommendation approach that fully leverages pre-trained models from the NLP domain.

In future work, we would like to address the three main limitations of our work. First, BPART5 requires node sequences of length l-1 (3 in our experiments) for providing recommendations. In future work, we want to investigate possible ways to use not only sequences of a particular length but arbitrary sequences of the process model under development to generate activity recommendations. Second, BPART5 does not yet incorporate information on task types or the organizational perspective, which we aim to include by extending BPART's verbalization procedure. For example, we could consider information about the organization, which owns the process model (pool label), or about the roles, systems or organization's departments that execute the process (lane labels). Additionally, we could also differentiate between gateway splits and gateway joins, or take edge labels into account. Third, our evaluation used artificial recommendation scenarios, whereas in the future we will study the perceived usefulness of our recommendations. In addition, similarly to the work by Meilicke et al. [34], who constructed a method to combine the outcomes of rule-based and latent approaches for knowledge base completion in a post-processing step, we also aim to develop an ensemble method that combines our rule-based approach [47] and BPART5 to generate better activity recommendations.

References

1. Abran, A., Moore, J.W., Bourque, P., Dupuis, R., Tripp, L.: Software engineering body of knowledge. IEEE Computer Society, Angela Burgess (2004)
2. Annane, A., Aussenac-Gilles, N., Kamel, M.: Bbo: Bpmn 2.0 based ontology for business process representation. In: 20th European Conference on Knowledge Management (ECKM 2019). vol. 1, pp. 49–59 (2019)
3. Bachhofner, S., Kiesling, E., Revoredo, K., Waibel, P., Polleres, A.: Automated process knowledge graph construction from bpmn models. In: Strauss, C., Cuzzocrea, A., Kotsis, G., Tjoa, A.M., Khalil, I. (eds.) Database and Expert Systems Applications. pp. 32–47. Springer International Publishing, Cham (2022)
4. Banerjee, S., Lavie, A.: Meteor: An automatic metric for mt evaluation with improved correlation with human judgments. In: Proceedings of the ACL workshop on intrinsic and extrinsic evaluation measures for machine translation and/or summarization. pp. 65–72 (2005)
5. Boehm, B.W., Papaccio, P.N.: Understanding and controlling software costs. IEEE transactions on software engineering **14**(10), 1462–1477 (1988)
6. 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)
7. Cer, D., Yang, Y., Kong, S.y., Hua, N., Limtiaco, N., John, R.S., Constant, N., Guajardo-Cespedes, M., Yuan, S., Tar, C., Sung, Y.H., Strope, B., Kurzweil, R.: Universal sentence encoder (2018)
8. Davies, I., Green, P., Rosemann, M., Indulska, M., Gallo, S.: How do practitioners use conceptual modeling in practice? Data & Knowledge Engineering **58**(3), 358–380 (2006)
9. 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)
10. Dumas, M., La Rosa, M., Mendling, J., Reijers, H.A.: Fundamentals of Business Process Management. Springer, Berlin, 2nd edn. (2018)
11. Ehrig, M., Koschmider, A., Oberweis, A.: Measuring similarity between semantic business process models. In: APCCM. vol. 7, pp. 71–80 (2007)
12. Fahland, D., Favre, C., Koehler, J., Lohmann, N., Völzer, H., Wolf, K.: Analysis on demand: Instantaneous soundness checking of industrial business process models. Data & Knowledge Engineering **70**(5), 448–466 (2011)
13. Fellmann, M., Delfmann, P., Koschmider, A., Laue, R., Leopold, H., Schoknecht, A.: Semantic technology in business process modeling and analysis. part 1: Matching, modeling support, correctness and compliance. EMISA Forum **35**, 15–31 (2015)
14. Fellmann, M., Delfmann, P., Koschmider, A., Laue, R., Leopold, H., Schoknecht, A.: Semantic technology in business process modeling and analysis. part 2: Domain patterns and (semantic) process model elicitation. EMISA Forum **35**(2), 12–23 (2015)
15. Fellmann, M., Zarvic, N., Metzger, D., Koschmider, A.: Requirements catalog for business process modeling recommender systems. In: WI. pp. 393–407 (2015)
16. 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)
17. Friedrich, F., Mendling, J., Puhlmann, F.: Process model generation from natural language text. In: CAiSE. pp. 482–496. Springer (2011)

