Ranking-Based Case Retrieval with Graph Neural Networks in Process-Oriented Case-Based Reasoning

Maximilian Hoffmann[1,2], Ralph Bergmann[1,2]
[1] Artificial Intelligence and Intelligent Information Systems, University of Trier, 54296 Trier, Germany
{hoffmannm,bergmann}@uni-trier.de
[2] German Research Center for Artificial Intelligence (DFKI)
Branch University of Trier, Behringstraße 21, 54296 Trier, Germany
{maximilian.hoffmann,ralph.bergmann}@dfki.de

Abstract

In *Process-Oriented Case-Based Reasoning (POCBR)*, experiential knowledge from previous problem-solving situations is retrieved from a case base to be reused for upcoming problems. The task of retrieval is approached in previous work by using *Graph Neural Networks (GNNs)* to learn workflow similarities which are, in turn, used to find similar workflows w.r.t. a query workflow. This paper is motivated by the fact that these GNNs are mostly used for predicting the similarity between two workflows (query and case), while the retrieval in CBR is only concerned with the ranking of the most similar workflows from the case base w.r.t. the query. Thus, we propose a novel approach to extend the GNN-based workflow retrieval by a *Learning-to-Rank (LTR)* component where rankings instead of similarities between cases are predicted. The main contribution of this paper addresses the changes to the GNNs from previous work, such that their model architecture predicts pairwise preferences between cases w.r.t. a query and that they can be trained using labeled preference data. In order to transform these preferences into a case ranking, we also describe *rank aggregation* methods with different levels of computational complexity. The experimental evaluation compares different models for predicting similarities and rankings in case retrieval scenarios. The results indicate the potential of our ranking-based approach in significantly improving retrieval quality with only small impacts on the performance.

1 Introduction

Hybrid approaches of *Case-Based Reasoning (CBR)* (Aamodt and Plaza 1994) and *Deep Learning (DL)* are becoming more and more popular across many CBR research groups, e. g., (Amin et al. 2020; Ye, Leake, and Crandall 2022; Hoffmann and Bergmann 2022). The common goal is to support or implement different parts of CBR applications with appropriate DL methods (Leake and Crandall 2020). Thereby, most of these efforts are focused on *case retrieval*, i. e., a case base of best-practice problem-solution-pairs is queried to find solution candidates for a given problem based on the similarity between the new problem and past problems. Supporting this task with DL methods is commonly approached by automatically learning similarity measures, for instance, to predict the similarity between the query and cases of the case base (Amin et al. 2020; Mathisen, Bach, and Aamodt 2021; Leake and Ye 2021). In this process, the role of the similarity measure is to control the retrieval process by declaring the most similar cases of the case base as the most useful solution candidates to the query. This leaves the concrete similarity values mainly as a criterion for *case ranking*. Especially when using DL-based similarity measures, predicting similarities instead of rankings might lead to imprecise retrieval results, since small similarity prediction errors can have great effects on the final ranking of the cases. A more intuitive approach in this scenario would be to bring the model predictions closer to the underlying task by predicting *pairwise preferences* such as "*case A is more relevant to the query than case B*". This reformulation has two advantages: First, the number of training examples increases as each pair of cases from the case base combined with a query case is one training example. When predicting similarities, on the other hand, a training example consists of the query and a single case, which effectively squares the number of training examples when training with pairwise preferences. Second, it redefines the learning task as a binary classification which utilizes different metrics, loss functions, and training parameters compared to a regression. This enables a new way of training the models, which might lead to more accurate predictions or shorter training times.

The described technique is part of the *Learning to Rank (LTR)* methodology (Liu 2011) that is also known as preference learning (Fürnkranz and Hüllermeier 2010). LTR deals with ranking items according to a given query by means of machine learning methods. It covers crucial aspects of the ranking procedure, such as suitable loss functions and training procedures. The use of LTR methods in CBR is not thoroughly investigated, but its applicability is indicated by the shared goal of ranking cases according to a query (Bichindaritz 2006) as well as the way experts think about cases which much more resembles preferences than similarity values (Hüllermeier and Schlegel 2011). Therefore, we investigate the potential of LTR for similarity-based retrieval in this paper. We use the context of *Process-Oriented Case-Based Reasoning (POCBR)* (Minor, Montani, and Recio-García 2014). POCBR is a subfield of CBR that deals with procedural knowledge, such as semantic graphs (Bergmann and Gil 2014). We discuss existing DL-based methods w. r. t. their contribution to LTR and integrate new LTR principles into the *Graph Neural Networks (GNNs)* from literature (Hoffmann et al. 2020; Hoffmann and Bergmann 2022) with the goal of improving the quality and performance of these models for the task of case retrieval in POCBR. The remainder of the paper is structured as follows: Section 2 shows foundations of the used semantic graph representation and gives an introduction to LTR before presenting related work. Further, Sect. 3 presents the concept for integrating LTR principles into GNNs for similarity assessment in POCBR. Section 4 presents the experimental evaluation before Sect. 5 concludes the paper and shows directions of future work.

2 Foundations and Related Work

The foundations include the semantic workflow representation that is used in the concept and the experiment evaluation, as well as the similarity assessment between pairs of these workflows. Additionally, the basic strategies of LTR are introduced, and related work is examined.

