Enhanced GAT: Expanding Receptive Field with Meta Path-Guided RDF Rules for Two-Hop Connectivity

Julie Loesch[1], Michel Dumontier[1] and Remzi Celebi[1]

[1]*Department of Advanced Computing Sciences, Maastricht University, Paul-Henri Spaaklaan 1, Maastricht, 6229 EN, Netherlands*

Abstract

Neuro-Symbolic Artificial Intelligence is an emerging field that combines neural networks and symbolic reasoning to tackle complex tasks, such as ontology reasoning, i.e. inferring new facts that are not expressed in an ontology. However, the integration of symbolic reasoning in neural networks for efficient reasoning over very large and complex ontologies still remains relatively unexplored. Therefore, this paper introduces a scalable neural-symbolic method called *2-Hop GAT* for reasoning over large and complex ontologies, which is an extension of the Graph Attention Network (GAT), leveraging meta paths of two hop to capture transitivity. By extending GAT to include nodes that are two hops away, the proposed method achieves enhanced reasoning capabilities. Additionally, the *Filtered 2-Hop GAT* variant is presented, which adds a filtering mechanism to guide meta paths of two hop to include two RDF rules. Namely, (1) subclass transitivity: if A is a subclass of B, and B is a subclass of C, then A is also a subclass of C, and (2) if A is a type of B and B is a subclass of C, then A is also a type of C. This paper reports experimental results using the datasets from the SemRec Challenge at ISWC 2023, demonstrating the effectiveness of the proposed methods. The latter approach shows promising results, achieving a Hits@5 score of 0.752 and a Hits@10 score of 0.803 for the class subsumption task.

Keywords

Neuro-Symbolic AI, Ontology Reasoner, Graph Neural Networks

1. Introduction

In general, a Knowledge Graph (KG), which is a graph composed of a set of edges labeled with relations that are expressed between entities, is considered to be incomplete and subject to inconsistency. KG reasoning aims to address these issues and refers to inferring new facts from existing ones by combining logical rules and constraints with existing facts in a KG [1]. Recent developments in deep learning have promoted neural reasoning over knowledge graphs including KG embeddings [2, 3] and Graph Neural Networks [4, 5] due to their ability to handle noisy data and incorporate background knowledge [6]. Graph Neural Networks (GNNs) are special types of neural networks for Knowledge Graphs (KGs) [7] and can capture structural information inherently stored in these KGs, showing to achieve state-of-the-art performance in graph representation learning.

SemREC'23: Semantic Reasoning Evaluation Challenge, ISWC'23, Nov 6 – 10, City of Athens, Greece
julie.loesch@maastrichtuniversity.nl (J. Loesch); michel.dumontier@maastrichtuniversity.nl (M. Dumontier); remzi.celebi@maastrichtuniversity.nl (R. Celebi)
0000-0003-4727-9435 (M. Dumontier); 0000-0001-7769-4272 (R. Celebi)

A KG can take the form of an ontology, which is a formal and explicit specification of concepts, relationships, constraints and rules within a specific domain and represents more complex relationships (i.e. negation, conjunction, disjunction). However, the use of GNNs for efficient reasoning over very large and complex ontologies remains relatively unexplored. One possible reason is that GNNs, in their current form, operate in a data-driven manner and lack explicit symbolic reasoning capabilities that are essential for ontologies. The receptive field of a single GNN is restricted to one-hop network neighborhoods, which means that a node can only attend to its immediate neighbors to compute its next layer representation. Although stacking multiple GNN layers can enlarge the receptive field, such deeper networks are prone to over-smoothing such that it may lead to a model where node representations converge to indistinguishable vectors (i.e., all node embeddings converge to the same value) [6].

In this work, we propose *2-Hop Graph Attention Network (GAT)*, a scalable and efficient neural-symbolic method to reason over large and complex ontologies with a focus on transitivity, which is a fundamental property ontology reasoning. *2-Hop GAT* is an extension of GAT [5], a neural network architecture that leverages masked self-attentional layers. In this way the receptive field includes nodes that are two hops away. We also explore a *Filtered 2-Hop GAT*, which adds a filtering mechanism such that meta paths of *two hop* are guided to include two RDF rules [8] and to capture transitivity, which is a fundamental property for ontology reasoning. The following rules are comprised: (1) subclass transitivity: if A is a subclass of B, and B is a subclass of C, then A is also a subclass of C, and (2) if A is a type of B and B is a subclass of C, then A is also a type of C. Evaluated against CaLiGraph10e5, the system achieves promising results for the class subsumption reasoning task with a Hits@5 score of 0.752 and Hits@10 score of 0.803.

