Augmenting Graph Inductive Learning Model With Topographical Features

Kalyani Selvarajah and Jae Muzzin

School of Computer Science, University of Windsor, Windsor, ON, Canada
{kalyanis,muzzin4}@uwindsor.ca

Abstract. Knowledge Graph (KG) completion aims to find the missing entities or relationships in a knowledge graph. Although many approaches have been proposed to construct complete KGs, graph embedding methods have recently gained massive attention. These methods performed well in transductive settings, where the entire collection of entities must be known during training. However, it is still unclear how effectively the embedding methods capture the relational semantics when new entities are added to KGs over time. This paper proposes a method, AGIL, for learning relational semantics in knowledge graphs to address this issue. Given a pair of nodes in a knowledge graph, our proposed method extracts a subgraph that contains common neighbors of the two nodes. The subgraph nodes are then labeled based on their distance from the two input nodes. Some heuristic features are computed and given along with the adjacency matrix of the subgraph as input to a graph neural network. The GNN predicts the likelihood of a relationship between the two nodes. We conducted experiments on five real datasets to demonstrate the effectiveness of the proposed framework. The AGIL in relation prediction outperforms the baselines both in the inductive and transductive setting.

Keywords: Knowledge Graphs · Graph Neural Networks · Subgraph

1 Introduction

In recent years, significant progress has been made in the construction and deployment of knowledge graphs (KGs) [29]. KGs represent structured relational information in the form of subject-predicate-object (SPO) triples, e.g,$\langle Justin\ Trudeau, fatherOf, XavierJames\rangle$. Freebase [4], YAGO [23], DBPedia [1], ConceptNet [22], and Never-ending language learning (NELL) [6] are a few prominent examples of large KGs. Recently, KGs have gained widespread attention because of their benefits in a variety of applications, including question answering [14], dialogue generation [13], information retrieval [30], entity linking [10] and recommendation systems [37].

Despite their usefulness and popularity, KGs are often noisy and incomplete because it is challenging to incorporate all information in the real world, and these data are typically dynamic and evolving, making it difficult to generate accurate and complete KGs [28]. Therefore, automating the construction

ICCS Camera Ready Version 2022
To cite this paper please use the final published version:
DOI: 10.1007/978-3-031-08757-8_60

of a complete KG is a tedious process. Various techniques have been proposed for knowledge graph completion, such as the traditional Statistical Relational Learning (SRL) methods and Knowledge graph embedding methods. Building a complete KG is possible by predicting objects (known as link prediction) and relations.

A relation or logical induction prediction problem discovers probabilistic logical rules from a given KG. Induction can be learned in several ways such as from examples [16] and from interpretations [7]. For example, let's say, "the 23rd prime minister of Canada, Justin Trudeau lives in Ottawa, and is married to Sophie Trudeau." The first-order logic of the above sentence would be $LivesIn(Justin\ Trudeau, Ottawa) \wedge MarriedTo(JustinTrudeau, SophieTrudeau)$. Therefore, a logical rule can be derived based on the concept that a married couple lives together (generally); $LivesIn(X, Y) \wedge MarriedTo(X, Z) \rightarrow LivesIn(Z, Y)$. This rule can be used to find the relation or possible hypothesis $LivesIn(SophieTrudeau, Ottawa)$. Here, the known logical rules have been generalized to derive a new rule or relationship which is true most of the time. Additionally, this rule predicts the relation for the entities which did not exist when KGs were trained. In reality, KGs evolve with time and new entities will join. Most of the existing embedding-based methods are highly successful in predicting the relations if the entities were seen when KGs were trained, which is transductive reasoning. Generalizing relational semantics is a challenging task and important to see the relationships in unseen entities, which is inductive reasoning. However, these embedding methods have some limitations in explicitly capturing the relational semantics when new entities are added to KGs over time.