Semantic Workflow Representation and Similarity Assessment

We represent all workflows as semantically annotated directed graphs referred to as *NEST* graphs, introduced by Bergmann and Gil [2014]. A *NEST* graph is defined as a quadruple $W = (N, E, S, T)$ that is composed of a set of nodes N and a set of edges $E \subseteq N \times N$. Each node and each edge has a specific type from Ω that is indicated by the function $T : N \cup E \to \Omega$. Additionally, the function $S : N \cup E \to \Sigma$ assigns a semantic description from Σ (*semantic metadata language*, e. g., an ontology) to nodes and edges. Whereas nodes and edges represent the structure of each workflow, types and semantic descriptions model additional semantic information. Figure 1 shows an exemplary *NEST* graph that represents a cooking recipe of a mayo-gouda sandwich. The sandwich is prepared by executing the cooking steps `coat` and `layer` (represented as task nodes) with the ingredients `mayo`, `baguette`, `sandwich dish`, and `gouda` (represented as data nodes). All components are connected by edges indicating relations, e. g., `coat` consumes `mayo`. Semantic information belonging to the workflow components is further specified by using semantic annotations for nodes and edges. For instance, the semantic description of the task node `coat` provides the information that a spoon and a baguette knife are needed to execute the task and that the estimated time is two minutes.

Since NEST graphs are used in the context of POCBR, Bergmann and Gil [2014] also propose a semantic similarity measure between two *NEST* graphs, i. e., a query workflow *QW* and a case workflow *CW*. The similarity is determined based on the local-global principle such that a global similarity, i. e., the similarity between two graphs, is composed of local similarities, i. e., the pairwise similarities of nodes and edges. Nodes with the same type can be mapped onto each other, and their similarity corresponds to the similarity

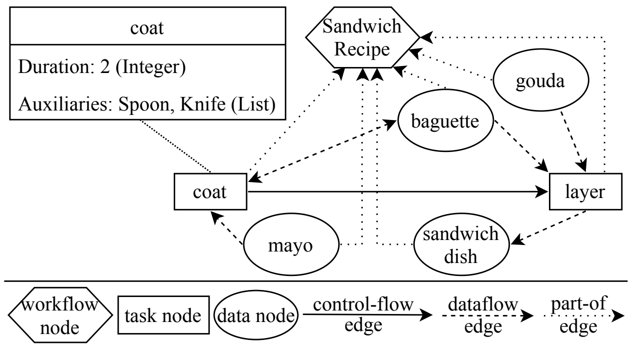

Figure 1: Exemplary Cooking Recipe represented as NEST Graph

of their semantic descriptions. The similarity between two edges with identical types does not only consider the similarity of the semantic descriptions of the edges, but in addition, the similarity of the connected nodes, as well. A mapping is, hence, admissible if all mapped nodes as well as all mapped edges with their source and destination nodes are of the same type. When assuming that the domain defines a similarity model with a similarity measure $sim_\Sigma : \Sigma \times \Sigma \to [0, 1]$ for all components of the semantic description, the global similarity of the two workflows $sim(QW, CW)$ is calculated by finding an injective partial mapping m that maximizes $sim_m(QW, CW)$.

$$sim(QW, CW) = max\{sim_m(QW, CW) \mid \text{admissible mapping } m\}$$

The complex process of finding a mapping that maximizes the global similarity between a query *QW* and a single case *CW* is tackled by an A* search algorithm (see Bergmann and Gil [2014] for more details). However, A* search is usually time-consuming and can lead to long retrieval times (Zeyen and Bergmann 2020; Klein, Malburg, and Bergmann 2019; Hoffmann et al. 2022) which motivates using automatic learning methods for the task of case retrieval.

Learning to Rank

In LTR, a ranked list of items is determined based on the relevance of these items w. r. t. a query (Liu 2011). A well-known application scenario is information retrieval, where the most relevant documents regarding some query are ranked and returned to a user. LTR approaches are commonly divided into three categories based on the way rankings are determined, i. e., pointwise, pairwise, listwise LTR (Liu 2011): *Pointwise LTR* approaches use existing metrics to compute a relevance score for each item. For instance, retrieving similar cases based on a query in a case retrieval in CBR could be associated to this category. These methods have the advantage that they can be developed and implemented rather quickly due to the metrics already existing, but the quality might not be sufficient, since the used metrics were probably not intended to be used in an LTR context. Furthermore, *pairwise LTR* or *label ranking* (Fürnkranz and Hüllermeier 2010) approaches determine the ranked list of items by evaluating preferences between pairs of items w. r. t. the query. A preference in this sense states whether one or the other item of a pair is more relevant to the query. To get a ranked list of items, *rank aggregation* is used and transforms the pairwise preferences. Pairwise methods are, compared to pointwise approaches, harder to conceptualize and implement due to the two involved components, but could probably lead to better overall results due to their focus on predicting actual rankings. Although the proposed approach only covers pointwise and pairwise methods, *listwise LTR*, also known as *object ranking* (Fürnkranz and Hüllermeier 2010), is still explained for the sake of completeness. This category of approaches is the most straightforward, as it returns a complete ranked list of items according to the query. There is no need to derive the ranked list of items from different metrics as in the other categories. Thus, listwise LTR methods promise better results than other methods due to their direct prediction of ranked lists but, in turn, are also generally harder to implement and train.

