Using Siamese Graph Neural Networks for Similarity-Based Retrieval in Process-Oriented Case-Based Reasoning*

Maximilian Hoffmann[1], Lukas Malburg[1], Patrick Klein[1], and Ralph Bergmann[1,2]

[1] Business Information Systems II, University of Trier, 54296 Trier, Germany
{hoffmannm,malburgl,kleinp,bergmann}@uni-trier.de
http://www.wi2.uni-trier.de
[2] German Research Center for Artificial Intelligence (DFKI)
Branch University of Trier, Behringstraße 21, 54296 Trier, Germany
ralph.bergmann@dfki.de

Abstract Similarity-based retrieval of semantic graphs is widely used in real-world scenarios, e. g., in the domain of business workflows. To tackle the problem of complex and time-consuming graph similarity computations during retrieval, the *MAC/FAC* approach is used in *Process-Oriented Case-Based Reasoning* (POCBR), where similar graphs are extracted from a preselected set of candidate graphs. These graphs result from a similarity computation with a computationally inexpensive similarity measure. The contribution of this paper is a novel similarity measure where vector space embeddings generated by two siamese *Graph Neural Networks* (GNNs) are used to approximate the similarities of a precise but therefore computationally complex graph similarity measure. Our approach includes a specific encoding scheme for semantic graphs that enables their usage in neural networks. The evaluation examines the quality and performance of these models in preselecting retrieval candidates and in approximating the ground-truth similarities of the graph similarity measure for two workflow domains. The results show great potential of the approach for being used in a MAC/FAC scenario, either as a preselection model or as an approximation of the graph similarity measure.

Keywords: Process-Oriented Case-Based Reasoning · MAC/FAC Retrieval · Graph Embeddings · Siamese Graph Neural Networks

1 Introduction

Nowadays, cases represented as semantic graphs are increasingly used in several domains, e. g., as cooking recipes in the form of simple business workflows

* The final authenticated publication is available online at https://doi.org/10.1007/978-3-030-58342-2_15

[19], as scientific workflows to represent data mining tasks [28], or as argument graphs for case-based argumentation [14]. The problem-solving paradigm of *Process-Oriented Case-Based Reasoning* (POCBR) [3,16] focused on these semantic graphs to represent workflows in scenarios of similarity-based retrieval and reuse of procedural experiential knowledge. Especially in retrieval situations, the main influencing factor on user experience is its runtime to retrieve useful cases. However, due to the need for computing multiple pairwise semantic graph similarities throughout a single retrieval, an increasing size and complexity of the used graphs has a strong influence on the overall retrieval time, which in turn results in slow and unresponsive applications for relatively large graphs. The *MAC/FAC* ("Many are called, but few are chosen") approach introduced by Forbus et al. [9] can be used to counteract the previously mentioned problem of slow retrieval times. To solve this, a two-phased retrieval is applied: The first phase (MAC) utilizes a simplified and often knowledge-poor similarity measure for a fast preselection of similar cases w. r. t. the query. The second phase (FAC) then applies the computationally intensive graph-based similarity measure to the results of the MAC phase. The strategy reveals the importance of a well-chosen MAC similarity measure because the preselection of candidates must not disregard highly similar workflows to maintain an appropriate retrieval quality.

Our previous work to design MAC similarity measures shifted the focus from manually-modeled approaches [5] to approaches based on machine learning techniques [13,18]. Recently, Klein et al. [13] embedded semantic graphs into a low-dimensional vector space using the general-purpose unsupervised embedding framework *StarSpace* [26]. In this approach, the graph similarity is determined by applying a standard vector similarity measure on the generated graph embeddings. However, semantic annotations and the graph structure are not considered at all, although this is indispensable in certain domains [14,28]. In this paper, we continue to pursue the idea of automatically learned low-dimensional graph representation vectors to speed-up retrieval. We investigate two novel siamese *Graph Neural Networks* (GNNs) specifically tailored for graph structures introduced by Li et al. [15], for generating more expressive graph embeddings. We propose a generic approach to modify those GNNs to fully include semantic annotations and the workflow structure into the embedding process.

In the following section, previous work on POCBR including representation of semantic workflows, similarity assessment between these workflows, and different MAC/FAC approaches is presented. Our concept for assessing the similarity of semantic graphs with the help of GNNs is introduced in Sect. 3. Next, we apply our developed concept to cooking recipes and evaluate it. Finally, Sect. 5 concludes the results and discusses future work.

2 Foundations and Previous Work

Research on *Process-Oriented Case-Based Reasoning* (POCBR) [3,16] deals with the integration of CBR and *Process-Aware Information Systems* (PAISs) such as workflow management systems [8]. For instance, the effectiveness of POCBR

has been demonstrated by Müller [19] for assisting workflow designers during the task of workflow modeling with best-practice workflows from a case base. Thus, POCBR supports the development of workflows as an experience-based activity [3,16]. Therefore, an appropriate case representation for workflows as well as a similarity measure that assesses the suitability of a workflow w. r. t. a new problem situation is important in POCBR.