This paper proposes an Augmenting Graph Inductive Learning (AGIL) framework to learn relational semantics in a given KG, and predict relations $(s, ?, o)$. Since much of the existing machine learning methods suffer from scalability issues, recently, PLACN [17] and GraIL [25] applied subgraph-based methods in link prediction and relation prediction, respectively, to overcome this problem. GraIL used a Graph Neural Network (GNN) based relations prediction method to learn relational semantics even if the entities were unseen during training. However, GraIL operated strictly on subgraphs and utilized no additional information. PLACN, on the other hand, successfully used local features as additional information for link prediction. So, our proposed model exploits both PLACN and GraIL to derive AGIL, which includes three primary steps. First, the subgraph is extracted with common neighbors of the target link between nodes i and j. The common neighbors in the enclosed subgraph are collected till k number of hops. Then the subgraph is labeled using the Double-Radius Node Labeling [35] method. In the final steps, the heuristic features of nodes for the entire subgraph are extracted and fed into GNN along with the adjacency matrix of the subgraph, which aggregates the feature vectors into a scoring function for the prediction.

Our Contribution: The followings are the summary of our contributions:

ICCS Camera Ready Version 2022
To cite this paper please use the final published version:
DOI: 10.1007/978-3-031-08757-8_60

1. We propose an Augmenting Graph Inductive Learning (AGIL) framework based on common neighbors-based subgraphs for relations prediction in both transductive and inductive settings.
2. We extract heuristic features of nodes from the entire subgraph and model a new prediction framework based on Graph Neural Networks (GNN);

The rest of the paper is organized as follows. Section 2 discusses related existing work. Our framework is presented in Section 3. Following that, Section 4 presents the experimental setup and the corresponding results. Finally, Section 5 concludes the research idea of this paper with directions for future work.

2 Related Work

Multiple methods have been proposed to construct a complete knowledge graph. Graph Embedding is one of the most broadly used solutions for Knowledge-Graph Completion challenges. Translation-based approach [5], [24], [5] Bilinear-based approach [33], [27] and Neural-Network-based approach [8], [2] are well-known graph embedding approaches.

Traditional approaches on the KG embedding methods are in a transductive manner. They require all entities during training. However, many real-world KGs are ever-evolving by adding new entities and relationships. Several inductive KG embedding approaches are proposed to address the issue of emergent entities. Graph2Gauss [3] is an approach to generalize to unseen nodes efficiently on large-scale attributed graphs using node features. Then Hamilton et.al. [11] proposed a generic inductive framework, GraphSAGE, that efficiently generates node embeddings for previously unseen data in a graph by leveraging node feature information. Node features are, however, not available in many KGs. In addition to these inductive embedding methods, DRUM [18], NeuralLP [34], and RuleN [15] are few models which learn logical rule and predict relations in KGs.

Recently, GraIL [25] was proposed to generalize inductive relation based on subgraph reasoning. Since GraIL shows comparatively better performance than the state-of-the-art methods, we consider extending it. Additionally, SEAL [35], PLACN [17] and DLP-LES [21] are few recent approaches that successfully extracted subgraphs from a given networks and applied heuristic features to train the model. Motivated by their high performance, we incorporate these heuristic features with our model.

3 Problem Definition and Proposed Approach:

Given a KG, $G = \langle V, E, R \rangle$ is a directed graph, where V is the set of vertices, E is the set of edges and R represents the set of relations. The edges in E connect two vertices to form triplets (h, r, t), where h is a head entity in V, t is a tail entity in E and r is a relation in R, i.e., $E = \{(h, r, t)|h \in V, r \in R, t \in V\}$.

ICCS Camera Ready Version 2022
To cite this paper please use the final published version:
DOI: 10.1007/978-3-031-08757-8_60

In a given KG, there is a high chance of missing relations $(h, ?, t)$, head entity $(?, r, t)$ and tail entity $(h, r, ?)$. Knowledge graph completion in a given KG, G is defined as the task of predicting missing triplets, $E' = \{(h, r, t)|h \in V, r \in R, t \in V, (h, r, t) \notin E\}$ in both transductive and inductive settings. In the transductive setting, the entities in a test triple are considered to be in the set of training entities. Predicting missing triplets in the transductive setting is defined as $E'' = \{(h, r, t)|h \in V, r \in R, t \in V, (h, r, t) \notin E\}$. In the inductive setting, the entities in a test triple are never seen in the set of training entities. Predicting missing triplets in the inductive setting is defined as $E''' = \{(h, r, t)|h \in V' \text{ or } t \in V', r \in R, (h, r, t) \notin E\}$, where $V' \cap V = \emptyset$ and $V' \neq \emptyset$.