Related Work

Related work in the context of CBR or POCBR dealing with rankings or preferences explicitly is rather scarce. One of the earliest approaches in this regard is proposed by Avesani et al. [2003]. They describe an interactive approach where users set preferences to help a system configure a model to answer queries based on the user's preferences. Brinker and Hüllermeier [2007] discuss pairwise ranking in the CBR context conceptually. They mainly focus on implementing a case retrieval when given preferences as labels. Hüllermeier and Schlegel [2011] extend this concept and propose a methodological framework for problem-solving in CBR with preferences. It describes the representation of experiential knowledge in the form of pairwise preferences between cases and case retrieval in this setup. While the aforementioned approaches are primarily concerned with a conceptualization of rankings in the CBR methodology, there are also approaches that use rankings in more application-oriented scenarios. For instance, Li and Sun [2011] use rankings and CBR to compute similarities between cases in finance applications. Glockner and Weis [2012] determine case rankings with a combination of machine learning and CBR for question answering. Bichindaritz [2006] combines CBR and information retrieval for scenarios from biomedicine. Our approach is novel regarding related work as it is a combination of the POCBR context, domain independence, and the use of DL methods to predict preferences.

3 Similarity-Based Retrieval by Using Learning to Rank Methods

Case retrieval with GNNs in POCBR applications returns the most similar cases according to the similarity predictions of the used neural networks. We propose an alternative approach to case retrieval that predicts preferences of cases with GNNs to form the retrieval result. The following sections, first, place GNN-based retrieval in the context of pointwise LTR and, second, describe the approach of retrieval by predicting pairwise case preferences.

Pointwise Ranking with GNNs

For the purpose of similarity-based retrieval in POCBR, Hoffmann et al. [2020] adapt two Siamese GNNs which are called *Graph Embedding Model (GEM)* and the *Graph Matching Network (GMN)* (see Fig. 2). Both models learn embeddings of semantic graphs by processing the nodes and the edges between those nodes, incl. their types and semantic annotations. These embeddings are then further aggregated to a single pairwise similarity value by using cosine vector similarity or a *Multi-Layer Perceptron (MLP)*. Thereby, both

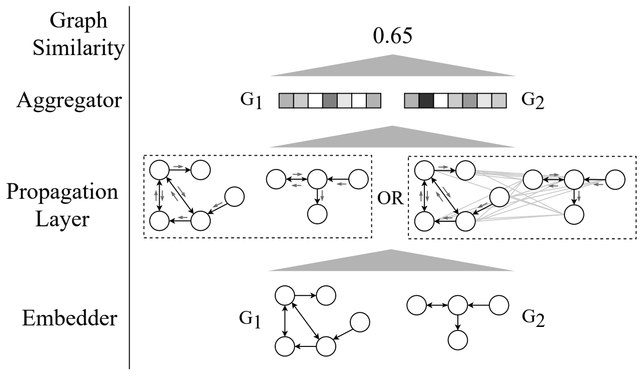

Figure 2: GEM (left branch) and GMN (right branch) (taken from Hoffmann and Bergmann [2022])

models have the same fundamental architecture that consists of four parts that the data passes through in succession: the *embedder*, the *propagation layer*, the *aggregator*, and the *graph similarity*. The embedder maps the types and semantic annotations of nodes and edges to their initial representations in the vector space. The propagation layer then propagates information about each node in its local neighborhood via message passing, where the vector representation of a single node is updated by merging it with the vector representations of all nodes that are connected by an incoming edge. The GEM computes the graph representations for each graph independently, and the node-level information is propagated exclusively in the respective graph. In contrast, the GMN uses a cross-graph matching mechanism that propagates information between two graphs based on node attentions. After multiple propagation iterations, the aggregator joins representations of all nodes to a single graph representation. The graph similarity of two graphs can finally be computed with their respective representations in the vector space, by using a cosine vector similarity in GEM and an MLP in GMN. The architectural differences between GEM and GMN induce a trade-off between expressiveness and performance, with the GMN being more expressive than the GEM but also computationally more expensive.

According to the categorization in Sect. 2, this approach is a pointwise ranking measure. The most relevant cases of the case base w. r. t. a query are retrieved by predicting pairwise similarities with GEM or GMN and sorting the cases according to their similarity. Thus, the GNN-based similarity measure can be used interchangeably with other similarity measures in POCBR applications. However, it might be too strict in retrieval-only applications, where only the case ranks are of interest and not the concrete similarity values.