The main contributions of this paper are:

- Development of *2-Hop GAT* and *Filtered 2-Hop GAT* which do not generate random walks as for *metapath2vec*, thereby significantly reducing time complexity.
- Including symbolic reasoning to Graph Neural Networks in the form of a filtering mechanism to add two RDF rules (i.e., RDFS entailment rules[1]).
- Application of Graph Neural Networks to address the challenges of two key ontology reasoning tasks, namely subclass and type reasoning tasks.

The remainder of the paper is organized as follows: previous related studies and the datasets provided by the SemRec Challenge at ISWC 2023 are presented in Section 2 and Section 3, respectively. Section 4 introduces the neural-symbolic methods for reasoning over very large ontologies. Section 5 explains the experimental setup and reports the obtained results, which is followed by a discussion in Section 6. Finally, Section 7 concludes this paper. The implementation to generate the analysis is made available at our Github repository: https://github.com/jloe2911/SemREC2023.

2. Related Work

Wang et al. [9] proposed Multi-hop Attention Graph Neural Network (MAGNA), which is an effective multi-hop self-attention mechanism for graph-structured data. MAGNA captures the

[1] https://www.w3.org/TR/rdf-mt/#rules

long-range interactions between nodes that are not directly connected but may be multiple hops away. In contrast, our proposed model simply enlarges the receptive field to nodes that are *two hops* away using Graph Attention Network [5].

Yuxiao et al. [10] propose *metapath2vec* for scalable representation learning in heterogeneous networks. *Metapath2vec* aims to maximize the likelihood of preserving both the structures and semantics of a given heterogeneous graph by simultaneously learning the low-dimensional and latent embeddings for multiple types of nodes and edges. The approach is based on meta-path-based random walks; however, Sun et al. [11] showed that heterogeneous random walks are biased toward types of nodes with a dominant number of paths and concentrated nodes, which have a governing percentage of paths pointing to a small set of nodes. To address this issue, the authors demonstrated how to use meta-paths to guide heterogeneous random walkers. Thus, in *metapath2vec*, the random walk is driven by a meta-path that defines the node type order. The random walks are then used to learn an embedding vector for each node in the graph, employing the Word2Vec algorithm [12]. To overcome the mentioned bias problem and at the same time, to integrate symbolic reasoning into Graph Neural Networks, we added a filtering mechanism such that meta-paths of *two hop* are guided to include two RDF rules [8].

Multi-hop has also been used in other neural reasoning methods than GNNs. Particularly, Mehryar and Celebi [13] extended TransE and rTransE to make subsumption and instance-checking reasoning possible by leveraging transitive relations. The authors further improved the quality of the embeddings using multi-hop samples generated by an agent's policy network. The agent is a neural network that takes as input an entity vector embedding (i.e., state) and outputs a relationship (i.e., action). The agent learns a good policy to choose actions that lead to a more meaningful and longer sequence of translations. Employing rTransE on the CaLiGraph10e5 dataset provided by the SemRec Challenge 2022, they achieved a Hits@5 score of 0.678 and a Hits@10 score of 0.713 for the class subsumption task, and a Hits@5 score of 0.017 and a Hits@10 score of 0.095 for the class membership task.

3. Datasets

This paper utilizes three ontologies that were provided by the SemRec Challenge at ISWC 2023, which are OWL2Bench [14], ORE [15], and CaLiGraph [16]. Usually, an ontology comprises a Terminological Box (Tbox) and an Assertional Box (Abox) axioms. The Tbox axioms describe the relationships between classes, such as subclass relationships and property restrictions. The Abox axioms represent individuals that are instances of the classes defined in the Tbox. Table 1 lists the frequency of each axiom. The OWL2Bench and ORE Tboxes encompass a wide range of axioms, including Class Expression Axioms, Object Property Axioms, Data Property Axioms, and Assertions, among others. In contrast, CaLiGraph's Tbox is primarily composed of class restrictions. In addition, the number of nodes/classes and edges is reported in Table 2, and Table 3 indicates the number of edges for each edge type and their proportion.