2.1 Semantic Workflow Representation

For the representation of workflows, we use semantically annotated directed graphs referred to as *NEST* graphs introduced by Bergmann and Gil [3]. 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. Hence, each node and each edge can have a semantic description.

To demonstrate the introduced representation and as part of the experimental evaluation, we use cooking recipes represented as workflows. Each workflow consists of tasks that represent cooking steps and data nodes that represent the ingredients that belong to the corresponding cooking steps. In the cooking domain, the semantic metadata language is defined by taxonomic ontologies, one for ingredients and one for cooking steps. Figure 1 shows a simple example of a *NEST* graph that represents a cooking recipe for making a sandwich. The cooking workflow contains two task nodes (`coat` and `layer`) as well as four

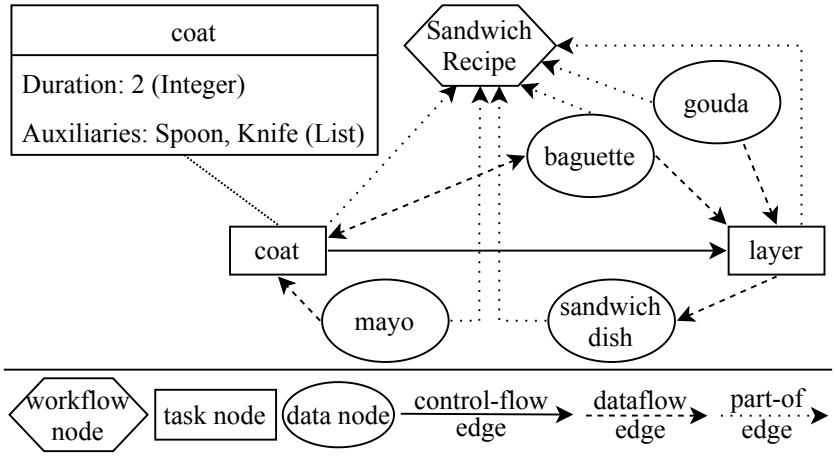


Fig. 1. Exemplary Cooking Recipe represented as NEST Graph

data nodes (`mayo`, `baguette`, `sandwich dish`, and `gouda`). Task nodes are connected by control-flow edges that define the order in which tasks are executed. Furthermore, dataflow edges are used to connect task nodes with data nodes in order to model that a task consumes inputs and produces outputs. For instance, the dataflow edge between `coat` and `layer` indicates that baguette has an interaction with the task `coat` and is consumed by the task `layer`. Semantic descriptions of task nodes and data nodes are used to further specify semantic information belonging to the workflow components in an attribute-value way. Figure 1 shows an example of the semantic description of the task node `coat`. The provided information is used to describe the task more precisely. In this case, a spoon and a baguette knife is needed to execute the task (`Auxiliaries`) and the estimated time that the task takes is two minutes (`Duration`).

2.2 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 the semantic descriptions and types of these components. Bergmann and Gil [3] propose a semantic similarity measure that determines a similarity based on the local-global principle [21]. A global similarity, i. e., the similarity between two graphs, is composed of 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 $sim_m(QW, CW)$.

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

The process of finding the mapping that maximizes the global similarity between a query QW and a single case CW is very complex due to the high number of possible mappings and thus requires solving an optimization problem. Bergmann and Gil [3] developed a parallelized version of the A^* search algorithm that can be used for finding a mapping solution by utilizing search heuristics and an adjustable A^* maximum queue size. The queue size defines the maximum number of not expanded solutions to store and influences the trade-off between quality of the mappings and time required for finding them, i. e., reducing the queue size results in solutions with worse quality at a lower computation time and vice versa. Only a queue of infinite length could deterministically find the optimal solution. Even when using the A^* search with a suitable heuristic, solving the problem to find the best-possible mapping is still very complex w. r. t. time and

memory consumption (see [20] for more details). That mainly motivates this paper.

2.3 MAC/FAC Retrieval for POCBR

In contrast to Zeyen and Bergmann [27] who tackled the aforementioned problems by optimizing the A^* search and its underlying heuristic, we used the approach of a two-phase retrieval procedure, referred to as MAC/FAC [9], to face long retrieval times [5,11,18]. It aims to decrease computation time by prefiltering the case base in order to reduce the number of cases that have to be evaluated by an often computationally complex similarity measure. The major difficulty with MAC/FAC retrieval in general is the definition of the filter condition of the MAC stage, as it has a great impact on the overall retrieval quality and performance.