Predicting Preferences with GNNs

In order to realize a case retrieval, we propose to adjust the similarity-predicting GNNs to predict pairwise preferences of case pairs. Let $G_q, G_1, G_2 \in \mathbb{G}$ be a query and two semantic case graphs from a graph space $\mathbb{G}$, we want to define a function $p : ((\mathbb{G}, \mathbb{G}), (\mathbb{G}, \mathbb{G})) \longrightarrow [0, 1]$ that determines a probabilistic preference between the pairs (G_q, G_1) and (G_q, G_2). A result of $p((G_q, G_1), (G_q, G_2)) \geq 0.5$ means that G_1 is more relevant to the query than G_2, and vice versa. The architectural changes to use GEM and GMN in this context address the part about the graph similarity that is replaced with a preference function. The embedder, the propagation layer, and the aggregator can be reused to embed the graphs into a latent space vector representation. Let $\mathrm{GNN} : (\mathbb{G}, \mathbb{G}) \longrightarrow (\mathbb{R}^n, \mathbb{R}^n)$ be a function that uses either GEM or GMN to embed two graphs to their n-dimensional vector representations, $(v_q, v_1) = \mathrm{GNN}(G_q, G_1)$, and $(v_q, v_2) = \mathrm{GNN}(G_q, G_2)$, then $p((G_q, G_1), (G_q, G_2))$ is defined as follows:

$$p((G_q, G_1), (G_q, G_2)) = \sigma(\mathrm{MLP}_p([v_q - v_1, v_q - v_2]))$$

The pairwise preference between these two graph pairs is computed by element-wise subtracting the vectors v_1 and v_2 from v_q, concatenating the results, processing the results by MLP_p, and activating the final value with a sigmoid activation. MLP_p can be an arbitrary MLP with the constraint that it has to have a single output neuron in the last layer in order to have a single preference value. Training GEM and GMN in this setup also differs from the original goal of predicting similarities. Instead of training on batches of graph pairs, the models are trained on batches of pairs of graph pairs, as function p also suggests. Instead of computing a *Mean Squared Error (MSE)* loss between predicted and label similarity, the predicted preferences are evaluated by computing a binary cross-entropy loss between the predictions and the labels. Please note that the proposed pairwise ranking approach can also be directly trained on label preferences between cases, for instance determined by direct user feedback. This is not possible with the pointwise models, but useful in the real world, since it is more natural for human experts to determine a preference between two alternatives than a similarity value (Hüllermeier and Schlegel 2011).

In order to transform the predicted pairwise preferences to actual ranks which can, in turn, be used as the retrieval result, it is necessary to perform *rank aggregation* (as introduced in Sect. 2). There are overviews of rank aggregators available in the literature (Allwein, Schapire, and Singer 2001; Hüllermeier and Fürnkranz 2004), where most of the methods can be used in this approach. For our use case, we want to present a slightly modified variant of the approach by Cohen et al. [1999] (see Alg. 1) and another algorithm that is simpler and faster to compute (see Alg. 2).

Algorithm 1 is provided with the case base *CB*, the query *QW*, and the preference function $p(\cdot, \cdot)$ introduced before. The rank aggregation starts by creating a temporary set of cases V, where each case is initialized with its *potential ranking score*, denoted by the function $s_1(\cdot)$ (see lines 2 – 4). The value of this function for any case v is initialized as the number of comparisons where v is preferred over some other case u, minus the number of opposed comparisons. The result of the algorithm is a ranked result list of the cases in descending order of relevance (see lines 5 – 12). The result list is put together in a loop, where the case v_{pot} with the highest value of $s_1(v)$ is extracted from V and added to the end of the list. Afterward, the values of function s_1 are updated for each case in V according to the comparisons with the extracted case v_{pot}. The algorithm resembles a greedy search through the graph of pairwise preferences. It has a quadratic computational complexity and is therefore faster than optimal search algorithms such as A*. Cohen et al. [1999] also prove that the algorithm's agreement is at least half the agreement of the optimal ranking order.

Algorithm 1: Greedy Rank Aggregation

```
   Data: case base CB; query workflow QW;
         GNN-based preference function $p(\cdot, \cdot)$
   Result: ranked list of cases
1  $V \longleftarrow CB$;   $result \longleftarrow$ empty list
2  foreach $v \in V$ do
3      $s_1(v) \longleftarrow$
         $\sum_{u \in V} p(v, u) - \sum_{u \in V} p((QW, v), (QW, u))$
4  end
5  while $|V| > 0$ do
6      $v_{\text{pot}} \longleftarrow argmax_{u \in V}\, s_1(u)$
7      add $v_{\text{pot}}$ to the end of $result$
8      $V = V \setminus \{v_{\text{pot}}\}$
9      foreach $v \in V$ do
10         $s_1(v) \longleftarrow$
             $s_1(v) + p((QW, v_{\text{pot}}), (QW, v)) - p(v, v_{\text{pot}})$
11     end
12 end
13 return $result$
```

Algorithm 2 is derived from Alg. 1 and simplifies two aspects to reduce computation time. First, the computation of the potential ranking score function $s_2(v)$ only includes the

Algorithm 2: Positive Rank Aggregation

```
   Data: case base CB; query workflow QW;
         GNN-based preference function $p(\cdot, \cdot)$
   Result: ranked list of cases
1  $V \longleftarrow CB$;   $result \longleftarrow$ empty list
2  foreach $v \in V$ do
3      $s_2(v) \longleftarrow \sum_{u \in V} p((QW, v), (QW, u))$
4  end
5  while $|V| > 0$ do
6      $v_{\text{pot}} \longleftarrow argmax_{u \in V}\, s_2(u)$
7      add $v_{\text{pot}}$ to the end of $result$
8      $V = V \setminus \{v_{\text{pot}}\}$
9  end
10 return $result$
```

preferences where v is preferred over another case u. The opposed preference relations where u is preferred over another case v are not counted, hence the naming as *positive* rank aggregation. Second, the creation of the result list is also done in a loop, but the values of s_2 are not continuously updated. This means that the preferences of cases extracted from V are not removed from the scores of the cases still in V. This rank aggregation algorithm also has a quadratic computational complexity, but requires less computation steps than the greedy one. The performance of both methods is evaluated in the experimental evaluation.

