Similarity-Based Retrieval in Process-Oriented Case-Based Reasoning Using Graph Neural Networks and Transfer Learning

Johannes Pauli[1*], Maximilian Hoffmann[1,2*], Ralph Bergmann[1,2]
[1] Artificial Intelligence and Intelligent Information Systems, University of Trier, 54296 Trier, Germany
{s4jopaul,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

Similarity-based retrieval of semantic graphs is a crucial task of *Process-Oriented Case-Based Reasoning (POCBR)* that is usually complex and time-consuming, as it requires some kind of inexact graph matching. Previous work tackles this problem by using *Graph Neural Networks (GNNs)* to learn pairwise graph similarities. In this paper, we present a novel approach that improves on the GNN-based case retrieval with a *Transfer Learning (TL)* setup, composed of two phases: First, the *pretraining* phase trains a model for assessing the similarities between graph nodes and edges and their semantic annotations. Second, the pretrained model is then integrated into the GNN model by either using *fine-tuning*, i. e., the parameters of the pretrained model are further trained, or *feature extraction*, i. e., the parameters of the pretrained model are converted to constants. The experimental evaluation examines the quality and performance of the models based on TL compared to the GNN models from previous work for three semantic graph domains with various properties. The results show the great potential of the proposed approach for reducing the similarity prediction error and the training time.

1 Introduction

Case-based reasoning (CBR) (Aamodt and Plaza 1994) is a problem-solving paradigm that solves a new problem (*query*) by using specific knowledge from previously experienced, concrete problem situations (bundled as *cases* in a *case base*). This procedure relies heavily on similarity computations between the query and each case of the case base to find best-matching cases for the current problem situation (so-called *case retrieval*), based on the assumption that similar problems have similar solutions. This paper focuses on the subdomain of *Process-Oriented CBR (POCBR)* (Minor, Montani, and Recio-García 2014; Bergmann and Gil 2014) where the CBR methodology is applied to procedural knowledge in the form of workflows and process descriptions. The cases in POCBR are commonly represented as semantic graphs (Bergmann and Gil 2014) with a relational structure of nodes and edges of different types and with individual semantic annotations. When computing pairwise similarities between semantic graphs, a single *global similarity* is computed from individual *local similarities* between nodes and edges of the graph and their types and semantic annotations. The similarity measures to compute local similarities usually involve manually-modeled similarity knowledge such as data types, value ranges, and ontologies, and use functions such as weighted sums or set overlaps. The global similarity measures, on the other hand, usually involve some kind of inexact graph matching to determine how well the nodes and edges of the query graph fit the nodes and edges of the case graph. However, finding an optimal mapping is an NP-complete problem (Zeng et al. 2009; Bunke and Shearer 1998; Hoffmann et al. 2022; Zeyen and Bergmann 2020), which can result in long retrieval times and have negative effects on the overall system performance and, ultimately, on the user experience.

To mitigate these issues, several approaches have been proposed (Bergmann and Stromer 2013; Klein, Malburg, and Bergmann 2019; Müller and Bergmann 2014), with the most recent one by Hoffmann and Bergmann [2022] using embedding methods based on *Graph Neural Networks (GNNs)*. GNNs transform semantic graphs to low-dimensional vector representations (so-called *embeddings*) that can be transformed to graph similarities with vector similarity measures or *Multi-Layer Perceptrons (MLPs)*. The models are trained on the prediction error between the predicted graph similarities and the labeled ground-truth similarities, aiming to approximate the manually-modeled ground-truth graph similarity measure based on graph matching. However, the used GNNs are only trained to predict the labeled global graph similarity without any explicit consideration of local similarities, despite this information being available as a result of the labeling process anyway. This can be problematic, as current work is not capable of training or applying local models separate from the global model. Enabling this might increase *flexibility*, as local models and global models can be used in different application scenarios or a single local model can be used in multiple global models. It might also improve *performance*, because the local similarity labels can be used to reduce the prediction error and the training time.

