Semi-Supervised Similarity Learning in Process-Oriented Case-Based Reasoning*

Nicolas Schuler[1,3], Maximilian Hoffmann[1,2], Hans-Peter Beise[3], and Ralph Bergmann[1,2]

[1] German Research Center for Artificial Intelligence (DFKI), Branch University of Trier, Trier, Germany
[2] University of Trier, Trier, Germany,
[3] Trier University of Applied Sciences, Trier, Germany
{hoffmannm,bergmann}@uni-trier.de
{schulern,beise}@hochschule-trier.de

Abstract Supervised learning is typically challenging with insufficient amounts of labeled training data and high costs for label acquisition, creating a demand for unsupervised learning methods. In the research area of *Process-Oriented Case-Based Reasoning (POCBR)*, this demand is created by training data that is manually-modeled and computationally-expensive labeling methods. In this paper, we propose a *semi-supervised transfer learning* method for learning similarities between pairs of semantic graphs in POCBR with *Graph Neural Networks (GNNs)*. The method aims to replace the fully supervised learning procedure from previous work with an unsupervised and a supervised training phase. In the first phase, the GNNs are pretrained with a triplet learning procedure that utilizes *graph augmentation* and random selection to enable unsupervised training. This phase is followed by a supervised one where the pretrained model is trained on the original labeled training data. The experimental evaluation examines the quality of the semi-supervised models compared to the supervised models from previous work for three semantic graph domains with different properties. The results indicate the potential of the proposed approach for improving retrieval quality.

Keywords: Semi-Supervised Learning · Transfer Learning · Process-Oriented Case-Based Reasoning · Deep Learning

1 Introduction

Case-Based Reasoning (CBR) [1] is a research field in artificial intelligence, focusing on experience-based problem-solving. The key components of a CBR application [31] are a case base which contains cases of experiential knowledge, similarity measures for determining which case of the case base is suited to solve a new problem, adaptation knowledge to enable modifications of cases for unseen situations, and a vocabulary that defines the underlying domain of the data

* The final authenticated publication is available online at https://doi.org/10.1007/978-3-031-47994-6_12

and the application. The focus of this work is the *retrieval* phase of a CBR application, in which the case base is searched for the most similar, hence most useful, cases w. r. t. a query. Thereby, similarity measures determine the similarity value between the query and the cases, with a higher similarity resulting in a higher problem-solving capability. *Process-Oriented Case-Based Reasoning (POCBR)* [3, 24] aims to integrate the CBR principles with process-oriented applications such as cooking assistance [25], manufacturing [23], and argumentation [22]. A retrieval in the context of POCBR typically assesses the similarity between semantically-annotated processes with similarity measures based on graph isomorphism checks [3]. Their inherent computational complexity [28] paired with large case bases can, thus, lead to long retrieval times and a limited practical applicability.

Different approaches tackle these shortcomings by approximating semantic graph similarities with computationally-inexpensive similarity measures that are, for instance, based on graph feature selection [4] or embeddings of entity-relationship triplets [20]. This paper builds upon an approach where whole-graph embeddings are learned by message-passing *Graph Neural Networks (GNNs)* [16, 17]. These GNNs embed semantic graphs with three consecutive operations: First, all nodes and edges including semantic annotations and types are embedded to an initial vector representations. The node representations then iteratively share information with each other by merging representations of neighboring nodes according to the edge structure. The whole-graph embedding vector is finally determined by aggregating the representations of all nodes. The resulting embedding in the latent, low-dimensional feature space can then be used to calculate the graph similarity, e. g., by a vector similarity measure such as cosine similarity or other distance-based measures such as normed Euclidean distance. The GNNs are trained with pairs of semantic graphs and ground-truth similarities that stem from a similarity measure based on graph matching [3]. The goal of the approach is to speed up retrieval by using GNNs as (fast) similarity measures to predict graph similarities, rather than computing them with computationally-expensive measures [16, 17].

In this paper, we replace the fully supervised training procedure by a semi-supervised transfer learning method [21, 34] that uses an unsupervised and a supervised training phase. In the first, unsupervised phase, the GNNs are trained with a triplet learning procedure [33]. Following is a second supervised training phase, where the original supervised training procedure from previous work [17] is employed. The motivation is to reduce the effort for label computation compared to a solely supervised approach, since an unsupervised training phase can be expected to reduce the amount of labeled data for a subsequent supervised training. This strategy has proven to be effective in other domains [35]. To the best of our knowledge, its application to the task of similarity learning between semantic graphs is novel. The remainder of the paper is structured as follows: Section 2 describes foundations on the used semantic graph representation format and on embedding semantic graphs with GNNs. Additionally, related work on unsupervised learning is discussed. The proposed approach of semi-supervised

transfer learning with semantic graphs is presented in Sect. 3. This approach is experimentally evaluated and compared with previous work in Sect. 4. Finally, Sect. 5 concludes the paper and shows areas of future work.