4 Experimental Evaluation

We conduct retrievals to compare pairwise ranking and pointwise ranking with GEM and GMN (see Sect. 3). All different GNN variants also include a version with sequence embedding and one with tree embedding to examine the effects of these semantic annotation embedding techniques, introduced by Hoffmann and Bergmann [2022]. The models based on pairwise ranking also compare the results of the different rank aggregators, i. e., the greedy and the positive approach. This gives a total of twelve evaluated retrievers (see Tab. 1 for the entire enumeration). We measure the quality of several retrieval runs by comparing the order of the results, as predicted by the pointwise and pairwise ranking models, with the ground-truth ordering. In addition, we look at a comparison between the different retrievers regarding retrieval time. The following hypotheses are investigated:

H1 The retrievers based on pairwise ranking generally outperform the retrievers based on pointwise ranking w. r. t. the retrieval quality.

H2 The retrievers based on pairwise ranking generally show reduced retrieval times compared to the retrievers based on pointwise ranking.

Experimental Setup

The experiments are performed with three case bases from different domains, that is of a recipe domain CB-I (Hoffmann et al. 2020), a data mining domain CB-II (Zeyen, Malburg, and Bergmann 2019), and a manufacturing domain CB-III (Malburg, Hoffmann, and Bergmann 2023). CB-I consists of 40 manually modeled cooking recipes that are extended to 800 workflows by adaptation techniques described in Müller [2018], resulting in 660 training cases, 60 validation cases, and 80 test cases. The workflows of the data mining domain CB-II are built from processes stemming from RapidMiner (see Zeyen et al. [2019] for more details), resulting in 509 training cases, 40 validation cases, and 60 test cases. CB-III contains workflows that represent production processes which are used in a smart manufacturing environment. The case base is composed of 75 training cases, nine validation cases, and nine test cases.

When training the neural networks with the data of these three case bases, all semantic graphs are encoded in a numeric vector format (see Hoffmann and Bergmann [2022] for more information). The pointwise ranking models, i. e., the base models, are trained to predict pairwise graph similarities. The number of pairwise graph similarities for training, testing, and validation is precisely the square of the numbers given before, as there is a ground-truth similarity value calculated for each possible case pair. The pairwise ranking models, i. e., the proposed approach, are trained on preferences between graph pairs, computed based on the ground-truth graph similarities. A single training iteration uses a batch of pairs of graph pairs, and the preferences are predicted between each possible pair of graph pairs. This way, the number of different preferences to train the pairwise ranking models is much higher than the number of pairwise graph similarities to train the pointwise ranking models.

The training cases are used as training input for the neural networks, while the validation cases are used to monitor the training process. The proposed models using pairwise ranking use the same hyperparameter settings as the respective base models using pointwise ranking. The training of all models using the GEM architecture runs for 20 epochs, and the training of all models using the GMN architecture runs for 40 epochs. These settings were identified as the approximate points of convergence in pretests. A snapshot of the model is exported after each training epoch and the models with the lowest validation error within these 20 or 40 exported models, respectively, are used in the experiments. The examined metrics cover retrieval quality and performance: Quality is measured in terms of *correctness* (see Cheng and Rademaker [2010] for more details), retrieval hits, and *k-NN quality* (see Müller and Bergmann [2014] for more details). The correctness (ranged between -1 and 1) describes the conformity of the ranking positions in the predicted ranking to the ground-truth ranking. Given two arbitrary workflow pairs $WP_1 = (QW, CW_1)$ and $WP_2 = (QW, CW_2)$, the correctness is decreased if WP_1 is ranked before WP_2 in the predicted ranking although WP_2 is ranked before WP_1 in the ground-truth ranking or vice versa. The retrieval hits measure the number of cases that are within the top-25 retrieved results in the predicted rankings and the ground-truth rankings. The k-NN quality (ranged between 0 and 1) extends the retrieval hits by considering what the similarity of the cases within the top-25 retrieval results is. Therefore, the 25 most similar cases from the ground-truth rankings are compared with the predicted rankings. Each case from the most similar cases that is missing in the predicted rankings decreases the quality, with highly relevant cases affecting the quality more than less relevant cases. In addition, the retrieval time in milliseconds is determined to compare the performance of the individual models. All experiments are computed on a machine, running Ubuntu 22.04, an AMD Ryzen 5700x, an NVIDIA RTX 3070 with 8 GB graphics RAM and 64 GB system RAM. Please note that a comparison between retrievers using A* search and retrievers using GNNs is presented by Hoffmann and Bergmann [2022] and, thus, not included in these experiments.