This paper aims to address these issues by using *Transfer Learning (TL)* (Kudenko 2014; Tan et al. 2018) in a

[*] These authors contributed equally to the work.

two-phased approach: In the *pretraining* phase, a model is trained to predict the local similarities between nodes and edges, and in the *adaptation* phase[1], the pretrained model is integrated into the training process of the graph embedding model to predict global pairwise graph similarities. Therefore, the currently used graph embedding models (Hoffmann and Bergmann 2022) are broken up into two models, one for embedding nodes and edges and for computing local similarities, and another one for embedding graphs and for computing graph similarities. The goal is to improve the currently used GNNs (Hoffmann and Bergmann 2022) and ultimately increase the similarity prediction quality, reduce the overall training time, and improve the flexibility of the training process. The paper is organized as follows: Section 2 describes foundations on the used semantic graph representation, the graph similarity computation as well as current graph embedding models and related work. Furthermore, our approach to use TL for improving semantic graph embedding and similarity prediction is presented in Sect. 3. Section 4 evaluates the approach and compares it against existing models. Finally, Section 5 summarizes the work and identifies areas for future work.

2 Foundations and Related Work

The foundations include the semantic workflow representation that is used in the concept and the experiments, as well as the similarity assessment between pairs of these workflows. Further, work on approximating pairwise graph similarities with GNNs and related work is presented.

Semantic Workflow Representation

We represent all workflows as semantically annotated directed graphs referred to as *NEST* graphs, introduced by Bergmann and Gil [2014]. More specifically, a *NEST* graph is 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 $T : N \cup E \rightarrow \Omega$. Additionally, the function $S : N \cup E \rightarrow \Sigma$ assigns a semantic description from Σ (*semantic metadata language*, e. g., an ontology) to nodes and edges. Whereas nodes and edges are used to build the structure of each workflow, types and semantic descriptions are additionally used to model semantic information. Figure 1 shows an exemplary *NEST* graph that represents a sandwich recipe. The mayo-gouda sandwich is prepared by executing the cooking steps `coat` and `layer` (task nodes) with the ingredients `mayo`, `baguette`, `sandwich dish`, and `gouda` (data nodes). All components are linked by edges that indicate relations, e. g., `mayo` is consumed by `coat`. Semantic descriptions of task nodes and data nodes are used to further specify semantic information belonging to the workflow components, e. g., the semantic description of the task node `coat` states that a spoon and a baguette knife are needed to execute the task (`Auxiliaries`) and the estimated time that the task takes is two minutes (`Duration`).

[1]Please note that the term "adaptation" refers to its meaning in the context of TL in the remainder of the paper and is not referring to the reuse phase in CBR.

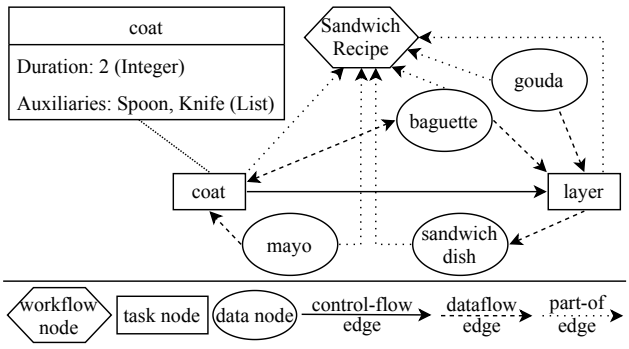

Figure 1: Exemplary Cooking Recipe represented as a NEST Graph

Similarity Assessment

Determining the similarity between two *NEST* graphs, i. e., a query workflow *QW* and a case workflow *CW*, requires a similarity measure that assesses the link structure of nodes and edges as well as their semantic descriptions and types. Bergmann and Gil [2014] propose a semantic similarity measure that determines a global similarity between two graphs based on local similarities, i. e., the pairwise similarities of nodes and edges. The similarity between two nodes with identical types is defined as the similarity of the semantic descriptions of these nodes. 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. In order to put together a global similarity by aggregating local similarities, the domain's similarity model has to define similarity measures for all components of the semantic description, i. e., $sim_\Sigma : \Sigma \times \Sigma \rightarrow [0, 1]$. The global similarity of the two workflows $sim(QW, CW)$ is finally calculated by finding an injective partial mapping m that maximizes $sim_m(QW, CW)$.

$$sim(QW, CW) = max\{sim_m(QW, CW) \mid \text{admissible mapping } m\} \quad (1)$$

A mapping is admissible if all mapped nodes as well as all mapped edges with their source and destination nodes are of the same type. The complex process of finding a mapping that maximizes the global similarity between a query *QW* and a single case *CW* is tackled by utilizing 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; Hoffmann et al. 2022) which motivates the use of automatic learning methods such as GNNs (Hoffmann and Bergmann 2022).