2 Foundations and Related Work

The foundations include the semantic workflow representation and its corresponding similarity measure that is the base for the concept and the experimental evaluation (see Sect. 2.1). In addition, semantic graph embedding with GNNs is examined (see Sect. 2.2) and related work concerning unsupervised learning with graphs (see Sect. 2.3) and unsupervised learning in CBR literature (see Sect. 2.4) is discussed.

2.1 Semantic Workflow Representation and Similarity Assessment

The workflow representation format used in the remainder of the paper is the NEST graph format, introduced by Bergmann and Gil [3] and common in POCBR literature (e. g., [17, 22, 23, 37]). A NEST graph is defined as a quadruple $W = (N, E, S, T)$ where N is a set of nodes, $E \subseteq N \times N$ a set of edges, $S : N \cup E \to \Sigma$ a function assigning a semantic description to each node and edge from the semantic metadata language Σ, and $T : N \cup E \to \Omega$ a function assigning a type from Ω to each node and edge. Σ is usually given by the domain definition in the form of an ontology or some other knowledge model. An exemplary NEST graph representing a sandwich recipe is given in Fig. 1. The sandwich starts by executing the cooking step *coat* (represented as a task node) with the ingredients *mayo* and *baguette* (represented as data nodes). A slice of gouda is then laid on the coated baguette to finish the simple sandwich. All nodes are connected by edges to indicate relations, e. g., *layer* consumes *baguette*. The workflow components, i. e., nodes and edges, are further specified by using semantic annotations (as shown for the example *coat*). The semantic annotation of this example defines a list of auxiliaries and the time to complete the task. In general, semantic annotations can be arbitrarily complex with different data types (e. g., numerics, strings, dates, etc.) and different compositions that form a tree structure, making similarity computation complex [17].

To calculate the similarity between two given NEST graphs, Bergmann and Gil [3] introduce a similarity measure based on the local-global principle [31]. The global similarity of two NEST graphs is determined by a graph matching procedure that takes into account the local similarities of the semantic annotations of mapped nodes and edges. In this process, nodes and edges of the query graph are mapped by an injective function to nodes and edges, respectively, of the case graph. Thereby, the nodes and edges are mapped onto each other in an A* search with the goal of maximizing the global similarity. The matching process is complex and can take a long time for larger graphs, since the number of possible mappings grows exponentially with the number of nodes and

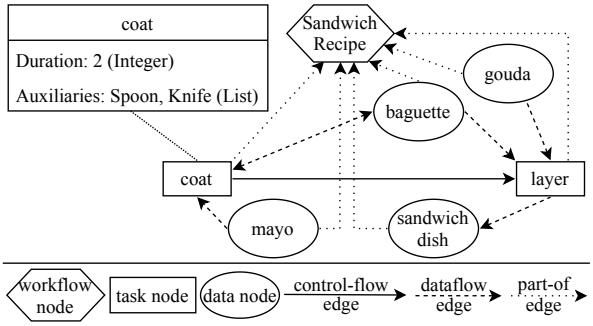

Fig. 1. Exemplary Cooking Recipe represented as NEST Graph.

edges (see [36] for a quantitative analysis). This emphasizes the need for fast, lightweight similarity measures, such as similarity approximation by GNNs [17], to speed up POCBR applications.

2.2 Semantic Graph Embedding

Hoffmann and Bergmann [17] present two Siamese GNNs, i. e., the *Graph Embedding Model (GEM)* and the *Graph Matching Network (GMN)*, for speeding up similarity-based retrieval in POCBR. The GEM and GMN, illustrated in Fig. 2, are trained to predict graph similarities by transforming the graph structure and the semantic annotations and types of all nodes and edges into a whole-graph latent vector representation. These vectors are then combined to calculate a similarity value. Both models follow a shared general architecture composed of four

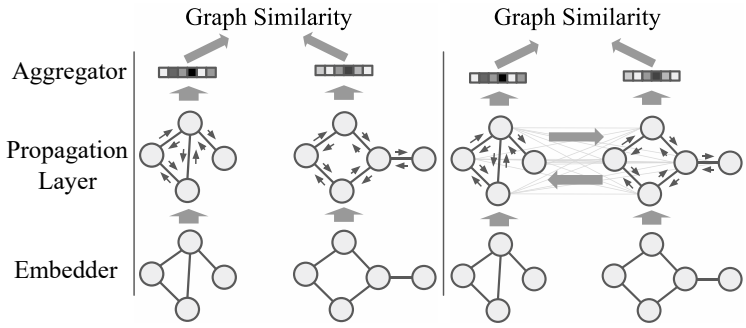

Fig. 2. GEM (left branch) and GMN (right branch) (taken from [16]).