Experimental Results

Table 1 shows the results of the experimental evaluation for all models on all domains regarding the metrics introduced before. The evaluated models are grouped regarding the used embedder, i. e., a tree embedder or a sequence embedder. The values highlighted in bold font mark the best

Table 1: Evaluation Results.

	Retriever	CB-I				CB-II				CB-III			
		Time (ms)	Corr.	Hits	Quality	Time (ms)	Corr.	Hits	Quality	Time (ms)	Corr.	Hits	Quality
Tree Emb.	GEM Pointwise	0.63	0.035	0.225	0.476	0.53	0.330	2.867	0.439	0.13	0.197	10.333	0.543
	GMN Pointwise	462.11	0.444	0.963	0.490	811.77	0.571	7.033	0.539	171.95	0.584	16.556	0.731
	GEM Pairwise (Positive)	695.63	0.490	3.688	0.546	394.78	0.507	3.567	0.449	20.50	0.470	15.222	0.682
	GEM Pairwise (Greedy)	799.49	0.478	**4.050**	**0.555**	431.98	0.506	3.650	0.451	20.27	0.478	15.333	0.687
	GMN Pairwise (Positive)	1473.28	0.550	2.150	0.519	1348.08	0.616	6.550	0.524	161.04	0.628	**17.333**	**0.760**
	GMN Pairwise (Greedy)	1570.79	**0.551**	2.150	0.519	1391.45	0.615	6.283	0.515	165.19	**0.629**	**17.333**	0.759
Sequence Emb.	GEM Pointwise	**0.48**	0.055	0.013	0.471	**0.44**	0.357	6.150	0.522	**0.08**	0.389	11.778	0.600
	GMN Pointwise	741.83	0.340	0.225	0.476	1110.58	0.615	6.933	0.532	297.90	0.600	15.889	0.710
	GEM Pairwise (Positive)	698.21	0.398	2.125	0.515	398.52	0.524	4.300	0.468	18.32	0.488	15.111	0.678
	GEM Pairwise (Greedy)	797.86	0.385	2.175	0.516	437.91	0.522	4.100	0.463	19.08	0.495	15.111	0.679
	GMN Pairwise (Positive)	1747.54	0.493	2.825	0.535	1674.35	**0.737**	9.600	0.601	284.88	0.555	14.444	0.674
	GMN Pairwise (Greedy)	1841.79	0.491	2.738	0.533	1712.71	0.736	**9.650**	**0.602**	285.98	0.563	14.556	0.677

metric values among all models for each domain. The results w. r. t. training time show a clear dominance of the pointwise GEM models, since they can cache embedding vectors due to their parallel model architecture (see Hoffmann and Bergmann [2022] for more details). There is a general time advantage of GEM as opposed to GMN, which can be seen for pointwise as well as pairwise variants. When directly comparing the pointwise variants with the pairwise variants, it is apparent that the pairwise variants in most cases have longer execution times with an advance between 154 and 1600 times for the GEM variants and 0.92 and 3.36 times for the GMN variants. The GMNs of CB-III do not follow this trend, since they are consistently faster than their respective base models. Additionally, the results also show that most of the pairwise GEMs retrieve faster than the pointwise GMNs, which indicates that the comparison of retrieval time should be done with similarly performing models in terms of quality. Comparing the two rank aggregators, it can be seen that positive rank aggregation is almost always faster than greedy rank aggregation, which is to be expected given the different levels of computational complexity.

The results regarding the quality metrics show that there is a general trend of pairwise models having a higher correctness and k-NN quality and a higher number of hits compared to pointwise models. Except for CB-I, the data also shows a superior quality of the GMN-based models over the GEM-based models. For instance, the differences in correctness between the pairwise models and the respective pointwise model are between 1.2 and 13.9 times for CB-I, 1.07 and 1.53 times for CB-II, and 0.92 and 2.41 times for CB-III. As with training time, some pairwise GMNs do not follow this trend and achieve worse quality values than the respective pointwise models. A comparison among the quality metrics for individual models shows an agreement on average with variations in some cases, e. g., a comparison of the pairwise GEM and GMN with greedy aggregation and tree embedding in CB-I shows a lower correctness of the GEM but, in turn, a higher k-NN quality. When comparing the average results of the two rank aggregators, it can be seen that greedy rank aggregation is usually on-par with positive rank aggregation, with small differences in favor of one or the other approach in some comparisons.

Overall, the results show consistent improvements in quality when using pairwise compared to pointwise ranking approaches. However, these improvements mostly come at the cost of higher retrieval times of pairwise ranking approaches. It leads to a trade-off between these two aspects that should be considered regarding the use case. There are still noteworthy examples where a pairwise GEM model outperforms a pointwise GMN model in terms of quality and performance, leading to a great potential for future use. The performance and the quality results are statistically significant according to a paired two-sample t-test ($p < 0.01$) and we accept hypothesis H1 and reject hypothesis H2.

5 Conclusion and Future Work

The proposed approach deals with case retrieval by using GNN-based predictions of case rankings in POCBR. Previous work on similarity-based case retrieval with GNNs is extended to use the neural networks for predicting pairwise preferences between cases and the query. To transform multiple preferences to a case ranking, two rank aggregation methods are discussed. The experimental evaluation indicates that the ranking-based models significantly increase the quality of the predicted rankings with a small increase in retrieval time compared to similarity-based models.