Our primary objective is to predict the relation between two nodes. We employ Graph Neural Network (GNN) [19] to learn the knowledge graph's structural semantics. The proposed model has the following steps;

1. Subgraph extraction.
2. Node labeling.
3. Feature matrix construction.
4. Scoring the subgraph using GNN.

3.1 Subgraph Extraction

For each triple in the knowledge graph, the subgraph is extracted with the goal of isolating the connecting nodes between the two target nodes u and v. We wish to isolate only the nodes which are found along every possible path between the head and tail of the knowledge triple, referred to as the target nodes of the subgraph. A few approaches in the existing literature [35, 17] have been proposed for subgraph extraction from a given graph. AGIL extracts subgraphs using common neighbors of any targeted nodes u and v because sufficient information of entire nodes of subgraphs can be taken for the training process [17]. Moreover, having additional information about the shared neighbours of both nodes u and v allows to determine the future existence of a relationship between them. We set a number k for the number of hops to collect the nodes in the subgraph, which can be defined as given below.

Definition 1. ***Subgraph:*** *For a given knowledge graph $G = \langle V, E, R \rangle$, let $\Gamma_k(x)$ be the neighbors of x within k hops. The subgraph of a target link between nodes u and v is given by the function $S : V^2 \rightarrow 2^V$, the function that returns the set of common neighbor nodes connecting u and v,*

$$S = \bigcup_{i=1}^{k} (\Gamma_i(u) \cap \Gamma_i(v)); \ for\ some\ m > 1 \tag{1}$$

where $\{u, v\} \in V'$, $V' \subseteq V$ and V' is a set of common neighbors for the targeted nodes, and $|V'| = \emptyset$.

ICCS Camera Ready Version 2022
To cite this paper please use the final published version:
DOI: 10.1007/978-3-031-08757-8_60

3.2 Subgraph Node Labeling

Generally, GNN takes both feature matrix X and adjacency matrix A as input, (A, X). To construct a feature matrix X of a subgraph, the position of nodes are really important to maintain the consistency of the structural information. GNN learns the existence of target links for prediction. So, we exploit the Double-Radius Node Labeling method, which was proposed by SEAL [35] to label the subgraphs.

Each label is a 2-tuple. The first element is the distance from the first target node, the second element is the distance from the second. The target node labels are always $(0, 1)$ and $(1, 0)$. Figure 1 is an example of a subgraph for target nodes $\langle University, ?, ComputerScience \rangle$, and the labeled subgraph.

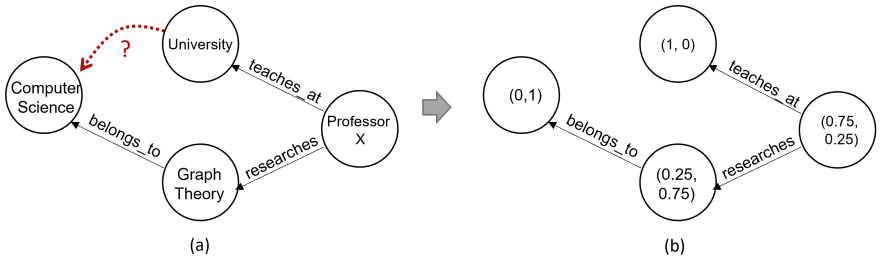

Fig. 1. (a) Subgraph of target nodes. (b)Labeled Subgraph.

3.3 Feature Matrix Construction

The GraIL [25] graph neural network architecture considers only the structural node feature X for predicting triplets of a given KG. However, we believe that in addition to the structural node feature, incorporating explicit features of the subgraph to the feature matrix X send additional information to the graph neural network training model. Since a knowledge graph does not always have any explicit feature information about a node, we decided to use the topological feature of the subgraph to see the importance of topological features in relation prediction. We belive that topological heuristics are useful in knowledge graph completion because entities are less likely to form relationships with entities that are farther away. Similarly with social networks, people tend to create new relationships with people that are closer to them. The motivation to apply topological heuristics to knowledge graphs was inspired by research in social networks. Our model uses the proximity measures taken from the topology as a heuristic in link prediction.