18. Goldstein, M., González-Álvarez, C.: Augmenting modelers with semantic autocompletion of processes. In: BPM Forum. pp. 20–36. Springer (2021)
19. Goldstein, M., González-Álvarez, C.: Evaluating semantic autocompletion of business processes with domain experts. In: ASE. pp. 1116–1120 (2021)
20. Graves, A.: Sequence transduction with recurrent neural networks. arXiv preprint arXiv:1211.3711 (2012)
21. Gunawardana, A., Shani, G., Yogev, S.: Evaluating Recommender Systems, pp. 547–601. Springer US, New York, NY (2022)
22. Jannach, D., Fischer, S.: Recommendation-based modeling support for data mining processes. In: RecSys. pp. 337–340 (2014)
23. Jannach, D., Jugovac, M., Lerche, L.: Supporting the design of machine learning workflows with a recommendation system. ACM TiiS **6**(1), 1–35 (2016)
24. Kampik, T., Warmuth, C., Sola, D., Schäfer, B., Axworthy, L., Ivarsson, E., Ouda, K., Eickhoff, D.: Sap signavio academic models (Aug 2022). https://doi.org/10.5281/zenodo.7012043
25. Kingma, D.P., Ba, J.: Adam: A method for stochastic optimization. In: ICLR (2015)
26. Klein, G., Kim, Y., Deng, Y., Senellart, J., Rush, A.M.: Opennmt: Open-source toolkit for neural machine translation. In: Proceedings of ACL 2017, System Demonstrations. pp. 67–72 (2017)
27. Kudo, T.: Subword regularization: Improving neural network translation models with multiple subword candidates. In: Gurevych, I., Miyao, Y. (eds.) ACL (1). pp. 66–75. Association for Computational Linguistics (2018)
28. Kudo, T., Richardson, J.: Sentencepiece: A simple and language independent subword tokenizer and detokenizer for neural text processing. CoRR **abs/1808.06226** (2018)
29. Leoni, M.d., Felli, P., Montali, M.: A holistic approach for soundness verification of decision-aware process models. In: International Conference on Conceptual Modeling. pp. 219–235. Springer (2018)
30. Leopold, H., Niepert, M., Weidlich, M., Mendling, J., Dijkman, R.M., Stuckenschmidt, H.: Probabilistic optimization of semantic process model matching. In: BPM. Lecture Notes in Computer Science, vol. 7481, pp. 319–334. Springer (2012)
31. Li, B., Han, L.: Distance weighted cosine similarity measure for text classification. In: International conference on intelligent data engineering and automated learning. pp. 611–618. Springer (2013)
32. 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)
33. Loshchilov, I., Hutter, F.: Decoupled weight decay regularization. In: International Conference on Learning Representations (2018)
34. Meilicke, C., Betz, P., Stuckenschmidt, H.: Why a naive way to combine symbolic and latent knowledge base completion works surprisingly well. In: 3rd Conference on Automated Knowledge Base Construction (2021)
35. 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)
36. Mendling, J., Reijers, H.A., van der Aalst, W.M.: Seven process modeling guidelines (7pmg). Information and Software Technology **52**(2), 127–136 (2010)
37. Mendling, J., Reijers, H.A., Recker, J.: Activity labeling in process modeling: Empirical insights and recommendations. Inf. Syst. **35**(4), 467–482 (2010)

38. Ott, S., Meilicke, C., Samwald, M.: Safran: An interpretable, rule-based link prediction method outperforming embedding models. In: 3rd Conference on Automated Knowledge Base Construction (2021)
39. Papineni, K., Roukos, S., Ward, T., Zhu, W.J.: Bleu: a method for automatic evaluation of machine translation. In: Proceedings of the 40th annual meeting of the Association for Computational Linguistics. pp. 311–318 (2002)
40. Paulus, R., Xiong, C., Socher, R.: A deep reinforced model for abstractive summarization. In: International Conference on Learning Representations (2018)
41. Pfeiffer, P., Lahann, J., Fettke, P.: Multivariate business process representation learning utilizing gramian angular fields and convolutional neural networks. In: BPM. pp. 327–344. Springer (2021)
42. Pittke, F., Leopold, H., Mendling, J.: Automatic detection and resolution of lexical ambiguity in process models. IEEE Transactions on Software Engineering **41**(6), 526–544 (2015)
43. Raffel, C., Shazeer, N., Roberts, A., Lee, K., Narang, S., Matena, M., Zhou, Y., Li, W., Liu, P.J.: Exploring the limits of transfer learning with a unified text-to-text transformer. Journal of Machine Learning Research **21**(140), 1–67 (2020)
44. Rosemann, M.: Potential pitfalls of process modeling: part a. Business Process Management Journal (2006)
45. 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: AI4BPM. Springer (2021)
46. Sola, D., Meilicke, C., Van der Aa, H., Stuckenschmidt, H.: A rule-based recommendation approach for business process modeling. In: CAiSE. Springer (2021)
47. 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** (2022)
48. Sola, D., Warmuth, C., Schäfer, B., Badakhshan, P., Rehse, J.R., Kampik, T.: Sap signavio academic models: A large process model dataset. arXiv e-prints pp. arXiv–2208 (2022)
49. Sutskever, I., Vinyals, O., Le, Q.V.: Sequence to sequence learning with neural networks. In: NIPS (2014)
50. Thomas, O., Fellmann, M.: Semantic process modeling - design and implementation of an ontology-based representation of business processes. Business & Information Systems Engineering **1**(6), 438–451 (2009)
51. Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A.N., Kaiser, Ł., Polosukhin, I.: Attention is all you need. Advances in neural information processing systems **30** (2017)
52. 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)
53. Wolf, T., Debut, L., Sanh, V., Chaumond, J., Delangue, C., Moi, A., Cistac, P., Rault, T., Louf, R., Funtowicz, M., Brew, J.: Huggingface's transformers: State-of-the-art natural language processing. CoRR **abs/1910.03771** (2019)