OWL2Bench [14] is a benchmark for evaluating the coverage, scalability, and query performance of ontology reasoners across the four OWL 2 profiles (EL, QL, RL, and DL). The authors extended the well-known University Ontology Benchmark (UOBM) to create four TBoxes for each of the four OWL 2 profiles. Furthermore, OWL2Bench has an ABox generator and comes

with a set of 22 SPARQL queries that involve reasoning. OWL2Bench is provided in two separate sub-datasets. OWL2Bench1 includes $7,989$ (99%) assertion relations in the training data and $2,283$ (99%) in the test data, as well as 105 (1%) subclass relations in the training data and 30 (1%) in the test data. OWL2Bench2, which is approximately twice the size of OWL2Bench1, contains $15,526$ (99%) assertion relations in the training data and $4,437$ (99%) in the test data, as well as 105 (1%) subclass relations in the training data and 30 (1%) in the test data. In addition, both sub-datasets have very few equivalence relations between a pair of classes (i.e., # of all other edges). Thus, OWL2Bench1 and OWL2Bench2 mainly comprise assertion edges (i.e., around 99%).

The second ontology that was provided by the challenge is OWL Reasoner Evaluation (ORE) 2014 [15]. The dataset is larger than OWL2Bench but similar in the sense that it includes mainly subclass and assertion relations, and only very few equivalence relations between a pair of classes. Compared to OWL2Bench, ORE comprises more subclass edges (i.e., around 13%) and fewer assertion relations (i.e., around 87%). ORE 2014 was issued by the OWL Reasoner Evaluation (ORE) competition, which is an annual competition (with an associated workshop) that pits OWL 2 compliant reasoners against each other on various standard reasoning tasks over naturally occurring problems.

Finally, the third dataset that we used is CaLiGraph, which was curated by Heist and Paulheim [16]. The dataset was generated from Wikipedia by exploiting the category system, list pages, and other list structures in Wikipedia, containing more than 15 million typed entities and around 10 million relation assertions. In addition, CaLiGraph has a large ontology, comprising more than $200,000$ class restrictions. The authors also introduced differently sized subsets of CaLiGraph, allowing for performing scalability experiments. Since CaLiGraph has a large ABox and ontology, it is an interesting benchmark dataset able to measure the trade-off between scalability and accuracy. In contrast to the two previously mentioned datasets, CaLiGraph is more challenging because it contains more relation types (i.e., edges that represent simple facts) and poses high scalability requirements.

4. Methodology

This section introduces *2-Hop Graph Attention Network* (2-Hop GAT) and *Filtered 2-Hop Graph Attention Network* (Filtered 2-Hop GAT), whose primary goal is to reason over two types of ontology relations, namely subclass and type reasoning. Subsection 4.1 provides a short background on Graph Attention Network (GAT). Subsection 4.2 presents *2-Hop GAT*, which is an extension of GAT that follows a message passing schema to update node representations using information from nodes that are *two hops* away. To add explicit symbolic reasoning to 2-Hop GAT, which is essential for ontology reasoning, we proposed *Filtered 2-Hop GAT* presented in Subsection 4.3. In short, *Filtered 2-Hop GAT* adds a filtering mechanism that guides meta paths of two hop to include two RDF rules [8]. Specifically, (1) subclass transitivity: if A is a subclass of B, and B is a subclass of C, then A is also a subclass of C, and (2) if A is a type of B and B is a subclass of C, then A is also a type of C.

Table 1
Number of axioms.

	O2B1	O2B2	ORE1	ORE2	ORE3
Class Expression Axioms					
Subclass Axioms	122	122	14,768	14,803	14,749
Equivalent Classes	21	21	39	39	39
Disjoint Classes	11	11	113	113	113
Disjoint Union of Class Expressions	9	9	-	-	-
Object Property Axioms					
Object Subproperties	63	63	349	350	349
Equivalent Object Properties	4	4	-	-	-
Inverse Object Properties	22	22	175	175	175
Object Property Domain	62	62	740	743	740
Object Property Range	57	57	691	693	691
Functional Object Properties	2	2	18	18	18
Irreflexive Object Properties	2	2	-	-	-
Symmetric Object Properties	2	2	24	24	24
Transitive Object Properties	5	5	27	27	27
Data Property Axioms					
Data Subproperties	2	2	-	-	-
Data Property Domain	7	7	59	60	59
Data Property Range	-	-	69	70	69
Functional Data Properties	3	3	3	3	3
Assertions					
Individual Equality	2	2	-	-	-
Individual Inequality	4	4	-	-	-
Class Assertions	3,885	7,625	8,675	8,684	8,661
Positive Object Property Assertions	27,794	55,090	3,240	3,244	3,225
Negative Object Property Assertions	2	2	-	-	-
Positive Data Property Assertions	18,446	36,363	665	668	665
Annotations	3	3	-	-	-

Table 2
Number of nodes and edges.