Prior work of Bergmann and Stromer [5] addressed this issue by utilizing a feature-based domain specific case representation of workflows and an appropriately modeled similarity measure in the MAC stage. In order to avoid additional modeling effort, Müller and Bergmann [18] developed a MAC/FAC approach that uses a hierarchically partitioned cluster tree that can be traversed for finding clusters with cases similar to the query. This algorithm shows acceptable performance if the case base has a strong cluster structure. However, it has not reached quality and retrieval speed of the feature-based MAC/FAC approach. Our recent work [13] applied the general-purpose embedding framework StarSpace [26] to POCBR. Therefore, the authors learned vector representations in an unsupervised manner based on structural properties of workflow graphs, e. g., relation between task, data, and workflow nodes. The resulting embeddings allow to efficiently compute the similarity between a given query and a workflow from the case base by vector similarity measures, without any consideration of knowledge-intensive manually-modeled similarity measures. The approach achieves a nearly comparable performance to the feature-based MAC/FAC retrieval w. r. t. retrieval time and quality. Since the embedding-based approach does not adequately consider semantic descriptions or the graph structure that are relevant for the semantic similarity assessment, we consider this weakness as a starting point for improvements.

3 Similarity Learning for Workflow Graphs with Siamese Graph Neural Networks

This section presents our approach for generating pairwise similarities of semantic graphs by using neural networks. Since semantic labels of nodes and edges contain valuable information for similarity assessment, it is necessary to provide these semantics in combination with the workflow structure as input data for the neural networks. To the best of our knowledge, the encoding of such semantic information for learning graph similarities is a rather unexplored research area (see [20] for an overview) also in POCBR. Consequently, we present our method

for encoding semantic graphs to be used as input data of neural networks (see Sect. 3.1). Additionally, we show how this data can be used to determine graph similarities with neural networks. Therefore, two graph neural networks developed by Li et al. [15] are adjusted in order to generate these similarities (see Sect. 3.2) and to enable usage in retrieval scenarios (see Sect. 3.3).

3.1 Encoding Semantic Graphs for Similarity Learning

Our encoding scheme for *NEST* graphs creates numeric vector space encodings that can be fed into neural networks for similarity assessment. To the best of our knowledge, encoding the semantic annotations of nodes and edges is often not considered as a main aspect in papers that present novel neural network structures (e. g., [2,15]) for processing graphs. However, for semantic graphs like *NEST* graphs, the semantic annotations at nodes and edges reflect domain-specific knowledge that has a great impact on the global similarity. Thus, it is crucial that the encoding methods can transform this knowledge to vector encodings. This leads to individually created encoding schemes for node and edge *types* and their *semantic annotations*.

Encoding Node and Edge Types Properly encoding node and edge types is important because, during similarity assessment, only nodes and edges with identical types are mapped (see Sect. 2.2). The types are encoded separately for nodes and edges by *one-hot encodings*. One-hot encoding vectors encode information in binary form by setting a single element as a `1` while all other vector elements are set to `0`. This way, a single one-hot encoding vector can only have as many different value allocations as it has vector elements. For *NEST* graphs, there are four different one-hot encodings of node types and edge types each (see Fig. 2). The main advantage is that all encodings are clearly distinguishable by a neural network that allows suitable processing of these vectors.

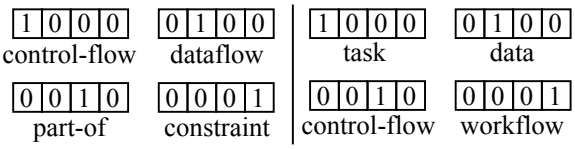


Fig. 2. One-Hot Encodings of Edge Types (left) and Node Types (right)

Encoding Semantic Descriptions Encoding the semantic descriptions of nodes and edges requires the transformation of several data types and complex data relations. The data types that can be used inside of a semantic description are not specified in detail according to the *NEST* graph publication [3].

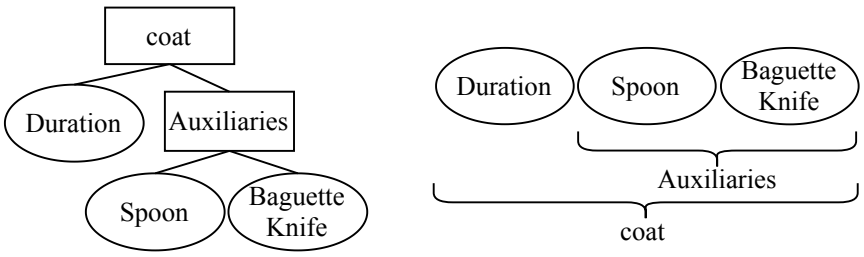
Referring to the ProCAKE framework[3] [4] that fully supports *NEST* graphs, a semantic description is composed of *atomic data types* and *composite data types*: Atomic types comprise *integer*, *double*, *boolean*, *string*, *void*, and *time* and composite data types consist of *list*, *set*, and *attribute-value pairs*. Each of these data types requires an individual encoding approach in order to map the semantics of the data to the encoding vectors as fully as possible. We implemented these individual encoding algorithms but due to space restrictions in this paper, only the general encoding approach of atomic and composite types is presented.