Neural Networks for Semantic Graph Embedding

For the purpose of similarity-based retrieval in POCBR, Hoffmann and Bergmann [2022] adapt two Siamese GNNs, namely the *Graph Embedding Model (GEM)* and the *Graph Matching Network (GMN)*, which are shown in Fig. 2. Both

models learn to predict pairwise graph similarities by embedding the graph structure and the semantic annotations and types to a whole-graph vector representation, before aggregating two of these vectors to compute a single similarity value. Thereby, both models share the same general model

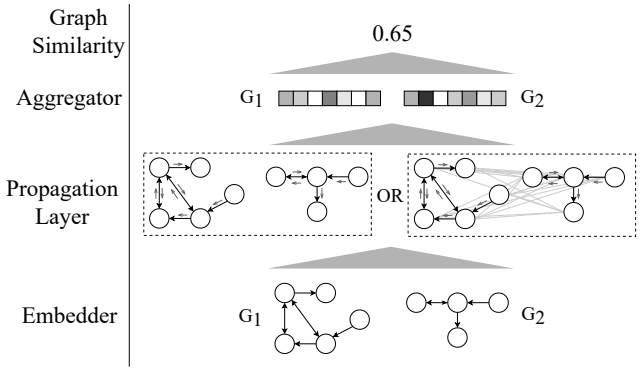

Figure 2: GEM (left branch) and GMN (right branch) (taken from Hoffmann and Bergmann [2022])

architecture that consists of four parts: The *embedder* maps the node and edge features to the initial node and edge vectors in the vector space. The *propagation layer* then accumulates information for each node in its local neighborhood via message passing. More specifically, the vector representation of a single node is updated by merging the vector representation of this node with the vector representations of the neighboring nodes, connected by an incoming edge. After multiple propagation iterations, the *aggregator* converts the set of node representations into a whole-graph representation. The *graph similarity* of two graphs can finally be computed based on their representations in the vector space, e. g., with the help of a vector similarity measure such as cosine similarity. Both models differ in the concrete realization of the propagation layer and the graph similarity. These varieties induce a trade-off between expressiveness and performance, with the GMN being more expressive than the GEM but also computationally more expensive (see Hoffmann and Bergmann [2022] for more details). A general shortcoming of both models, however, is the missing explicit integration of local similarity knowledge, motivating this work for using GEM and GMN in a TL setup to improve similarity predictions.

Related Work

We consider approaches as related that deal with TL in the environment of CBR and specifically POCBR, or apply Deep Learning (DL) methods in the research field of CBR. Approaches using TL in CBR are of limited number. The work of Klenk, Aha, and Molineaux [2011] discusses the perspective that solution reuse as part of the CBR cycle can be considered as TL, since it involves the transfer of knowledge from previous cases to new problems. In this context, other methods for case reuse, e. g., (Müller 2018), can be seen as contributing to TL in this respect. Herold and Minor [2018] investigate the feasibility of using ontologies for TL in POCBR. They try to find an automated ontology-based learning approach for knowledge transfer between two domains, while earlier work (Minor et al. 2016) is based on the manual creation of ontologies. There are also approaches, e. g., (Aha, Molineaux, and Sukthankar 2009), where CBR methods are used to support or improve TL methods, which we want to mention for the sake of completeness.

A greater research interest can be observed for the use of DL methods in CBR (Leake and Crandall 2020). Mathisen, Bach, and Aamodt [2021] and Amin et al. [2020] utilize Siamese neural networks in CBR retrieval to learn similarity measures used in aquaculture and natural language processing tasks, respectively. Leake, Ye, and Crandall [2021], Liao, Liu, and Chao [2018], and Ye, Leake, and Crandall [2022] discuss the application of DL methods in the reuse phase of CBR. Leake, Wilkerson, and Crandall [2022] use neural networks to learn features in CBR tasks. In a more narrow sense, regarding the integration of DL methods in the retrieval of POCBR, the work of Klein, Malburg, and Bergmann [2019] and the work of Hoffmann and Bergmann [2022], on which the present work builds, are related. They have in common that they learn vector representations of semantic graphs and evaluate the learned similarity function in a retrieval scenario. The approach of Klein, Malburg, and Bergmann [2019] only learns on structural properties of semantic graphs, while Hoffmann and Bergmann [2022] additionally consider the semantic information and types of nodes and edges. The proposed approach differs from the aforementioned works in the sense that it is the first work to combine the above topics. In particular, TL is used to improve the generalization capability of the DL models developed by Hoffmann and Bergmann [2022] to improve on the task of similarity-based retrieval.