components: First, the *embedder* transforms the features of nodes and edges to initial node and edge embeddings in a vector space. These features comprise semantic annotations and types and are encoded and processed in a very specific

way for semantic graphs (see [17] for more information). Second, the *propagation layer* gathers information for each node from its local neighborhood by passing messages [12]. Specifically, the vector representation of a node is updated by merging its vector representation with those of neighboring nodes connected by incoming edges. This process is iterated multiple times. Subsequently, the *aggregator* combines the node embeddings at that state to form a vector representation of the entire graph. The *graph similarity* between two graphs is eventually computed based on their vector representations in the vector space, for instance, utilizing a vector similarity measure like cosine similarity.

The two models exhibit differences in how they implement the propagation layer and the final graph similarity. These differences result in a trade-off between expressiveness and performance. Specifically, while the GMN provides greater expressiveness than the GEM, it is also associated with a higher computational cost. For a more in-depth exploration of these variances, we refer to Hoffmann and Bergmann [17]. Furthermore, an integral part of the training procedure is to learn the general (implicit) characteristics of semantic graphs without the concrete focus on similarity. This motivates using the GEM and the GMN in an unsupervised or semi-supervised training setup to reduce the needed amount of labeled training data.

2.3 Unsupervised Learning with Graphs

A popular model architecture for unsupervised learning with graphs are *Siamese Neural Networks (SNNs)*, originally introduced by Bromley et al. [5] (see [8] for an in-depth review). These neural networks are used by a variety of unsupervised [13], semi-supervised [32], and supervised [17] approaches. At its core, an SNN is a multitude of identical networks sharing trainable parameters [32]. For graph representation learning, specifically, the neural network itself can be any GNN that processes graphs and yields a vector-space representation. To train the shared weights of the networks, different loss functions can be employed, which include, for instance, triplet loss [33].

Another popular model architecture for unsupervised learning with graphs are *Graph Autoencoders (GAEs)* (see [14] for an in-depth review). These models consist of an encoder, reducing the dimensionality of the input data to a (simple) vector representation, and a decoder, increasing the dimensionality of the vector representation. The training procedure has the goal of improving the encoder and decoder to the point where the decoded data closely matches the original input data. In particular, GAEs offer several features that make them suitable models for POCBR applications. First, their flexible nature allows utilizing a variety of architectures [14] for the encoder, which in turn can also be used to pretrain an encoder that might be used later in a different context. In addition, an autoencoder enables the decoding of the embedded graph from the latent space back to its initial representation, enabling generative model architectures. Albeit initially developed for relational graphs such as social networks, Kipf and Welling [19] present variational GAEs as a generative model that is generally also applicable to semantic graphs representing processes.

2.4 Unsupervised Learning in CBR

In the present work, unsupervised learning of graph representations is paired with supervised graph embedding and transfer learning. For a more comprehensive discussion of related deep learning and transfer learning applications in the CBR context, see previous work [17, 30]. Approaches utilizing unsupervised learning in CBR research are examined in the following: In the field of textual CBR, Naqvi et al. [27] use an unsupervised autoencoder to fine tune a deep language model to adopt it to a specific CBR domain in the context of prognostics and health management. Amin et al. [2] utilize word embeddings for unsupervised text vectorization and word representation learning with SNNs in CBR to perform customer service tasks. Similarly, Lenz et al. [22] apply supervised and unsupervised methods from textual CBR to argument graphs, that is, arguments represented as a graph, in the context of similarity-based retrieval. The work focuses on semantic textual similarity, which is improved by utilizing unsupervised word embeddings and similarity measures beyond simple vector measures. Chourib et al. [9] apply unsupervised k-means clustering to generate representations of medical knowledge from the case base for similarity estimations and quality assessments. For POCBR in particular, Klein et al. [20] use a generic triplet embedding framework for unsupervised triplet representation learning in the context of similarity-based retrieval. However, this work did neither consider the entire graph structure nor the semantic annotations of nodes and edges [20].

These examples show that many of the approaches do not focus on graph-structured data and are thus not suitable to be applied in POCBR applications. To the best of our knowledge, the proposed approach is novel in the sense that it combines unsupervised learning, supervised graph embedding, and transfer learning in a POCBR context.

3 Semi-Supervised Transfer Learning with Semantic Graphs

In order to reduce the effort for labeling training data for supervised graph similarity learning tasks in POCBR, the approach proposed in this paper aims at using semi-supervised transfer learning with SNNs. An overview of the components and the main steps is given in Fig. 3. The approach uses three separate learning paradigms that are combined and explained in the remainder of this section, i. e., unsupervised triplet learning, supervised graph similarity learning, and a transfer learning scenario.