	# of Nodes		# of Edges	
	Train	Test	Train	Test
O2B1	3,683	1,865	8,096	2,315
O2B2	7,132	3,641	15,633	4,469
ORE1	9,222	7,117	61,245	17,501
ORE2	9,200	7,147	61,289	17,512
ORE3	9,186	7,122	61,204	17,490
CaLi10e4	24,556	13,752	127,801	36,519
CaLi10e5	198,896	73,461	265,139	75,757

Table 3
Number of edges by edge type.

	# of Subclass edges		# of Assertion edges		# of all other edges	
	Train	Test	Train	Test	Train	Test
O2B1	105 (1%)	30 (1%)	7,989 (99%)	2,283 (99%)	2 (0%)	2 (0%)
O2B2	105 (1%)	30 (1%)	15,526 (99%)	4,437 (99%)	2 (0%)	2 (0%)
ORE1	8,194 (13%)	2,342 (13%)	53,048 (87%)	15,157 (87%)	3 (0%)	2 (0%)
ORE2	8,204 (13%)	2,344 (13%)	53,081 (87%)	15,166 (87%)	4 (0%)	2 (0%)
ORE3	8,187 (13%)	2,340 (13%)	53,014 (87%)	15,148 (87%)	3 (0%)	2 (0%)
CaLi10e4	59,956 (47%)	17,132 (47%)	51,577 (40%)	14,738 (40%)	16,268 (13%)	4,649 (13%)
CaLi10e5	96,273 (36%)	27,508 (36%)	29,973 (11%)	8,565 (11%)	138,893 (52%)	39,684 (52%)

4.1. Graph Attention Network

Graph Attention Networks (GATs) are neural network architectures that operate on graph-structured data, leveraging masked self-attentional layers to address the shortcomings of prior methods based on graph convolutions [5]. For instance, by stacking layers, in which nodes can attend over their neighborhoods' features, GATs enable specifying different weights to different nodes in a neighborhood without requiring any kind of costly matrix operation or depending on knowing the graph structure upfront. GATs feature an attention mechanism in the aggregation process by learning extra attention weights to the neighbors of each node. All the neighbors $u \in N(v)$ are not equally important to a node v. Moreover, GATs have demonstrated promising results across four established transductive and inductive graph benchmarks: the *Cora*, *Citeseer*, and *Pubmed* citation networks, as well as a *protein-protein interaction* dataset. Formally, the l-th GAT layer can be defined by $h_v^{(l)} = \sigma(\sum_{u \in N(v)} \alpha_{uv} W^{(l)} h_u^{(l-1)})$, where α_{uv} are the attention weights.