3 Transfer Learning for Improving Semantic Graph Embedding Models

The global similarity between two semantic graphs can be determined from the local similarities, that is, their node and edge similarities. However, the computation of graph similarities using GEM or GMN lacks the opportunity to take this teaching input into account, possibly reducing the prediction quality. The approach proposed in this paper aims at integrating local similarity knowledge into GEM and GMN by utilizing a *network-based deep TL* strategy (Pan and Yang 2010; Kudenko 2014; Tan et al. 2018). Thereby, a neural network trained on the *source task*, i. e., to predict local similarities, is transferred and adapted to the *target task*, i. e., to predict pairwise graph similarities with GEM and GMN, to aim at improving prediction quality of the GNNs. While pairwise graph similarity prediction consists of a single step with only one training procedure, our approach is composed of two separate phases, i. e., *pretraining* and *adaptation*, that both perform a training step (see Fig. 3). The knowledge transfer is conducted in the adaptation phase, where the neural network, which is pretrained to predict local similarities, becomes a part of GEM or GMN in the target domain

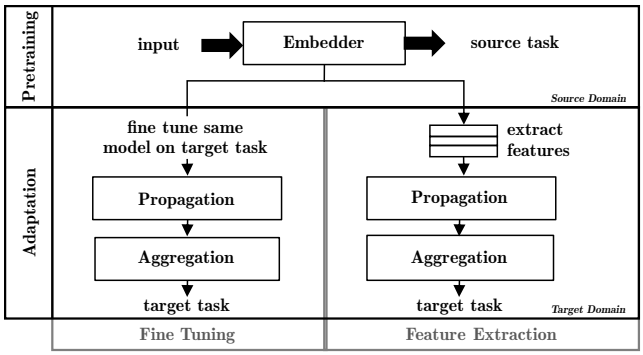

Figure 3: Pretraining and Adaptation Phases of Deep Transfer Similarity Learning.

(Peters, Ruder, and Smith 2019). In this scenario, we extract the embedder from GEM and GMN and then pretrain it to predict local similarities (see Fig. 2). The embedder is well-suited, as it is the only part of the GNN architecture that processes semantic descriptions and types of nodes and edges, which are used to compute the local similarities (see Sect. 2). As already mentioned in the motivation, training the embedder individually has three key advantages: First, it can be used standalone to predict local similarities in various scenarios, increasing flexibility. For instance, it can be used with other similarity measures such as feature-based approaches (Bergmann and Stromer 2013) or to predict local similarities in a matching algorithm (Bergmann and Gil 2014). Second, the trained embedder can be reused for multiple model instances of GEM or GMN, thus, reducing the training effort for these models. Third, the overall precision of the similarity predictions can be improved, as the embedder is directly trained with labeled local similarities, instead of being indirectly trained as part of the whole GNN via the labeled global similarity.

Pretraining

In the pretraining phase, the embedder is trained to predict local similarities of nodes and edges by, first, embedding their semantic descriptions and types and, second, aggregating the embedding vectors to pairwise similarities. The encoding methods of the semantic descriptions and types can be reused from GEM and GMN (see Hoffmann and Bergmann [2022] for more details). We propose to train two separate embedders, one for nodes and one for edges, since their semantic descriptions may be structured differently.

Analogous to GEM and GMN, the pretrained embedder is trained as a Siamese neural network, which is the de-facto standard in metric learning tasks (see Chicco [2021] for an overview). This means that local similarities are learned by embedding two nodes or edges with identical networks and aggregating the two embedding vectors to a single similarity value. The parameters are shared between the two networks and trained by minimizing a loss function, which measures the difference between the ground-truth local similarity and the predicted similarity in the vector space. The embedding procedure can be performed with any kind of neural network architecture and mainly depends on the domain of the semantic descriptions. In previous work (Hoffmann and Bergmann 2022), embedders for sequence and graph-structured data in the form of *Recurrent Neural Networks (RNNs)* and GNNs are presented, which are both applicable in this scenario. The final step of computing a local similarity between the embedding vectors is conducted by an MLP, as also used in the GMN, since it is expressive and easy to compute.

The training examples for pretraining are composed of pairs of semantic descriptions as well as types of nodes and edges. Each training pair is thereby labeled with the ground-truth similarity value in the range of $[0, 1]$, which is determined using local similarity measures (see Sect. 2). Please note that these local similarities are already available from the process of generating labeled data for GEM and GMN and, thus, result in no additional cost. The training is a regression-like problem, where any standard regression loss function such as the *Mean Squared Error (MSE)* can be used. To optimize the parameters of the neural network according to the MSE loss value, any optimizer using stochastic gradient descent, e. g., the Adam optimizer (Kingma and Ba 2015), can be used.