Each atomic type is encoded to a single encoding vector with length α that is part of the vector space $\mathbb{R}^{\alpha}$. Thereby, these encoding vectors are made up of two subvectors, where the first one encodes the atomic data type (e. g., string, boolean, double) and the second one encodes the actual value to encode (e. g., "Hello World", *true*, 1.0). The additional encoding of the used atomic data type serves as further semantic information for the neural networks that process the encoding vectors. By that, encodings of semantic description entries of different types can be distinguished more easily although they are mapped to a common vector space. Encoding composite types is not as straightforward as encoding atomic types due to the complexity of semantic descriptions. The task node `coat` from Fig. 1, for instance, is made up of composite attribute-value pairs with three entries. The entry `Duration` is an atomic type and the entry `Auxiliaries` is a list of atomic strings (composite type) that is nested inside of the attribute-value pair. This arrangement of composite types nesting other atomic or composite types can be visualized as a *tree structure* (see Fig. 3a). Regarding the transformation of these semantic descriptions to numeric vectors, this means that the encoding of a composite type aggregates the encodings of the nested types. However, using tree-structured data in a neural network requires the definition of complex layers and very special additional encoding schemes (e. g., [22,24]). In order to simplify the tree structures to encode, each composite type is redefined as a *sequence of atomic types* (see Fig. 3b). In case a composite type contains another composite type (e. g., the tree structure in Fig. 3a), the encoding sequences of parent type and child type are computed recursively and put together to a single sequence. Converting tree structures to sequences is in particular motivated by techniques that are used in natural language processing (e. g., [7,23]), where a word or a sentence is often represented as a sequence of encoding vectors. Similar to the previously mentioned approaches, we also use *Recurrent Neural Networks* (RNNs) to process these sequences, which is described in the next section.

3.2 Similarity Learning for Workflow Graphs

The neural networks that are used for graph retrieval are based on the *Graph Embedding Model* (GEM) and the *Graph Matching Network* (GMN) introduced by Li et al. [15]. In their work, they present two neural networks that are capable

[3] https://procake.uni-trier.de



(a) Tree Arrangement **(b)** Sequence Arrangement

Fig. 3. Tree and Sequence Arrangement of Semantic Description Components

of learning to assess the similarity of graphs. Both neural networks compute compact vector representations of graphs that can be eventually compared using a vector similarity measure. Thereby, the GEM is designed to enable a lightweight, fast similarity assessment, whereas the GMN is optimized to learn more expressive similarity patterns on the pairwise graph features. Both neural networks feature three main components (see Fig. 4): the *encoder*, the *propagation layer*, and the *aggregator*. We completely reuse the propagation layer of both networks from the original implementation of Li et al.[4] and adjusted the encoder and the final graph similarity for our application scenario.

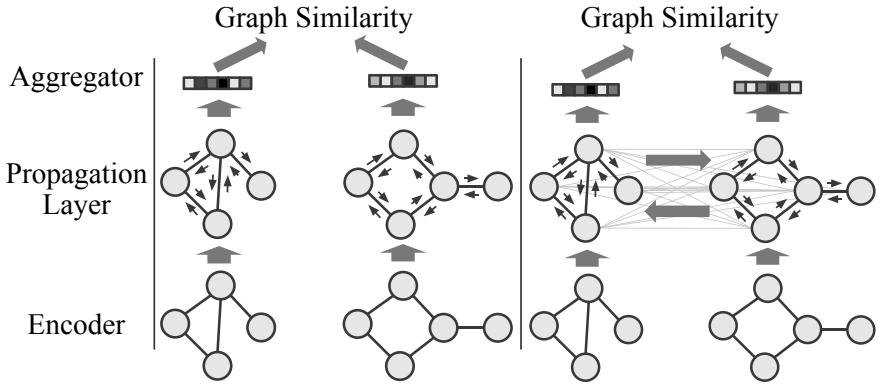


Fig. 4. GEM (left) and GMN (right) (based on [15])

The *encoder* transforms the raw graph input data into a first embedding of all nodes and edges. The components that are embedded in this process cover the semantic descriptions and types allocated to each node and edge (see Sect.

[4] https://github.com/deepmind/deepmind-research/tree/master/graph_matching_networks

2). These components are embedded independent of each other and concatenated afterwards to a single embedding vector for each node and edge of the graph. This way, separate neural network structures can be adequately trained to generate suitable embeddings. Whereas the embeddings of the types are generated by using separate feed-forward networks for node and edge types, the embedding procedure of the semantic descriptions utilizes an RNN [7]. An RNN is specifically designed to handle sequences of inputs as they are present in the encodings of semantic descriptions. Please note that the influence of both parts of the embedding vector can be controlled by manipulating the respective vector lengths.

The *propagation layer* iteratively combines the node embeddings according to the edge structure of the graph in order to capture information on the local neighborhood in each node's embedding. Therefore, an iterative process is used that updates the node embeddings in multiple steps. In each step, the embedding of a single node is updated by merging the node's embedding with the embeddings of nodes that are connected via an edge. This enables information to be distributed by the node embeddings across the propagation steps. The definition of a node's neighborhood is the main difference of the propagation approach present in GEM and GMN. As depicted in Fig. 4, the GEM only propagates information within a single graph. This means that a node's embedding vector is updated according to all nodes that are connected via an edge that only allows information flow within a single graph. In contrast, the GMN also accumulates information across both graphs during the similarity assessment by using an attention-based matching component. This enables information to distribute between both graphs in an early state of similarity computation that contributes to the increased expressiveness of GMN compared to GEM.