Future research for the proposed approach should examine concepts of listwise ranking methods (Cao et al. 2007) for similarity-based retrieval in POCBR, since it is, in general, more powerful than pointwise and pairwise ranking (Liu 2011). Furthermore, the proposed approach could be used with a wider variety of rank aggregators, which could include more complex or even simpler rank aggregators for different domains or retrieval scenarios (Allwein, Schapire, and Singer 2001; Hüllermeier and Fürnkranz 2004). Additionally, the approach could be extended to factor in uncertainty, e. g., by not only predicting a pairwise preference, but also the degree of uncertainty for this prediction. Existing methods such as Guiver and Snelson [2008] could then be used to do rank aggregation and determine the final ranking.

Acknowledgements This work is funded by the German Federal Ministry for Economic Affairs and Climate Action under grant No. 22973 *SPELL*.

References

Aamodt, A., and Plaza, E. 1994. Case-based reasoning: Foundational issues, methodological variations, and system approaches. *AI Commun.* 7(1):39–59.

Allwein, E. L.; Schapire, R. E.; and Singer, Y. 2001. Reducing Multiclass to Binary: A Unifying Approach for Margin Classifiers. *J. Mach. Learn. Res.* 1:113–141.

Amin, K.; Kapetanakis, S.; Polatidis, N.; Althoff, K.-D.; and Dengel, A. 2020. DeepKAF: A Heterogeneous CBR & Deep Learning Approach for NLP Prototyping. In Ivanovic, M., ed., *International Conference on INnovations in Intelligent SysTems and Applications*, 1–7. Piscataway, New Jersey: IEEE.

Avesani, P.; Ferrari, S.; and Susi, A. 2003. Case-Based Ranking for Decision Support Systems. In Kevin D. Ashley, and Derek G. Bridge., eds., *Case-Based Reasoning Research and Development, 5th International Conference on Case-Based Reasoning, ICCBR 2003, Trondheim, Norway, June 23-26, 2003, Proceedings*, volume 2689 of *Lecture Notes in Computer Science*, 35–49. Springer.

Bergmann, R., and Gil, Y. 2014. Similarity assessment and efficient retrieval of semantic workflows. *Inf. Syst.* 40:115–127.

Bichindaritz, I. 2006. Memory Organization as the Missing Link Between Case-Based Reasoning And Information Retrieval in Biomedicine. *Computational Intelligence* 22(3-4):148–160.

Brinker, K., and Hüllermeier, E. 2007. Label Ranking in Case-Based Reasoning. In Weber, R., and Richter, M. M., eds., *Case-Based Reasoning Research and Development, 7th International Conference on Case-Based Reasoning, ICCBR 2007, Belfast, Northern Ireland, UK, August 13-16, 2007, Proceedings*, volume 4626 of *Lecture Notes in Computer Science*, 77–91. Springer.

Cao, Z.; Qin, T.; Liu, T.-Y.; Tsai, M.-F.; and Li, H. 2007. Learning to rank: from pairwise approach to listwise approach. In Ghahramani, Z., ed., *Proceedings of the 24th international conference on Machine learning*, 129–136. New York, NY: ACM.

Cheng, W.; Rademaker, M.; de Baets, B.; and Hüllermeier, E. 2010. Predicting Partial Orders: Ranking with Abstention. In Bandini, S.; Manzoni, S.; Umeo, H.; and Vizzari, G., eds., *Cellular automata*, volume 6321 of *Lecture Notes in Computer Science*. Berlin: Springer. 215–230.

Cohen, W. W.; Schapire, R. E.; and Singer, Y. 1999. Learning to Order Things. *Journal of Artificial Intelligence Research* 10:243–270.

Fürnkranz, J., and Hüllermeier, E. 2010. Preference Learning: An Introduction. In Fürnkranz, J., and Hüllermeier, E., eds., *Preference learning*. Berlin and Heidelberg: Springer. 1–17.

Glockner, I., and Weis, K.-H. 2012. An Integrated Machine Learning and Case-Based Reasoning Approach to Answer Validation. In Tao, D., ed., *11th International Conference on Machine Learning and Applications (ICMLA), 2012*. Piscataway, NJ: IEEE.

Guiver, J., and Snelson, E. 2008. Learning to rank with SoftRank and Gaussian processes. In Chua, T.-S., ed., *Proceedings of the 31st annual international ACM SIGIR conference on Research and development in information retrieval*, 259. New York, NY: ACM.

Hoffmann, M., and Bergmann, R. 2022. Using graph embedding techniques in process-oriented case-based reasoning. *Algorithms* 15(2):27.

Hoffmann, M.; Malburg, L.; Klein, P.; and Bergmann, R. 2020. Using siamese graph neural networks for similarity-based retrieval in process-oriented case-based reasoning. In Watson, I., and Weber, R. O., eds., *Case-Based Reasoning Research and Development - 28th International Conference, ICCBR 2020, Salamanca, Spain, June 8-12, 2020, Proceedings*, volume 12311 of *Lecture Notes in Computer Science*, 229–244. Springer.