Adaptation

In the adaptation phase of the TL approach, the pretrained embedder is integrated into GEM and GMN by replacing the standard, untrained embedder (see Sect. 2). This adaptation to the task of the target domain is mainly implemented in two ways in the literature (Peters, Ruder, and Smith 2019), either by *feature extraction* or by *fine-tuning*. In feature extraction, similar to classical feature-based approaches (Koehn, Och, and Marcu 2003), the trainable parameters of the pretrained embedder are converted to constants ("frozen"), and the model will not be further trained in the training procedure of GEM and GMN. Alternatively in a fine-tuning setup, the parameters of the pretrained embedder are used as non-random initializations (Dai and Le 2015) and are further trained, i. e., fine-tuned, during the training procedure of GEM and GMN. On the one hand, the general advantage of using feature extraction over fine-tuning is the reduced training effort of the graph embedding models, as the "frozen" weights and biases do not have to be trained. On the other hand, there can be further potential for improvement of prediction quality when fine-tuning the pretrained embedders w. r. t. the source task. Either way, the adaptation is expected to reduce the training time due to a non-random initialization of the neural network parameters. Since there are many factors that affect the choice of the adaption method (see Peters, Ruder, and Smith [2019] for more information), both methods are further examined and evaluated in the remainder of this paper.

4 Experimental Evaluation

To evaluate the impact of integrating local similarities into GEM and GMN, the results of the model training with and without the use of TL are compared. Therefore, we conduct an experiment that consists of two parts: The first part is

concerned with pretraining different embedder variants to find the best-performing one. The best-performing variant is then used in the second phase of the experiment, where it is adapted to GEM and GMN via TL. We examine the quality of the predictions in terms of the MSE between the predicted similarities and the ground-truth similarities. Furthermore, the training time is measured to quantify the impact of TL on the performance of the training procedure, which is important across all DL scenarios. The following hypotheses are investigated:

H1 The integration of local similarities into GEM and GMN based on TL generally improves the prediction quality of the similarities.

H2 The integration of local similarities into GEM and GMN based on TL generally reduces the training time.

Experimental Setup

The experiments are performed with three case bases from different domains, namely workflows of a cooking 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 generalization and specialization of ingredients and cooking steps (see Müller [2018] for more details), resulting in 660 training cases, 60 validation cases, and 80 test cases. The workflows of the data mining domain (CB-II) are built from sample processes that are delivered with RapidMiner (see Zeyen, Malburg, and Bergmann [2019] for more details), resulting in 509 training cases, 40 validation cases, and 60 test cases. CB-III contains workflows that stem from a smart manufacturing IoT environment and represent sample production processes. The case base consists of 75 training cases, nine validation cases, and nine test cases.

While the GNNs in the adaption phase use pairs of semantic graphs and their respective ground-truth similarity as training input, the embedders are pretrained based on the semantic annotations and the types of the nodes[2] from each graph. The examples of the pretraining phase sum up to 14259 training cases, 1439 test cases, and 1219 validation cases for CB-I, 8539 training cases, 1032 test cases, and 670 validation cases for CB-II, and 2136 training cases, 242 test cases, and 265 validation cases for CB-III. The number of graph pairs and node pairs used as examples for training, testing, and validation is exactly the square of the respective numbers of graphs and nodes given before, as there is a ground-truth similarity value calculated for each possible graph and node pair, respectively.

The training examples are used as training input for the neural networks, while the validation examples are used to monitor the training process. The models using TL use the same hyperparameter settings as the respective base models. The training of all models and for both phases, i. e., pretraining and adaption, runs for 40 epochs, and a snapshot of the model is exported after each epoch. The models with the lowest validation error within these 40 exported models are used as reference. The MSE determines the out-of-sample prediction accuracy of the GNNs by computing the squared difference between the predicted values of the GNNs and the actual similarity values for all pairs of graphs in the test data set. In addition, the training time is determined as the time interval between the start of training and the time when the validation loss permanently falls below a certain threshold ϵ. Measuring this time indicates how well a model converges, and enables a comparison regarding training effort between different models.

We evaluate three different model architectures for the embedders, mainly based on related work (Hoffmann and Bergmann 2022): (1) A *Recurrent Neural Network (RNN)* processes the encoded node information as sequence data. (2) An *Attention Neural Network (ANN)* (Vaswani et al. 2017) that uses attention mechanisms to give more weight to certain elements of the node information than others (see Bahdanau, Cho, and Bengio [2015] and Vaswani et al. [2017] for more details on attention). (3) A GNN that processes the node information in a tree-based structure. All experiments are computed on a single NVIDIA Tesla V100-SXM2 GPU with 32 GB graphics RAM.