After iteratively propagating information within the graphs, the *aggregator* merges the final node embeddings of all graph nodes to form an embedding for the whole graph. The embeddings of each of the two graphs are then used to determine a *graph similarity* value in [0,1]. Therefore, we use cosine similarity for the GEM and a feed-forward neural network layer for the GMN. Given two graph embedding vectors, the cosine similarity is defined to be the dot product of these two vectors, divided by the product of the Euclidian vector length of both vectors. This leads to a computationally inexpensive way of generating the final graph similarity value. The feed-forward neural network that is used to compute the final similarity for the GMN can be trained in order to learn the characteristics of the whole-graph embedding vectors resulting from the aggregator. Thus, this process is more expressive than using a vector similarity measure, at the expense of a higher computation effort.

The *training* procedure for both networks utilizes the gradient-descent-based optimizer Adam [12] in a mini-batch setup. Each training graph pair from the batch of training examples is labeled with the ground truth similarity value that is determined using the semantic similarity measure *sim* introduced in Sect. 2.2. This data is used to compute the *Mean Squared Error* (MSE) that serves as a differentiable loss function. The MSE sums up all squared differences of

the similarity predicted by the neural network and the labeled similarity, and then divides by the amount of all batched training examples to get the average deviation.

3.3 Siamese GNN-Based Workflow Retrieval

In a workflow retrieval, the k-most similar cases CW are retrieved from the case base CB, according to the similarity to a query workflow QW, i. e., $sim(QW, CW)$. The neural networks can be integrated into this process as the similarity measure for generating the pairwise graph similarities, i. e., $fsim(QW, CW)$. Therefore, the query workflow and all cases from the case base have to be encoded to a numerical vector format first (see Sect. 3.1). After that, an offline training session can be started that trains the neural networks GEM and GMN for predicting the similarities of all cases from the case base. The machine learning framework *Tensorflow*[5] [1] is utilized for this purpose. Given the encoded graphs and the trained neural networks, either GEM or GMN can be used to determine the pairwise similarities for the query and all cases from the case base, i. e., $sim(QW, CW)$ is approximated by $fsim(QW, CW)$. Eventually, the retrieval result is finalized by putting together the k-most similar workflows according to the similarity computed by the neural network.

4 Experimental Evaluation

To evaluate our approach, we measure performance and quality of GEM and GMN in different retrieval scenarios. Thereby, both neural networks are compared to the feature-based retriever by Bergmann and Stromer [5] (FBR), to the latest embedding-based retriever by Klein et al. [13] (EBR), and to the A^*-retriever by Bergmann and Gil [3] (A*R). We investigate the following hypotheses in two experiments:

- **H1** Using GEM and GMN as a MAC retriever of a MAC/FAC retrieval leads to better retrieval results than using EBR as MAC retriever.
- **H2** The GMN retriever is able to approximate the ground-truth graph similarities better than A*R, using parameter settings such that the retrieval time of both retrievers is comparable.

The first experiment examines the retrievers in a MAC/FAC setup, where the focus is put on the suitability of GEM and GMN as a MAC similarity measure (see H1). The second experiment examines to which degree the retrievers are capable of approximating the ground-truth A^*-similarities (see H2).

4.1 Experimental Setup

In the evaluation, workflows representing cooking recipes [4] and workflows representing Data Mining processes from RapidMiner[6] [28] are examined, with a

[5] https://tensorflow.org/
[6] https://rapidminer.com/

training and a testing case base for each domain. The cooking workflows (CB-I) are derived from 40 manually-modeled cooking recipes that are extended to 800 workflows by previously developed adaptation methods [19], resulting in 680 training cases and 120 testing cases. The workflows of the Data Mining domain (CB-II) are built from sample processes that are delivered with RapidMiner, resulting in 529 training cases and 80 testing cases. We build these different case bases in order to investigate if our approach performs differently regarding the complexity of the workflow domains. Therefore, we evaluate on the cooking workflows with rather simple semantic descriptions and on the RapidMiner workflows with more complex semantic descriptions.

The metrics that are used to evaluate our approach cover performance and quality. The performance is measured by taking the retrieval time in seconds. The quality of the results to evaluate RL_{eval} is measured by comparing them to the ground-truth retrieval results RL_{true} in terms of Mean Absolute Error (MAE), correctness (see [6] for more details), and k-NN quality (see [13] and [18] for more details). The MAE (ranged between 0 and 1) expresses the average similarity error between all pairs of query workflow and case workflow in RL_{true} and the same pairs in RL_{eval}. The correctness (ranged between -1 and 1) describes the conformity of the ranking positions of the workflow pairs in RL_{eval} according to RL_{true}. 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 RL_{eval} although WP_2 is ranked before WP_1 in RL_{true} or vice versa. The k-NN quality (ranged between 0 and 1) quantifies to which degree highly similar cases according to RL_{true} are present in RL_{eval}. Therefore, the $|\, RL_{\text{eval}} \,|$ most-similar cases from RL_{true} are compared with RL_{eval}. Each case from the most-similar cases that is missing in RL_{eval} decreases the quality, with highly relevant cases affecting the quality stronger than less relevant cases.