Hoffmann, M.; Malburg, L.; Bach, N.; and Bergmann, R. 2022. Gpu-based graph matching for accelerating similarity assessment in process-oriented case-based reasoning. In Keane, M. T., and Wiratunga, N., eds., *Case-Based Reasoning Research and Development - 30th International Conference, ICCBR 2022, Nancy, France, September 12-15, 2022, Proceedings*, volume 13405 of *Lecture Notes in Computer Science*, 240–255. Springer.

Hüllermeier, E., and Fürnkranz, J. 2004. Comparison of ranking procedures in pairwise preference learning. In Bouchon-Meunier, B.; Coletti, G.; and Yager, R., eds., *Proceedings of the 10th International Conference on Information Processing and Management of Uncertainty in Knowledge-Based Systems (IPMU-04), Perugia, Italy*, volume 21.

Hüllermeier, E., and Schlegel, P. 2011. Preference-based CBR: first steps toward a methodological framework. In Ram, A., and Wiratunga, N., eds., *Case-Based Reasoning Research and Development - 19th International Conference on Case-Based Reasoning, ICCBR 2011, London, UK, September 12-15, 2011. Proceedings*, volume 6880 of *Lecture Notes in Computer Science*, 77–91. Springer.

Klein, P.; Malburg, L.; and Bergmann, R. 2019. Learning workflow embeddings to improve the performance of similarity-based retrieval for process-oriented case-based reasoning. In Bach, K., and Marling, C., eds., *Case-Based Reasoning Research and Development - 27th International Conference, ICCBR 2019, Otzenhausen, Germany, September 8-12, 2019, Proceedings*, volume 11680 of *Lecture Notes in Computer Science*, 188–203. Springer.

Leake, D., and Crandall, D. J. 2020. On bringing case-based reasoning methodology to deep learning. In Watson, I., and Weber, R. O., eds., *Case-Based Reasoning Research and Development - 28th International Conference, ICCBR 2020, Salamanca, Spain, June 8-12, 2020, Proceedings*, volume 12311 of *Lecture Notes in Computer Science*, 343–348. Springer.

Leake, D., and Ye, X. 2021. Harmonizing case retrieval and adaptation with alternating optimization. In Sánchez-Ruiz, A. A., and Floyd, M. W., eds., *Case-Based Reasoning Research and Development - 29th International Conference,* *ICCBR 2021, Salamanca, Spain, September 13-16, 2021, Proceedings*, volume 12877 of *Lecture Notes in Computer Science*, 125–139. Springer.

Li, H., and Sun, J. 2011. Predicting business failure using forward ranking-order case-based reasoning. *Expert Systems with Applications* 38(4):3075–3084.

Liu, T.-Y. 2011. *Learning to Rank for Information Retrieval*. Berlin and Heidelberg: Springer.

Malburg, L.; Hoffmann, M.; and Bergmann, R. 2023. Applying MAPE-K control loops for adaptive workflow management in smart factories. *Journal of Intelligent Information Systems*.

Mathisen, B. M.; Bach, K.; and Aamodt, A. 2021. Using extended siamese networks to provide decision support in aquaculture operations. *Appl. Intell.* 51(11):8107–8118.

Minor, M.; Montani, S.; and Recio-García, J. A. 2014. Process-oriented case-based reasoning. *Inf. Syst.* 40:103–105.

Müller, G., and Bergmann, R. 2014. A Cluster-Based Approach to Improve Similarity-Based Retrieval for Process-Oriented Case-Based Reasoning. In Torsten Schaub; Gerhard Friedrich; and Barry O'Sullivan., eds., *ECAI 2014 - 21st European Conference on Artificial Intelligence, 18-22 August 2014, Prague, Czech Republic - Including Prestigious Applications of Intelligent Systems (PAIS 2014)*, volume 263 of *Frontiers in Artificial Intelligence and Applications*, 639–644. IOS Press.

Müller, G. 2018. *Workflow Modeling Assistance by Case-based Reasoning*. Springer.

Ye, X.; Leake, D.; and Crandall, D. J. 2022. Case adaptation with neural networks: Capabilities and limitations. In Keane, M. T., and Wiratunga, N., eds., *Case-Based Reasoning Research and Development - 30th International Conference, ICCBR 2022, Nancy, France, September 12-15, 2022, Proceedings*, volume 13405 of *Lecture Notes in Computer Science*, 143–158. Springer.

Zeyen, C., and Bergmann, R. 2020. A*-based similarity assessment of semantic graphs. In Watson, I., and Weber, R. O., eds., *Case-Based Reasoning Research and Development - 28th International Conference, ICCBR 2020, Salamanca, Spain, June 8-12, 2020, Proceedings*, volume 12311 of *Lecture Notes in Computer Science*, 17–32. Springer.

Zeyen, C.; Malburg, L.; and Bergmann, R. 2019. Adaptation of scientific workflows by means of process-oriented case-based reasoning. In Bach, K., and Marling, C., eds., *Case-Based Reasoning Research and Development - 27th International Conference, ICCBR 2019, Otzenhausen, Germany, September 8-12, 2019, Proceedings*, volume 11680 of *Lecture Notes in Computer Science*, 388–403. Springer.