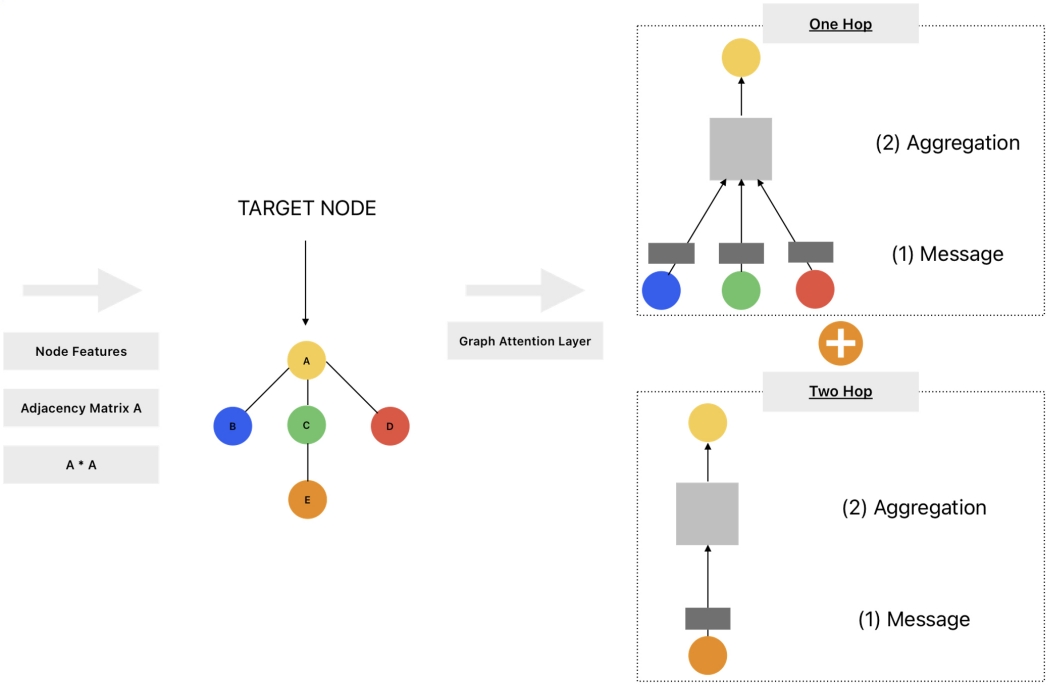
4.2. 2-Hop Graph Attention Network

Standard Graph Neural Networks (GNN) follow a message passing schema to update node representations using information from *one hop* neighborhoods iteratively. In general, GNN layer compresses a set of vectors into a single vector through a two-step process, namely, message and aggregation. In the first step, each node computes a message. For each node $u \in \{N(v) \cup v\}$, the message function is defined as follows: $m_u^{(l)} = MSG^{(l)}(h_u^{(l-1)})$. In the second step, each node aggregates the messages from its neighbors: $h_v^{(l)} = AGG^{(l)}(\{m_u^{(l)}, u \in N(v)\}, m_v^{(l)})$. Hence, the l-th GNN layer takes as input node embeddings from the node itself $h_v^{(l-1)}$ and node embeddings from the neighboring nodes $h_{u \in N(v)}^{(l-1)}$ and outputs node embedding $h_v^{(l)}$.

However, to explicitly capture transitivity, a fundamental property for ontology reasoning, it is essential to extend the Graph Attention Network such that it follows a message passing schema incorporating information from nodes that are *two hops* away to learn node representations. Thus, in this paper, we propose an extended version of the Graph Attention Network [5], which we call *2-Hop Graph Attention Network* (2-Hop GAT). Two hop is crucial for an ontology reasoner so as to capture longer-range dependencies within the graph. By considering information beyond immediate neighbors, the model gains a broader context of the graph structure and can better

understand how distant nodes influence each other.

Figure 1: Two-hop graph attention layer.



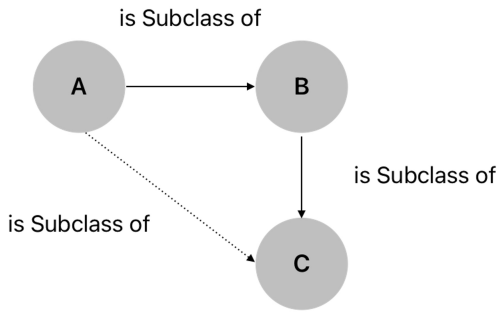
The graph attention layer is expanded to two hops, enhancing attention mechanism to incorporate not only the direct neighbors of a node but also their neighbors up to two hops away. Figure 1 illustrates how the node embedding of a node (i.e., yellow node A) is calculated by aggregating all the messages from its direct neighbors and also their neighbors that are two hops away. Concretely, the inputs of the two-hop graph attention layer are randomly initialized node features/embeddings and adjacency matrices representing the graph's one-hop and two-hop connectivity structures. Then, for each node, the node embeddings with respect to its direct neighbors (nodes up to one hop away) and the node embeddings with respect to its neighbors' neighbors (nodes up to two hops away) are computed by the message passing schema. At the end, the node representations obtained from its direct neighbors are aggregated with the node representations acquired from its neighbors' neighbors.

4.3. Filtered 2-Hop Graph Attention Network

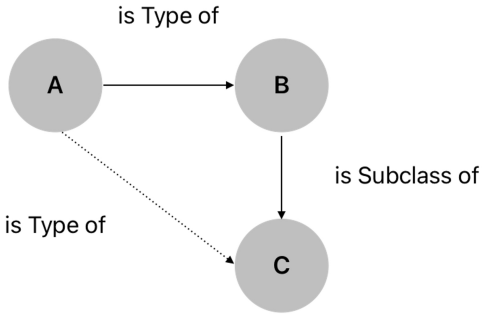
We modified *2-Hop GAT* to guide the message passing schema within the 2-hop graph attention layer to include two RDF rules [8]. By introducing the filtered version of *2-Hop GAT*, we direct the attention mechanism to specific two-hop paths. This mechanism helps the model focus on relevant information and reduce noise or irrelevant signals from the graph. The two RDF rules that were put in the graph attention layer are as follows.