Transfer learning acts on the highest level of the architecture and combines the other two components. The general idea is that knowledge gained from a source domain in the pretraining phase (see step 2) can be transferred to a target domain in the adaptation[1] phase (see step 3) [21, 34]. In our context,

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

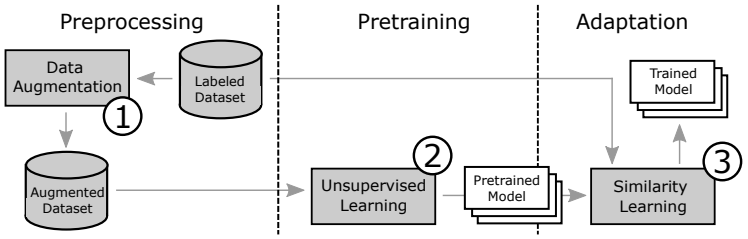

Fig. 3. Architecture for Semi-Supervised Transfer Learning With Its Three Main Steps.

a graph embedding model, acting as the knowledge, is transferred from the pretraining phase, an unsupervised learning procedure, to the adaptation phase, a supervised learning procedure. A preprocessing phase for data augmentation (see step 1) precedes the pretraining phase to create a large unlabeled dataset to be used by the unsupervised learning method. The underlying assumption is that unsupervised pretraining with augmented data reduces the need for a large amount of labeled data in a subsequent supervised training, as supported by literature [35]. However, to apply this strategy, all phases of the transfer learning process must be designed to be compatible.

3.1 Unsupervised Triplet Learning

The pretraining phase employs an SNN with a triplet loss [33] for learning unsupervised graph embeddings (see Fig. 4). As the concrete models to be used in the SNN configuration, we use the GEM and GMN from previous work [17] (see Sect. 2.2). The idea is to generate similar embeddings for similar graphs and dissimilar embeddings for dissimilar graphs (in terms of vector space similarity). Since we do not compute the ground-truth similarities for graphs in the training data, the loss function operates on graph triplets of an *anchor*, a *negative*, and a *positive*, maximizing the similarity for the pair of the anchor and the positive and minimizing the similarity for the pair of the anchor and the negative. More formally, the triplet loss is defined in Eq. 1: Let $T = (x^a, x^p, x^n)$ be a triplet consisting of one input vector for anchor x^a, positive x^p, and negative x^n. Let $f(x_i) = m_i \in \mathbb{R}^n$ denote the embedding of an input vector x_i as the n-dimensional embedding m_i. To train a model based on the triplet loss, the loss function L is minimized with α as a margin hyperparameter and N as the cardinality of a batch of triplets [33].

$$L = \sum_{j}^{N} \max\{0, \|m_j^a - m_j^p\|^2 - \|m_j^a - m_j^n\|^2 + \alpha\} \tag{1}$$

The main challenge for this training method in an unsupervised context is to put together triplet training examples of an anchor, a negative, and a positive. There are different solutions in literature, depending on the type of data, the learning task, and other factors (e. g., [15, 29]). We propose to reuse popular

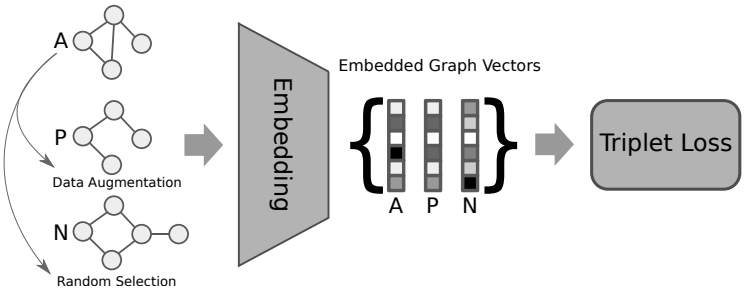

Fig. 4. Unsupervised Triplet Graph Embedding using an Anchor (A), a Positive (P), and a Negative (N).

methods that originally stem from the domain of computer vision [6]. In this domain, an anchor is commonly selected from the training set and the positive is generated based on the anchor by data augmentation, while the negative is randomly selected from the training set. For instance, Chen et al. [6] use data augmentation like rotation, cropping, color distortion, filtering, and blurring of images. However, while the random selection of a graph from the dataset to generate the negative is trivial, the generation of the positive via data augmentation is challenging due to the complexity of the underlying semantic graphs (see Sect. 2.1).