All experiments are computed on a PC with an Intel i7 6700 CPU (4 cores, 8 threads) and an NVIDIA GTX 1080 GPU with 16 GB RAM, running Windows 10 64-bit. The retrievers (EBR, GEM, and GMN) that require an offline training phase are trained with the two training case bases, resulting in two models per retriever, i. e., one for each domain. The training time for GEM on both case bases is approx. 12 hours, for GMN approx. 18 hours, and for EBR approx. 6 minutes. Each retriever uses all processing cores of CPU or GPU for calculating the similarities. A retrieval is always conducted with a query from the testing case base and with the cases from the training case base. To produce meaningful performance and quality values, the results of the retrieval runs of all query cases from a single domain are averaged.

4.2 Experimental Results

The first experiment evaluates our neural networks as retrievers in a scenario of MAC/FAC retrieval. Table 1 shows the results (k-NN quality and retrieval time) as compared with FBR and EBR, since these two retrievers are specifically designed for MAC/FAC applications. For CB-I, FBR outperforms all other retrievers w. r. t. quality for all combinations of FS and k. EBR and GMN have

Table 1. Evaluation Results of MAC/FAC Experiment

			GMN		GEM		FBR		EBR	
	FS	k	Quality	Time	Quality	Time	Quality	Time	Quality	Time
CB-I	5	5	0.534	07.91	0.521	00.27	0.598	00.86	0.552	00.29
	50	5	0.564	10.31	0.535	02.52	0.704	03.22	0.639	02.55
	10	10	0.581	08.19	0.550	00.53	0.647	01.11	0.599	00.56
	80	10	0.641	11.81	0.572	03.99	0.749	04.76	0.697	03.98
	25	25	0.649	09.02	0.600	01.31	0.721	01.92	0.658	01.34
	100	25	0.727	12.83	0.628	04.95	0.814	05.82	0.748	04.91
CB-II	5	5	0.584	08.48	0.356	00.19	0.659	00.43	0.348	00.13
	50	5	0.861	10.78	0.508	02.54	0.909	02.10	0.483	01.80
	10	10	0.625	08.60	0.408	00.42	0.658	00.57	0.384	00.25
	80	10	0.875	12.69	0.550	03.99	0.916	03.85	0.550	03.60
	25	25	0.718	08.00	0.474	01.13	0.696	01.10	0.450	00.97
	100	25	0.895	14.14	0.602	04.62	0.887	05.13	0.585	04.60

quality values in a similar range, with both consistently outperforming the quality values of GEM. When only considering the time, EBR and GEM clearly outperform all other retrievers. For CB-II, the quality values of GMN outperform those of EBR and are in a similar range as those of FBR. The quality values of GEM surpass those of EBR with comparable retrieval times. The results show that the suitability of GEM and GMN increases for retrieval situations with more complex semantic descriptions of task and data nodes, as present in CB-II. The performance of GEM and GMN for retrieving graphs from a rather simple domain, such as those of CB-I, is respectable but does not lead to a replacement of current approaches (EBR and FBR). Anyhow, the FBR with its manually-modeled similarity measure still performs best for both case bases, taking into account the combination of quality and time. When only looking at the automatically-learned retrievers in the results for CB-II, i. e., GEM, GMN, and EBR, GEM is the most suitable for a MAC/FAC scenario since it shows a good combination of very low retrieval times and high quality values. Thus, H1 is partly confirmed due to different results for the two case bases. The results for CB-I do not confirm H1 since EBR outperforms GEM in terms of quality and even though GMN shows better quality results than EBR, it has infeasible retrieval times for a MAC/FAC setup. For CB-II, H1 can be clearly accepted due to higher quality values with approximately equal retrieval times, when comparing GEM and EBR.