1. Subclass transitivity: A rdfs:subClassOf B & B rdfs:subClassOf C $\implies$ A rdfs:subClassOf C (see Figure 2a)
2. A rdfs:type B & B rdfs:subClassOf C $\implies$ A rdfs:type C (see Figure 2b)

Figure 2: Rules that are used to guide the message passing schema in a graph attention layer.



(a) If A is a subclass of B and B is a subclass of C, then A is also a subclass of C (i.e., RDF rule 11 [8]).



(b) If A is a type of B and B is a subclass of C, then A is also a type of C (i.e., RDF rule 9 [8]).

In our implementation, we focused on subclass transitivity by taking one-hop and two-hop subclass relations (corresponding to the path: A rdfs:subClassOf B & B rdfs:subClassOf C) as input to the two-hop graph attention layer. Additionally, we considered one-hop assertion and two-hop subclass relations for the other rule (representing the path: A rdfs:type B & B rdfs:subClassOf C). Particularly, we trained the *2-Hop GAT* for each rule separately and then combined the models by concatenating them at the final stage.

Mathematically, given a graph $G = (V, E)$, where V is the set of nodes and E is the set of edges, and using the notation $h_v^{(l)}$ to represent the node features of node v at layer l, the filtered 2-Hop graph attention layer concatenates the node embeddings obtained from:

- The standard graph attention layer: $h_v^{(l)} = \sigma(\sum_{u \in N(v)} \alpha_{uv} W^{(l)} h_u^{(l-1)})$.
- The 2-hop graph attention layer taking as input one-hop and two-hop subclass relations: $h_v^{(l)} = \sigma(\sum_{u \in N^{sub}(v)} \alpha_{uv} W^{(l)} h_u^{(l-1)}) + \sigma(\sum_{u \in N_2^{sub}(v)} \alpha_{uv} W^{(l)} h_u^{(l-1)})$, where $N^{sub}(v)$ is the set of neighboring nodes connected to v by edges of type *subclass* and $N_2^{sub}(v)$ includes the nodes that are two hops away from v through second-order edges of type *subclass*.
- The 2-hop graph attention layer taking as input one-hop assertion and two-hop subclass relations: $h_v^{(l)} = \sigma(\sum_{u \in N^{ass}(v)} \alpha_{uv} W^{(l)} h_u^{(l-1)}) + \sigma(\sum_{u \in N_2^{sub}(v)} \alpha_{uv} W^{(l)} h_u^{(l-1)})$, where $N^{ass}(v)$ is the set of neighboring nodes connected to v by edges of type *assertion*.

5. Results

In the context of a link prediction task, computing the Hits@k metric is a common way to evaluate the performance of a link prediction model. Hits@k measures the percentage of positive

examples that appear in the top-k ranked predictions. Specifically, for each test instance, we predicted the tail entity t given the head entity h and the relation r and assessed whether the rank of the predicted tail entity is below k.

We calculated Hits@5 and Hits@10 by employing the Graph Attention Network (GAT) [5] and our two proposed models *2-Hop GAT* and *Filtered 2-Hop GAT* on the datasets provided by the SemRec Challenge at ISWC 2023, namely OWL2Bench [14], ORE [15], and CaLiGraph [16]. We trained GAT, 2-Hop GAT, and Filtered 2-Hop GAT on all the training edges and reported Hits@k by splitting the test edges into Class Subsumption edges, Class Membership edges, and all the edges.

Table 4
Results for Class Subsumption edges, Class Membership edges and all the edges using Hits@5 and Hits@10.