3.2 Semantic Graph Augmentation

In this unsupervised setup, a challenge arises when defining suitable positives x^p using data augmentation methods [11], as it requires configuring the augmentation techniques in a way that ensures the positive samples are actually similar to the anchor. So called hard or semi-hard triplets [15] are needed. In a semi-hard triplet, the negative's embedding has a greater vector-space distance to the anchor's embedding than the positive's embedding, but still within the range of the margin parameter α (see Eq. 1). A hard triplet is stricter, such that the negative's embedding must be closer to the anchor's embedding than the positive's embedding. As there are no similarity labels to ensure this property for generated triplets, the data augmentation methods can only work with heuristics. This is particularly challenging for the domain of semantic graphs due to their characteristics and definition of similarity (see Sect. 2.1) where small changes, for instance on the level of a single semantic annotation, can have large effects on the global similarity. We present two methods of semantic graph augmentation in the following (see [10, 26] for a comprehensive overview of graph augmentation techniques) that are simple by design. Their purpose is to demonstrate the aspect of semantic graph augmentation and to be used in the experimental evaluation. A thorough investigation of other, more complex augmentation methods is beyond the scope of this work and, therefore, postponed to future work.

The first method augments processes by randomly changing the order of two subsequent task nodes in the control-flow sequence. Looking at the example in Fig. 1, the method would swap the order of *coat* and *layer*, resulting in *layer* being the first and *coat* being the second task node in the control-flow sequence. In a larger graph, the other parts of the control-flow remain unchanged. The method can be parameterized by the number of swaps and a random seed to ensure reproducibility. The second method randomly deletes edges of a certain type to augment processes. For instance, in Fig. 1, the method could randomly delete two of the shown part-of edges. The connected nodes remain unchanged. The method can be parameterized by the number of deletions, the type of the edges to delete, and a random seed to ensure reproducibility. While both augmentation methods are simple, they are still fundamentally different regarding the resulting augmented processes. The first augmentation method maintains the syntactic correctness of the augmented process in terms of the NEST graph format (see Sect. 2.1). This means that the augmented process could, for instance, be executed in a process execution engine. The second method does not preserve syntactic correctness, making the augmented process unusable for execution or other tasks that require syntactic correctness. Both methods, however, do not preserve semantic correctness that is defined by the POCBR domain and similarity models, e. g., , "a baguette cannot be *layer*ed before *coat*ed". The method we employ is based on the assumption that neural networks are still able to learn the basic structure and composition of the original processes from the augmented processes, even if the latter do not feature syntactic or semantic correctness.

3.3 Graph Embedding with GEM and GMN

The GEM and the GMN (see Sect. 2.2) are used for the task of graph embedding in the *pretraining* phase with triplet learning and in the *adaptation* phase. Their task in the adaptation phase is supervised embedding of graph pairs, which exactly matches their original purpose (see previous work, e. g., [17], and Fig. 2). However, the triplet learning method in the pretraining phase requires adjustments to the models. The GEM is not used, as originally introduced, with pairs but with triplets of graphs in the pretraining phase. These graph triplets are also processed with shared parameters. As the GEM supports processing tuples of graphs independent of their number out of the box, this change is more focused on the implementation. The modifications to the GMN are more substantial due to the cross-graph matching procedure in the propagation layer that is integral to the superior prediction quality of the GMN compared to the GEM. That is, this matching procedure requires pairs of graphs to function properly due to the involved information propagation between the two graphs (see Fig. 2). Since the triplet learning procedure in the pretraining phase works with graph triplets, i. e., anchor, positive, and negative, the GMN must be adjusted. We employ the GMN to compute embeddings for two graph pairs, i. e., a pair of anchor and positive and a pair of anchor and negative. The four resulting graph embedding vectors are then reduced to three for computing the triplet loss. The reduction is

done by aggregating the embedding vectors of both anchors in a trainable procedure, which, in this work, is implemented by an MLP. These adjustments to the GEM and the GMN allow the unsupervised pretraining and the supervised adaptation of both models in the transfer learning process.

4 Experimental Evaluation

The experimental evaluation compares similarity-based retrieval between the proposed semi-supervised transfer learning approach and the supervised baseline approach (as introduced in Sect. 2.2). We evaluate the quality of the retrieval results in terms of the similarity prediction error and the errors in the order of the retrieval results (see Sect. 4.1 for more details). The evaluation consists of multiple experiments, where the effects of different amounts of labeled training data are investigated. In total, 36 different retrievers with different model configurations are compared in three workflow domains. The aim of the evaluation is to investigate the effect of the proposed approach on the quality of the retrieval results.