The second experiment examines to which degree GEM, GMN, EBR, and FBR are able to approximate the ground-truth graph similarities. Since GMN and GEM are evaluated as MAC retrievers in the first experiment, the second experiment focuses more on the suitability as FAC retrievers by measuring the prediction errors (see Tab. 2). Therefore, we compare all previously mentioned retrievers to a variant of the A*R with an adjusted queue size (see Sect. 2.2) so that the retrieval time of A*R is approx. equal to that of GMN. Aligning the

retrieval times of A*R and GMN enables a fair comparison of the resulting MAE and correctness. For CB-I, GMN has the lowest MAE and A*R has the highest

Table 2. Evaluation Results of A-Star Approximation Experiment

	Retriever	MAE	Correctness	Time
CB-I	A*R	0.054	0.753	8.203
	GMN	0.049	0.479	7.612
	GEM	0.123	0.231	0.008
	FBR	0.187	0.646	0.539
	EBR	0.354	0.190	0.006
CB-II	A*R	0.040	0.778	9.520
	GMN	0.021	0.797	8.444
	GEM	0.170	0.064	0.006
	FBR	0.199	0.580	0.350
	EBR	0.404	0.064	0.004

correctness. FBR achieves a high level of correctness but lags behind in terms of MAE. When comparing the results of CB-I and CB-II, it becomes apparent that GMN still has the lowest MAE and now also has the highest value of correctness. This leads to the assumption that the suitability of GMN increases with more complex cases. The reason for that might be the different levels of computational complexity of both retrievers, i. e., exponential complexity for A*R and quadratic complexity for GMN. GMN outperforming A*R in terms of MAE is even more remarkable when considering that GMN learns to assess the similarity of graphs without knowing the original algorithmic context, e. g., similarities of semantic descriptions or node and edge mappings. Additionally, this experiment shows that FBR and EBR are not suitable for generating similarities that are close to the ground-truth similarities. The reason for this could be the inadequate processing of semantic annotations and the workflow structure. Thus, we clearly accept H2 for CB-II and partly accept this hypothesis for CB-I.

5 Conclusion and Future Work

This paper examines the potential of using two siamese GNNs in a retrieval scenario in POCBR. Therefore, an encoding scheme is presented that covers the workflow structure, the types of nodes and edges, and their semantic descriptions. The encoded workflows are furthermore processed by two neural networks GEM and GMN that are adjusted and optimized for being used in retrieval scenarios. The evaluation of both neural networks investigates how both approaches perform in being used in a MAC/FAC setup and in approximating the ground-truth similarities of the graph similarity measure. Compared to previous retriever approaches, the results show great potential: GEM is suitable for

a MAC/FAC setup, due to its fast similarity computation and reasonable retrieval quality. Furthermore, GMN shows great potential in approximating the ground-truth graph similarities.

A focus of future research should be on optimizing the presented approach of a GNN-based retrieval. This optimization ranges from aspects of parameterization to adjustments of the data encoding scheme and the usage of different neural network structures. The neural network structures could be optimized to better process other graph domains, e. g., argument graphs [14], or even other types of complex similarity measures [17]. Two more optimizations could be, for instance, using a differentiable ranking loss function that optimizes according to the ground-truth ordering of the retrieval results (e. g., [25]) or considering the relationships between different graphs from a case base during training (e. g., Neural Structured Learning[7]). Furthermore, the neural networks that are used in this work are not capable of explaining the results they produce, i. e., black boxes. In current research (also in the CBR community, e. g., [10]), this lack of explainability is tackled in the context of Explainable Artificial Intelligence (XAI). Future research should address this issue by investigating which methods are suitable for increasing the explainability of the presented neural networks.

Acknowledgments. This work is funded by the German Research Foundation (DFG) under grant No. BE 1373/3-3 and grant No. 375342983.

References

1. Abadi, M., et al.: TensorFlow: Large-Scale Machine Learning on Heterogeneous Distributed Systems. CoRR **abs/1603.04467** (2016)
2. Bai, Y., et al.: SimGNN: A Neural Network Approach to Fast Graph Similarity Computation. In: Proc. of the 12th ACM Int. Conf. on Web Search and Data Mining 2019, Australia. pp. 384–392. ACM (2019)
3. Bergmann, R., Gil, Y.: Similarity assessment and efficient retrieval of semantic workflows. Inf. Syst. **40**, 115–127 (2014)
4. Bergmann, R., Grumbach, L., Malburg, L., Zeyen, C.: ProCAKE: A Process-Oriented Case-Based Reasoning Framework. In: Workshops Proc. for the 27th Int. Conf. on Case-Based Reason. Res. and Dev. CEUR Workshop Proc., vol. 2567, pp. 156–161. CEUR-WS.org (2019)
5. 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)
6. Cheng, W., Rademaker, M., Baets, B.D., Hüllermeier, E.: Predicting Partial Orders: Ranking with Abstention. In: Machine Learning and Knowledge Discovery in Databases, Part I. LNCS, vol. 6321, pp. 215–230. Springer (2010)
7. Cho, K., et al.: Learning Phrase Representations using RNN Encoder-Decoder for Statistical Machine Translation. In: Proc. of the Conf. on Empirical Methods in Natural Language Processing, EMNLP 2014, Qatar. pp. 1724–1734. ACL (2014)
8. Dumas, M., v. d. Aalst, W.M.P., ter Hofstede, A.H.M.: Process-Aware Information Systems: Bridging People and Software Through Process Technology. Wiley (2005)