Experimental Results

Table 1 shows the validation loss w. r. t. the MSE of all evaluated, pretrained embedders for all three case bases. The embedders with the lowest MSE values are highlighted in bold font and are used within the adaptation phase for the respective domain. For both CB-I and CB-III, the best MSE can be observed by the RNN while the GNN achieves the lowest MSE value for CB-II. This shows variations between the embedder architectures for the different domains, which is in-line with other experimental results, e. g., (Hoffmann and Bergmann 2022).

Table 1: Evaluation Results of the Pretraining Phase.

	RNN	ANN	GNN
CB-I	**0.0004**	0.0215	0.0080
CB-II	0.0197	0.0645	**0.0170**
CB-III	**0.0045**	0.0183	0.0078

Table 2 shows the results of the experiments for the adaptation phase. The models using TL are labeled with superscripts indicating the methods used to adapt the pretrained embedder to the downstream task, namely feature extraction (^{FE}) and fine-tuning (^{FT}). The table shows the loss on the test data w. r. t. the MSE and the time span (in seconds) to fall below a certain threshold ϵ for all evaluated models and for all three case bases. The value of ϵ is set to be the lowest validation loss that is reached by all compared models of the same domain. The models are grouped according to their underlying model architecture. This results in a different threshold for each model architecture and all domains. The lowest MSE and time span values are highlighted in

[2]We do not use the edges for pretraining, since they are not semantically annotated in the evaluated domains. The approach, however, supports doing this.

Table 2: Evaluation Results of the Adaptation Phase.

		GEM			GMN		
		GEM	GEM^{FE}	GEM^{FT}	GMN	GMN^{FE}	GMN^{FT}
CB-I			$\epsilon = 0.022$			$\epsilon = 0.006$	
	Training Time (s)	13273	**12070**	15078	123432	**59523**	137524
	MSE	0.0767	0.0712	**0.0661**	0.0028	**0.0017**	0.0024
CB-II			$\epsilon = 0.084$			$\epsilon = 0.008$	
	Training Time (s)	8970	**6707**	8306	81724	**24012**	73960
	MSE	0.0882	0.0841	**0.0709**	0.0068	**0.0054**	0.0063
CB-III			$\epsilon = 0.215$			$\epsilon = 0.033$	
	Training Time (s)	260	171	**47**	1452	**598**	745
	MSE	0.2035	0.1867	**0.1838**	0.0237	0.0206	**0.0193**

bold font, grouped by base model and domain.

The qualities w. r. t. MSE show a significant influence of TL on the base models. Both for GEM and GMN, the models using TL consistently outperform the base models, regardless of the domain. Thereby, the two adaptation methods feature extraction and fine-tuning show different effects for the three case bases. The use of fine-tuning in GEMs leads to an MSE decrease between 10 % and 20 %, while feature extraction decreases the MSE between 5 % and 8 %. The results of the GMN variants are also promising: GMN^{FE} has the best overall MSE values for CB-I and CB-II with an MSE decrease over the base model of about 39 % and 21 %, respectively. GMN^{FT} decreases the MSE by about 14 % for CB-I and 7 % for CB-II. In terms of CB-III, fine-tuning outperforms feature extraction with a 5 % lower MSE.

The training times also show promising results of the TL models. Thereby, feature extraction reduces the training time independent of the model architecture in all domains. GEM^{FE}, for instance, undercuts the threshold ϵ between 9 % and 34 % faster than the base models. GMN^{FE} follows this trend and undercuts ϵ between 52 % and 71 % faster than the base models. Fine-tuning has a positive effect on training time only in CB-II and CB-III, whereas in CB-I it has a negative effect. GEM^{FT} shows a 7 % and 82 % and GMN^{FT} a 10 % and 49 % decrease in training time compared to the base models for CB-II and CB-III, respectively. Hence, for GMN, feature extraction outperforms fine-tuning in terms of training time, while for GEM this is only true for CB-I and CB-II.

In conclusion, the experiments indicate the potential of the proposed approach for reducing the prediction error and the training time. The improved prediction error is likely caused by the explicit use of additional label local similarities that are not used by the base models. In most cases, fine tuning also outperforms feature extraction which suggests that GEM and GMN can benefit from the additional training of parameters in the adaptation phase. Hypothesis H1 is therefore accepted due to a positive effect of TL across all evaluated domains and base models. The results regarding training time are overall very positive, except for GEM^{FT} and GMN^{FT} of CB-I. It is not clear why these two cases lead to worse results when using TL. As a rule of thumb, feature extraction should be preferred for minimizing training time. Thus, we only partly accept H2 but report a high potential for a reduction of training time.