4.1 Experimental Setup

The experiments involve three different case bases from different domains. These domains are the cooking domain (CB-I), the data mining domain (CB-II), and the manufacturing domain (CB-III). CB-I comprises 40 cooking recipes, manually modeled and then expanded to 800 workflows through the generalization and specialization of ingredients and cooking steps (more details in [17, 25]). The case base contains a total of 660 training cases, 60 validation cases, and 80 test cases. In CB-II, the workflows are derived from sample processes provided with RapidMiner. The case base consists of 509 training cases, 40 validation cases, and 60 test cases. Additional information on this domain can be found in [37]. CB-III encompasses workflows originating from a smart manufacturing IoT environment, representing sample production processes. The case base includes 75 training cases, nine validation cases, and nine test cases (more details in [23]).

The workflows of these domains are augmented with the two methods described in Sect. 3.2 to increase the size of the dataset. The augmentation is done in the following set of steps: First, for each workflow of the initial original dataset, one random part-of edge is deleted. In the second step, the set containing the original and the augmented workflows from the first step are augmented again by changing the order of one random pair of subsequent task nodes. Thus, the result is a new, unlabeled dataset containing four times the amount of data of the initial dataset, with parts of the dataset being syntactically and semantically correct and some being not. The augmented datasets are only used for the unsupervised training process in the pretraining phase, while the supervised adaptation phase only uses the original, non-augmented data. To evaluate the effect of the amount of labeled data that is available for training, each supervised training procedure in the adaptation phase is conducted once with 100

percent, 50 percent, and 25 percent of the labeled training data, respectively. Please note, the number of graph pairs used as examples for training, testing, and validation is exactly the square of the respective numbers of graphs given before. For supervised training, there is one ground-truth similarity value calculated for each possible graph pair and for unsupervised training, we select each possible graph pair as anchor and negative and select the positive from the list of augmentations of the anchor. This ultimately results in the following number of supervised/unsupervised training cases: 435,600/6,494,400 for CB-I, 259,081/4,145,296 for CB-II, and 5625/90,000 for CB-III. All unsupervised and supervised models are trained until convergence based on the training progress on the validation data (early stopping).

The examined metrics cover the retrieval quality. Quality is measured in terms of *Mean Absolute Error (MAE)* and *correctness* (see [7] for more details). The MAE (lower values are better) measures the average absolute difference between the ground-truth similarity and the predicted similarity of the retrievers. The correctness metric, which ranges between -1 and 1 (higher values are better), evaluates the degree to which the predicted ranking aligns with the ground-truth ranking by penalizing inconsistencies where the predicted ranking contradicts the order of the ground-truth ranking for a given pair of workflows. Consider two arbitrary workflow pairs $p_1 = (W_i, W_j)$ and $p_2 = (W_k, W_l)$. The correctness value is reduced if p_1 is ranked before p_2 in the predicted ranking, despite p_2 being ranked before p_1 in the ground-truth ranking, and vice versa.

4.2 Experimental Results

Table 1 shows the results of the experimental evaluation for all models in all domains regarding the metrics introduced before. The models are grouped into semi-supervised and supervised and the percentage of total labeled data used in the supervised training step. The values highlighted in bold font mark the best metric values among a domain for a particular model (i. e., GEM or GMN). The median relative distance gives the aggregated difference between the semi-supervised and supervised models of the respective metric, with positive values indicating a better performance of the semi-supervised model and vice versa. The

Table 1. Evaluation Results.

		Semi-Supervised						Supervised						Med. Rel. Diff.	
		GEM			GMN			GEM			GMN			GEM	GMN
		25%	50%	100%	25%	50%	100%	25%	50%	100%	25%	50%	100%		
CB-I	Correctness	**0.271**	0.219	0.148	0.352	0.315	0.406	-0.006	0.061	0.030	0.363	**0.477**	0.411	260%	-3%
	MAE	0.270	0.286	**0.162**	0.076	0.070	**0.054**	0.382	0.283	0.296	0.097	0.076	0.056	29%	8%
CB-II	Correctness	0.335	0.333	0.178	0.478	0.562	**0.581**	0.246	**0.355**	0.329	0.407	0.543	0.532	-6%	9%
	MAE	0.299	0.280	**0.198**	0.067	0.054	**0.053**	0.331	0.406	0.430	0.077	0.070	0.059	31%	13%
CB-III	Correctness	**0.414**	0.349	0.358	0.670	0.747	**0.781**	0.258	0.301	0.195	0.464	0.479	0.583	60%	44%
	MAE	0.153	0.145	**0.111**	0.051	0.046	**0.032**	0.399	0.386	0.402	0.078	0.081	0.064	62%	43%

results regarding the MAE show a stronger performance of the semi-supervised over the supervised models. The only exception is the GEM (50%) of CB-I, where the supervised model outperforms the semi-supervised one. It is particularly notable that the semi-supervised GEMs trained on 25% of the data consistently outperform their supervised counterparts trained on 100% of the data. The results w. r. t. the correctness show no clear dominant training method. In the smallest domain CB-III, the semi-supervised models show a stronger performance than the supervised models. The results in CB-I and CB-II, however, are mixed and give no clear indication on the superiority of either the semi-supervised or the supervised models w. r. t. correctness. The median relative differences indicate two aspects: First, the simpler GEM model profits more from the semi-supervised training than the already stronger performing GMN model, though both models show an increased performance w. r. t. their MAEs. Second, the smallest domain CB-III profits much more from semi-supervised training than the two larger domains CB-I and CB-II, with improvements of up to 83%.