[7] https://www.tensorflow.org/neural_structured_learning

9. Forbus, K.D., Gentner, D., Law, K.: MAC/FAC: A Model of Similarity-Based Retrieval. Cogn. Sci. **19**(2), 141–205 (1995)
10. Keane, M.T., Kenny, E.M.: How Case-Based Reasoning Explains Neural Networks: A Theoretical Analysis of XAI Using Post-Hoc Explanation-by-Example from a Survey of ANN-CBR Twin-Systems. In: Case-Based Reason. Res. and Dev. - 27th Int. Conf. LNCS, vol. 11680, pp. 155–171. Springer (2019)
11. Kendall-Morwick, J., Leake, D. B.: A study of two-phase retrieval for process-oriented case-based reasoning. In: Successful Case-based Reasoning Applications-2, pp. 7–27. Springer (2014)
12. Kingma, D.P., Ba, J.: Adam: A Method for Stochastic Optimization. In: 3rd Int. Conf. on Learning Representations, ICLR 2015, USA, Conf. Track Proc. (2015)
13. Klein, P., Malburg, L., Bergmann, R.: Learning Workflow Embeddings to Improve the Performance of Similarity-Based Retrieval for Process-Oriented Case-Based Reasoning. In: Case-Based Reason. Res. and Dev.: 27th Int. Conf., ICCBR 2019, Germany, Proc. pp. 188–203. Springer. (2019)
14. Lenz, M., Ollinger, S., Sahitaj, P., Bergmann, R.: Semantic Textual Similarity Measures for Case-Based Retrieval of Argument Graphs. In: Case-Based Reason. Res. and Dev. - 27th Int. Conf. LNCS, vol. 11680, pp. 219–234. Springer (2019)
15. Li, Y., Gu, C., Dullien, T., Vinyals, O., Kohli, P.: Graph Matching Networks for Learning the Similarity of Graph Structured Objects. In: Chaudhuri, K., Salakhutdinov, R. (eds.) Proc. of the 36th Int. Conf. on Machine Learning, ICML 2019, USA. Proc. of Machine Learning Research, vol. 97, pp. 3835–3845. PMLR (2019)
16. Minor, M., Montani, S., Recio-García, J.A.: Process-oriented Case-based Reasoning. Inf. Syst. **40**, 103–105 (2014)
17. Mougouie, B., Bergmann, R.: Similarity Assessment for Generalizied Cases by Optimization Methods. In: Adv. in Case-Based Reas., 6th Europ. Conf., ECCBR. LNCS, vol. 2416, pp. 249–263. Springer (2002)
18. Müller, G., Bergmann, R.: A Cluster-Based Approach to Improve Similarity-Based Retrieval for Process-Oriented Case-Based Reasoning. In: ECAI 2014 - 21st Europ. Conf. on Artif. Intell. pp. 639–644. IOS Press (2014)
19. Müller, G.: Workflow Modeling Assistance by Case-based Reasoning. Springer Fachmedien Wiesbaden (2018)
20. Ontañón, S.: An overview of distance and similarity functions for structured data. Artif Intell Rev (2020)
21. Richter, M.M.: Foundations of Similarity and Utility. In: Proc. of the 20th Int. Florida Artif. Intell. Res. Society Conf. pp. 30–37. AAAI Press (2007)
22. Socher, R., Lin, C.C., Ng, A.Y., Manning, C.D.: Parsing Natural Scenes and Natural Language with Recursive Neural Networks. In: Proc. of the 28th Int. Conf. on Machine Learning, ICML 2011, USA. pp. 129–136. Omnipress (2011)
23. Sutskever, I., Vinyals, O., Le, Q.V.: Sequence to Sequence Learning with Neural Networks. In: Adv. in Neural Inf. Process. Syst. 27. pp. 3104–3112 (2014)
24. Tai, K.S., Socher, R., Manning, C.D.: Improved Semantic Representations From Tree-Structured Long Short-Term Memory Networks. In: Proc. of the 53rd Ann. Meeting of the Assoc. for Comput. Linguist. and the 7th Int. Joint Conf. on NLP of the Asian Federation. pp. 1556–1566. The Assoc. for Comput. Linguist. (2015)
25. Taylor, M.J., Guiver, J., Robertson, S., Minka, T.: SoftRank: optimizing non-smooth rank metrics. In: Proc. of the Int. Conf. on Web Search and Web Data Mining, WSDM 2008, USA. pp. 77–86. ACM (2008)
26. Wu, L.Y., Fisch, A., Chopra, S., Adams, K., Bordes, A., Weston, J.: StarSpace: Embed All The Things! In: Proc. of the 32nd AAAI Conf. on Artif. Intell., USA, 2018. pp. 5569–5577. AAAI Press (2018)

27. Zeyen, C., Bergmann, R.: A*-Based Similarity Assessment of Semantic Graphs. In: Case-Based Reason. Res. and Dev.: 28th Int. Conf., ICCBR 2020, LNCS, vol. 12311, pp. 17–32. Springer (2020)
28. 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.: 27th Int. Conf., ICCBR 2019. LNCS, vol. 11680, pp. 388–403. Springer (2019)