5 Conclusion and Future Work

This paper explores the potential of using TL to predict GNN-based similarities of semantic graphs. The proposed approach splits two existing GNN architectures, that is the GEM and the GMN, into two parts that are trained individually to predict local and global similarities. The resulting models can be used in a similarity-based retrieval, in the context of POCBR, to predict the similarities of pairs of semantic graphs. The models with and without TL are evaluated, focusing on changes in retrieval quality and training time. The experimental results indicate that the integration of local similarities can improve the similarity evaluation quality of both models and reduce training times.

The evaluated implementation of the proposed approach pretrains separate models for each domain, which can lead to a significant training effort for multi-domain setups or when frequently changing the domain definition. Future work could investigate performing GNN model pretraining in the sense of unsupervised learning on a large collection of unlabeled graph data, deriving generic, transferable, and domain-independent knowledge that encodes intrinsic graph properties. Further, analogous to Lu et al. [2021], one can argue that there is a divergence between the two steps of pretraining and adaptation due to different optimization goals targeting the source domain and the target domain, respectively. The pretraining process might currently ignore the need to quickly adapt to the downstream task with a few fine-tuning updates, which could be further analyzed in future work. Additionally, future approaches could consider both node and edge-level and graph-level tasks for pretraining since solely pretraining on nodes and edges may be suboptimal for graph-level similarity assessment, as performed with GEM and GMN, where capturing structural similarities of local neighborhoods may be more important than capturing pairwise local similarities. In addition, a focus of future work should be to perform a more comprehensive evaluation of the proposed approach, and particularly the different embedder variants, to measure the impact for a higher number of domains and DL setups.

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.

Aha, D. W.; Molineaux, M.; and Sukthankar, G. 2009. Case-based reasoning in transfer learning. In McGinty, L., and Wilson, D. C., eds., *Case-Based Reasoning Research and Development, 8th International Conference on Case-Based Reasoning, ICCBR 2009, Seattle, WA, USA, July 20-23, 2009, Proceedings*, volume 5650 of *Lecture Notes in Computer Science*, 29–44. Springer.

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.

Bahdanau, D.; Cho, K.; and Bengio, Y. 2015. Neural machine translation by jointly learning to align and translate. In Bengio, Y., and LeCun, Y., eds., *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*.

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

Bergmann, R., and Stromer, A. 2013. MAC/FAC Retrieval of Semantic Workflows. In Chutima Boonthum-Denecke, and G. Michael Youngblood., eds., *Proceedings of the Twenty-Sixth International Florida Artificial Intelligence Research Society Conference, FLAIRS 2013, St. Pete Beach, Florida, USA, May 22-24, 2013*. AAAI Press.

Bunke, H., and Shearer, K. 1998. A graph distance metric based on the maximal common subgraph. *Pattern Recognit. Lett.* 19(3-4):255–259.

Chicco, D. 2021. Siamese neural networks: An overview. In Cartwright, H. M., ed., *Artificial Neural Networks - Third Edition*, volume 2190 of *Methods in Molecular Biology*. Springer. 73–94.

Dai, A. M., and Le, Q. V. 2015. Semi-supervised sequence learning. In Cortes, C.; Lawrence, N. D.; Lee, D. D.; Sugiyama, M.; and Garnett, R., eds., *Advances in Neural Information Processing Systems 28: Annual Conference on Neural Information Processing Systems 2015, December 7-12, 2015, Montreal, Quebec, Canada*, 3079–3087.

Herold, M., and Minor, M. 2018. Ontology-based representation of workflows for transfer learning. In Gemulla, R.; Ponzetto, S. P.; Bizer, C.; Keuper, M.; and Stuckenschmidt, H., eds., *Proceedings of the Conference "Lernen, Wissen, Daten, Analysen", LWDA 2018, Mannheim, Germany, August 22-24, 2018*, volume 2191 of *CEUR Workshop Proceedings*, 139–149. CEUR-WS.org.

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.

Kingma, D. P., and Ba, J. 2015. Adam: A method for stochastic optimization. In Bengio, Y., and LeCun, Y., eds., *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015*.

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.

Klenk, M.; Aha, D. W.; and Molineaux, M. 2011. The case for case-based transfer learning. *AI Mag.* 32(1):54–69.