We want to further discuss the correctness values and the influence of smaller amounts of labeled data. The correctness values can generally not be improved as much as the MAE values throughout the experiment. We suppose the focus on minimizing the similarity prediction error in training to be the main reason for this. The results might be different when training directly to improve the correctness, as done in previous work [18]. Furthermore, it can be observed that the median relative differences of CB-III show the strongest results in this comparison. The reason for this might be that it is by far the smallest dataset and, thus, benefits more from additional unlabeled training data. However, the effect of smaller amounts leading to a higher relative difference of the semi-supervised compared to the supervised model is not consistent within the domains. For instance, the MAE values of the GEMs of CB-II show a relative difference of 9.7% for 25%, 31% for 50%, and 54% for 100% of the dataset.

In summary, the quality results show a consistent improvement of the MAE metric and mixed effects w. r. t. the correctness when comparing semi-supervised and supervised learning.

5 Conclusion and Future Work

The proposed approach presents a semi-supervised transfer learning method for similarity learning in POCBR applications. Thereby, previous work on supervised graph similarity prediction is combined with an unsupervised pretraining phase that uses a large, unlabeled and augmented dataset. The goal is to reduce the amount of labeled training data needed in this procedure, with the underlying assumption that unsupervised pretraining with augmented data enables this. The experiments compare the proposed semi-supervised models with baseline supervised models in retrieval scenarios w. r. t. retrieval quality. Overall, the experimental results indicate that the semi-supervised models predict similarities more accurately.

In future work, it is planned to increase the efficacy of the introduced SNN architecture by improving the data augmentation process with other augmentation methods. These methods should deal with the specific properties of semantic workflows, which complicates the use of standard augmentation methods, as described in [10, 26]. Furthermore, future work should aim at implementing the proposed unsupervised training method with GAEs, which are briefly discussed in Sect. 2.3. A major benefit of using GAEs is the ability to extend the decoding process for enabling the generation of new graphs (so-called Variational GAEs [19]). Graph generation might be an important topic for many active research areas of CBR and POCBR such as case reuse (e. g., [23, 37]).

References

1. Aamodt, A., Plaza, E.: Case-based reasoning: Foundational issues, methodological variations, and system approaches. AI Communications **7**(1), 39–59 (1994)
2. Amin, K., et al.: Advanced similarity measures using word embeddings and siamese networks in CBR. In: Adv. in Int. Syst. and Comp., pp. 449–462. Springer (2019)
3. Bergmann, R., Gil, Y.: Similarity assessment and efficient retrieval of semantic workflows. Information Systems **40**, 115–127 (2014)
4. Bergmann, R., Stromer, A.: MAC/FAC Retrieval of Semantic Workflows. In: Proc. of the 26th Int. Florida Artif. Intell. Res. Society Conf. AAAI Press (2013)
5. Bromley, J., et al.: Signature Verification Using A "Siamese" Time Delay Neural Network. IJPRAI **7**(4), 669–688 (1993)
6. Chen, T., Kornblith, S., Norouzi, M., Hinton, G.E.: A simple framework for contrastive learning of visual representations. In: Proc. of the 37th ICML, Virtual Event. vol. 119, pp. 1597–1607. PMLR (2020)
7. Cheng, W., et al.: Predicting Partial Orders: Ranking with Abstention. In: ML and Knowl. Disc. in Databases, Part I. LNCS, vol. 6321, pp. 215–230. Springer (2010)
8. Chicco, D.: Siamese neural networks: An overview. In: Methods in Molecular Biology, pp. 73–94. Springer US (2020)
9. Chourib, I., Guillard, G., Farah, I.R., Solaiman, B.: Structured case base knowledge using unsupervised learning. In: 2022 6th International Conference on Advanced Technologies for Signal and Image Processing (ATSIP). IEEE (2022)
10. Ding, K., Xu, Z., Tong, H., Liu, H.: Data Augmentation for Deep Graph Learning. ACM SIGKDD Explorations Newsletter **24**(2), 61–77 (2022)
11. van Dyk, D.A., Meng, X.L.: The art of data augmentation. Journal of Computational and Graphical Statistics **10**(1), 1–50 (2001)
12. Gilmer, J., et al.: Neural message passing for quantum chemistry. In: Proc. of the 34th ICML, Australia. vol. 70, pp. 1263–1272. PMLR (2017)
13. Gong, Y., Yue, Y., Ji, W., Zhou, G.: Cross-domain few-shot learning based on pseudo-siamese neural network. Scientific Reports **13**(1) (2023)
14. Hamilton, W.L.: Graph Representation Learning. Springer International Publishing (2020)
15. Hermans, A., Beyer, L., Leibe, B.: In defense of the triplet loss for person reidentification. CoRR **abs/1703.07737** (2017)
16. Hoffmann, M., et al.: Using siamese graph neural networks for similarity-based retrieval in process-oriented case-based reasoning. In: Case-Based Reason. Res. and Dev.: 28th Int. Conf. ICCBR 2020, Spain, Proc., pp. 229–244. Springer (2020)