		Subclass Relations		Assertion Relations		All Relations	
		Hits@5	Hits@10	Hits@5	Hits@10	Hits@5	Hits@10
O2B1	GAT	0.067	0.267	**0.035**	**0.153**	0.024	0.168
	2-Hop GAT	**0.100**	**0.667**	0.018	0.081	0.130	0.389
	Filtered 2-Hop GAT	0.000	0.100	0.018	0.089	**0.161**	**0.448**
O2B2	GAT	0.000	0.200	**0.061**	**0.192**	**0.017**	**0.165**
	2-Hop GAT	0.167	0.400	0.036	0.184	0.010	0.104
	Filtered 2-Hop GAT	**0.267**	**0.467**	0.019	0.053	0.008	0.029
ORE1	GAT	0.020	0.121	0.021	0.127	0.011	0.092
	2-Hop GAT	0.029	0.181	0.040	0.245	0.009	0.076
	Filtered 2-Hop GAT	**0.090**	**0.263**	**0.073**	**0.309**	**0.030**	**0.137**
ORE2	GAT	0.024	0.124	0.016	0.155	0.011	**0.108**
	2-Hop GAT	0.035	0.176	0.034	**0.221**	0.009	0.043
	Filtered 2-Hop GAT	**0.082**	**0.208**	**0.040**	0.177	**0.017**	0.074
ORE3	GAT	0.027	0.132	0.013	0.109	0.009	0.063
	2-Hop GAT	**0.064**	**0.237**	**0.039**	0.194	**0.028**	**0.137**
	Filtered 2-Hop GAT	0.036	0.175	0.033	**0.206**	0.007	0.039
CaLi10e4	GAT	**0.436**	**0.709**	**0.372**	**0.659**	**0.386**	**0.692**
	2-Hop GAT	0.399	0.624	0.350	0.533	0.351	0.648
	Filtered 2-Hop GAT	0.120	0.500	0.106	0.441	0.148	0.566
CaLi10e5	GAT	0.001	0.006	0.002	0.018	0.001	0.009
	2-Hop GAT	0.532	0.535	0.150	0.156	0.242	0.245
	Filtered 2-Hop GAT	**0.752**	**0.803**	**0.230**	**0.424**	**0.428**	**0.562**

Table 4 reveals that no specific model performs neither very well nor very poorly for OWL2Bench. On the other hand, for ORE, we can observe that *2-Hop GAT* and *Filtered 2-Hop GAT* outperform *GAT* for the class subsumption and class membership reasoning tasks.

For CaLiGraph10e4, the best results were achieved using *GAT*, while for CaLiGraph10e5, the highest results were obtained employing *Filtered 2-Hop GAT*. Precisely, for CaLiGraph10e5 with the *Filtered 2-Hop GAT* approach, we achieved a Hits@5 score of 0.752 and a Hits@10 score of 0.803 for the class subsumption task, and a Hits@5 score of 0.230 and a Hits@10 score of 0.424 for the class membership task, outperforming the results of Mehryar and Celebi [13].

Overall, the results suggest that the performance of different models varies depending on

the dataset and the type of relations considered, but it can be observed that *Filtered 2-Hop GAT* generally performs well across several datasets and relation types.

6. Discussion

The performance of various models was found to vary with the dataset owing to the distribution of edge types. *GAT* outperforms the other two models for the class membership reasoning task with OWL2Bench, which has a high proportion of assertion relations (99%) and a low proportion of subclass relations (1%). The two-hop model may not be appropriate as few instances follow the rule: if A is a type of B and B is a subclass of C, then A is also a type of C. In contrast, *2-Hop GAT* and *Filtered 2-Hop GAT* performed better than *GAT* for the subsumption reasoning task, suggesting that they capture subclass transitivity.

For CaLiGraph10e4, the best results were achieved using *GAT*, and for CaLiGraph10e5, the highest results were obtained by employing *Filtered 2-Hop GAT*. CaLiGraph10e4 consists of 47% subclass edges, 40% assertion edges, and 13% simple facts, whilst CaLiGraph10e5 contains 36% subclass relations, 11% assertion relations, and 52% simple facts.

The results for ORE are more consistent. *2-Hop GAT* and *Filtered 2-Hop GAT* outperformed *GAT* for the class subsumption and class membership reasoning tasks. This can be explained by the fact that the (relative) number of subclass and assertion edges is similar across the different ORE datasets.

Furthermore, it is important to mention that we solely relied on the provided triplets to train and test the models, without incorporating any axioms from the T-boxes. Consequently, significant and relevant information necessary for ontology reasoning is missing.