Koehn, P.; Och, F. J.; and Marcu, D. 2003. Statistical phrase-based translation. In Hearst, M. A., and Ostendorf, M., eds., *Human Language Technology Conference of the North American Chapter of the Association for Computational Linguistics, HLT-NAACL 2003, Edmonton, Canada, May 27 - June 1, 2003*. The Association for Computational Linguistics.

Kudenko, D. 2014. Special issue on transfer learning. *Künstliche Intell.* 28(1):5–6.

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.; Wilkerson, Z.; and Crandall, D. J. 2022. Extracting case indices from convolutional neural networks: A comparative study. 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*, 81–95. Springer.

Leake, D.; Ye, X.; and Crandall, D. J. 2021. Supporting case-based reasoning with neural networks: An illustration for case adaptation. In Martin, A.; Hinkelmann, K.; Fill, H.; Gerber, A.; Lenat, D.; Stolle, R.; and van Harmelen, F., eds., *Proceedings of the AAAI 2021 Spring Symposium on Combining Machine Learning and Knowledge Engineering*

(AAAI-MAKE 2021), Stanford University, Palo Alto, California, USA, March 22-24, 2021, volume 2846 of *CEUR Workshop Proceedings*. CEUR-WS.org.

Liao, C.; Liu, A.; and Chao, Y. 2018. A machine learning approach to case adaptation. In *First IEEE International Conference on Artificial Intelligence and Knowledge Engineering, AIKE 2018, Laguna Hills, CA, USA, September 26-28, 2018*, 106–109. IEEE Computer Society.

Lu, Y.; Jiang, X.; Fang, Y.; and Shi, C. 2021. Learning to pre-train graph neural networks. In *Thirty-Fifth AAAI Conference on Artificial Intelligence, AAAI 2021, Thirty-Third Conference on Innovative Applications of Artificial Intelligence, IAAI 2021, The Eleventh Symposium on Educational Advances in Artificial Intelligence, EAAI 2021, Virtual Event, February 2-9, 2021*, 4276–4284. AAAI Press.

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.; Bergmann, R.; Müller, J.; and Spät, A. 2016. On the transferability of process-oriented cases. In Goel, A. K.; Díaz-Agudo, M. B.; and Roth-Berghofer, T., eds., *Case-Based Reasoning Research and Development - 24th International Conference, ICCBR 2016, Atlanta, GA, USA, October 31 - November 2, 2016, Proceedings*, volume 9969 of *Lecture Notes in Computer Science*, 281–294. Springer.

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.

Pan, S. J., and Yang, Q. 2010. A survey on transfer learning. *IEEE Trans. Knowl. Data Eng.* 22(10):1345–1359.

Peters, M. E.; Ruder, S.; and Smith, N. A. 2019. To tune or not to tune? adapting pretrained representations to diverse tasks. In Augenstein, I.; Gella, S.; Ruder, S.; Kann, K.; Can, B.; Welbl, J.; Conneau, A.; Ren, X.; and Rei, M., eds., *Proceedings of the 4th Workshop on Representation Learning for NLP, RepL4NLP@ACL 2019, Florence, Italy, August 2, 2019*, 7–14. Association for Computational Linguistics.

Tan, C.; Sun, F.; Kong, T.; Zhang, W.; Yang, C.; and Liu, C. 2018. A survey on deep transfer learning. In Kurková, V.; Manolopoulos, Y.; Hammer, B.; Iliadis, L. S.; and Maglogiannis, I., eds., *Artificial Neural Networks and Machine Learning - ICANN 2018 - 27th International Conference on Artificial Neural Networks, Rhodes, Greece, October 4-7, 2018, Proceedings, Part III*, volume 11141 of *Lecture Notes in Computer Science*, 270–279. Springer.

Vaswani, A.; Shazeer, N.; Parmar, N.; Uszkoreit, J.; Jones, L.; Gomez, A. N.; Kaiser, L.; and Polosukhin, I. 2017. Attention is all you need. In Guyon, I.; von Luxburg, U.; Bengio, S.; Wallach, H. M.; Fergus, R.; Vishwanathan, S. V. N.; and Garnett, R., eds., *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, December 4-9, 2017, Long Beach, CA, USA*, 5998–6008.

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.

Zeng, Z.; Tung, A. K. H.; Wang, J.; Feng, J.; and Zhou, L. 2009. Comparing stars: On approximating graph edit distance. *Proc. VLDB Endow.* 2(1):25–36.

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.