17. Hoffmann, M., Bergmann, R.: Using graph embedding techniques in process-oriented case-based reasoning. Algorithms **15**(2), 27 (2022)
18. Hoffmann, M., Bergmann, R.: Ranking-Based Case Retrieval with Graph Neural Networks in Process-Oriented Case-Based Reasoning. The International FLAIRS Conference Proceedings **36** (2023)
19. Kipf, T.N., Welling, M.: Variational graph auto-encoders. CoRR **abs/1611.07308** (2016)
20. Klein, P., Malburg, L., Bergmann, R.: Learning Workflow Embeddings to Improve the Performance of Similarity-Based Retrieval for Process-Oriented Case-Based Reasoning. In: 27th ICCBR 2019, Germany. pp. 188–203. Springer (2019)
21. Kudenko, D.: Special issue on transfer learning. Künstliche Intell. **28**(1), 5–6 (2014)
22. Lenz, M., et al.: Semantic textual similarity measures for case-based retrieval of argument graphs. In: 27th ICCBR 2019, Germany, pp. 219–234. Springer (2019)
23. Malburg, L., Hoffmann, M., Bergmann, R.: Applying MAPE-K control loops for adaptive workflow management in smart factories. Journal of Int. Inf. Syst. (2023)
24. Minor, M., Montani, S., Recio-García, J.A.: Process-oriented case-based reasoning. Information Systems **40**, 103–105 (2014)
25. Müller, G.: Workflow Modeling Assistance by Case-based Reasoning. Springer (2018)
26. Mumuni, A., Mumuni, F.: Data augmentation: A comprehensive survey of modern approaches. Array **16**, 100258 (2022)
27. Naqvi, S.M.R., et al.: CBR-based decision support system for maintenance text using NLP for an aviation case study. In: 2022 Prognostics and Health Management Conference (PHM), London, 2022. IEEE (2022)
28. Ontañón, S.: An overview of distance and similarity functions for structured data. Artificial Intelligence Review **53**(7), 5309–5351 (2020)
29. Ott, F., et al.: Cross-modal common representation learning with triplet loss functions. CoRR **abs/2202.07901** (2022)
30. Pauli, J., Hoffmann, M., Bergmann, R.: Similarity-Based Retrieval in Process-Oriented Case-Based Reasoning Using Graph Neural Networks and Transfer Learning. The International FLAIRS Conference Proceedings **36** (2023)
31. Richter, M.M.: Foundations of similarity and utility. Proceedings of the Twentieth International Florida Artificial Intelligence Research Society Conference (2007)
32. Sahito, A., Frank, E., Pfahringer, B.: Semi-supervised learning using siamese networks. In: Liu, J., Bailey, J. (eds.) AI 2019: Adv. in Artif. Int. - 32nd Australasian Joint Conference, Australia. LNCS, vol. 11919, pp. 586–597. Springer (2019)
33. Schroff, F., Kalenichenko, D., Philbin, J.: Facenet: A unified embedding for face recognition and clustering. In: CVPR, USA. pp. 815–823. IEEE (2015)
34. Tan, C., et al.: A survey on deep transfer learning. In: 27th International ICANN, Greece. LNCS, vol. 11141, pp. 270–279. Springer (2018)
35. Weiss, K., Khoshgoftaar, T.M., Wang, D.: A survey of transfer learning. Journal of Big Data **3**(1) (2016)
36. Zeyen, C., Bergmann, R.: A*-Based Similarity Assessment of Semantic Graphs. In: Case-Based Reason. Res. and Dev.:, LNCS, vol. 12311, pp. 17–32. Springer (2020)
37. Zeyen, C., Malburg, L., Bergmann, R.: Adaptation of Scientific Workflows by Means of Process-Oriented Case-Based Reasoning. In: Case-Based Reason. Res. and Dev.:, LNCS, vol. 11680, pp. 388–403. Springer (2019)