7. Conclusion

We developed an extension of the Graph Attention Network, allowing the receptive field to reach nodes that are two hops away. We introduced a filtering mechanism to guide meta paths of two hop to include two RDF rules. We demonstrated that our approaches *2-Hop GAT* and *Filtered 2-Hop GAT* performed well across several datasets and relation types. For CaliGraph10e5, *Filtered 2-Hop GAT* achieved a Hits@5 score of 0.752 and a Hits@10 score of 0.803 for the class subsumption task, and a Hits@5 score of 0.230 and a Hits@10 score of 0.424 for the class membership task. Our results suggest that *2-Hop GAT* and *Filtered 2-Hop GAT* enable transitive reasoning over two types of ontology relations, namely subclass and type reasoning.

References

[1] X. Chen, S. Jia, Y. Xiang, A review: Knowledge reasoning over knowledge graph, Expert Systems with Applications 141 (2020) 112948. URL: https://www.sciencedirect.com/science/article/pii/S0957417419306669. doi:https://doi.org/10.1016/j.eswa.2019.112948.

[2] A. Bordes, N. Usunier, A. Garcia-Durán, J. Weston, O. Yakhnenko, Translating embeddings for modeling multi-relational data, in: Proceedings of the 26th International Conference

on Neural Information Processing Systems - Volume 2, NIPS'13, Curran Associates Inc., Red Hook, NY, USA, 2013, p. 2787–2795.
[3] T. Trouillon, J. Welbl, S. Riedel, Éric Gaussier, G. Bouchard, Complex embeddings for simple link prediction, 2016. arXiv:1606.06357.
[4] T. N. Kipf, M. Welling, Semi-supervised classification with graph convolutional networks, CoRR abs/1609.02907 (2016). URL: http://arxiv.org/abs/1609.02907. arXiv:1609.02907.
[5] P. Veličković, G. Cucurull, A. Casanova, A. Romero, P. Liò, Y. Bengio, Graph attention networks, in: International Conference on Learning Representations, 2018. URL: https://openreview.net/forum?id=rJXMpikCZ.
[6] Q. Li, Z. Han, X. Wu, Deeper insights into graph convolutional networks for semi-supervised learning, CoRR abs/1801.07606 (2018). URL: http://arxiv.org/abs/1801.07606. arXiv:1801.07606.
[7] N. Dehmamy, A.-L. Barabasi, R. Yu, Understanding the representation power of graph neural networks in learning graph topology, in: H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, R. Garnett (Eds.), Advances in Neural Information Processing Systems, volume 32, Curran Associates, Inc., 2019. URL: https://proceedings.neurips.cc/paper_files/paper/2019/file/73bf6c41e241e28b89d0fb9e0c82f9ce-Paper.pdf.
[8] J. Urbani, S. Kotoulas, E. Oren, F. Harmelen, Scalable distributed reasoning using mapreduce, volume 5823, 2009, pp. 634–649. doi:10.1007/978-3-642-04930-9_40.
[9] G. Wang, R. Ying, J. Huang, J. Leskovec, Direct multi-hop attention based graph neural network, CoRR abs/2009.14332 (2020). URL: https://arxiv.org/abs/2009.14332. arXiv:2009.14332.
[10] D. Yuxiao, N. Chawla, A. Swami, metapath2vec: Scalable representation learning for heterogeneous networks, 2017, pp. 135–144. doi:10.1145/3097983.3098036.
[11] Y. Sun, J. Han, X. Yan, P. Yu, T. Wu, Pathsim: Meta path-based top-k similarity search in heterogeneous information networks, PVLDB 4 (2011) 992–1003. doi:10.14778/3402707.3402736.
[12] T. Mikolov, K. Chen, G. Corrado, J. Dean, Efficient estimation of word representations in vector space, 2013. arXiv:1301.3781.
[13] S. Mehryar, R. Celebi, Improving transitive embeddings in neural reasoning tasks via knowledge-based policy networks, ????, pp. 16–27. URL: https://ceur-ws.org/Vol-3337/semrec_paper3.pdf.
[14] G. Singh, S. Bhatia, R. Mutharaju, OWL2Bench: A Benchmark for OWL 2 Reasoners, 2020, pp. 81–96. doi:10.1007/978-3-030-62466-8_6.
[15] N. Matentzoglu, B. Parsia, Ore 2014 reasoner competition dataset, 2014. URL: https://doi.org/10.5281/zenodo.10791. doi:10.5281/zenodo.10791.
[16] N. Heist, H. Paulheim, The caligraph ontology as a challenge for OWL reasoners, CoRR abs/2110.05028 (2021). URL: https://arxiv.org/abs/2110.05028. arXiv:2110.05028.