The specific heuristics used in this research were chosen to give a precise measurement of the notion of proximity of entities within the knowledge graph. In research done by Liben-Nowell and Kleinberg [12] with neighbour-based proximity measures, it was found that predictions outperformed chance by factors of

ICCS Camera Ready Version 2022
To cite this paper please use the final published version:
DOI: 10.1007/978-3-031-08757-8_60

40 to 50, which led the authors to concluded that topology does indeed contain latent information which can be used to predict missing or future relationships.

We chose to use multiple proximity measurements as each has their own characteristics. AGIL uses the following five simple heuristics as used in PLACN model. Here $\Gamma(v)$ and $\Gamma(u)$ specify the set of neighbors within k hops for nodes v and u respectively.

Common Neighbors (CN) counts how many neighbours any two vertices v and u have in common.

$$CN_{u,v} = |\Gamma_k(v) \cap \Gamma_k(u)| \tag{2}$$

Jaccard Coefficient (JC) produces the normalized form of CN.

$$JC_{u,v} = \frac{|\Gamma_k(v) \cap \Gamma_k(u)|}{|\Gamma_k(v) \cup \Gamma_k(u)|} \tag{3}$$

Adamic-Adar (AA) is a modified version of JC, which gives a higher priority to the common neighbors with lower degree.

$$AA_{u,v} = \sum_{w \in |\Gamma_k(v) \cup \Gamma_k(u)|} \frac{1}{log|\Gamma_k(w)|} \tag{4}$$

Preferential Attachment (PA) The idea behind PA is that a node with a higher degree has a better probability of forming new connections.

$$PA_{u,v} = |\Gamma_k(v).\Gamma_k(u)| \tag{5}$$

Resource Allocation (RA) is much more similar to AA, but gives higher priority to low-degree common neighbors.

$$RA_{u,v} = \sum_{w \in |\Gamma_k(i) \cup \Gamma_k(j)|} \frac{1}{|\Gamma_k(w)|} \tag{6}$$

Let $f : V^2 \rightarrow R^5$ be the function which returns the set of above five heuristic features for the pair of nodes u, v. So, $f(u, v)$ returns a vector of five components, each holding the CN, JC, AA, PA, and RA Value. Let S be the set of nodes in the enclosing subgraph of target nodes u and v, and $\{u, v\} \in S$ and $S \subseteq V$. Then for every node $i \in S$, we can evaluate the heuristic features of x and each of x's neighbors $y \in \Gamma(x)$ using $f(x, y)$.

Here, we discuss how we calculate the feature vector of node x. Let P_x be the matrix whose columns are label of nodes in S and rows are five feature vectors, i.e, $P_x = [R_0, R_1, \ldots, R_n]$, where $R_y = f(x, y)\ \forall y \in S$ and $n = |S|$. The matrix P_x contains all five heuristic features of every possible pair of nodes in the subgraph. The number of rows and columns of P_x are 5 and $|S|$ respectively.

ICCS Camera Ready Version 2022
To cite this paper please use the final published version:
DOI: 10.1007/978-3-031-08757-8_60

Example 1 *Consider a subgraph* $S = \{u, v, w, x, y, z\}$*, and* u *and* v *are the target nodes of the subgraph* S*, then*

$$P_x = \begin{bmatrix} CN_u & CN_v & CN_w & 0 & CN_y & CN_z \\ AA_u & AA_v & AA_w & 0 & AA_y & AA_z \\ JC_u & JC_v & JC_w & 0 & JC_y & JC_z \\ RA_u & RA_v & RA_w & 0 & RA_y & RA_z \\ PA_u & PA_v & PA_w & 0 & PA_y & PA_z \end{bmatrix}_{5\times n}$$

Where the column for x *is zero, because it is pointless to compare* x*'s topology to itself.*

Since the size of P_x depends on the size of the subgraph, $|S|$, which is variable, this is not suitable for scaling in training on when the node degree of the graph is very high. A very large feature vector can cause critical performance issues in the model. This is where the Fixed Sized Subgraph and Variable Sized Subgraph models diverge. Each take a different approach in deriving a feature vector F_x from the topology matrix P_x.

PLACN uses a constant value k, which is the absolute maximum size a subgraph may reach. k is derived in a way to be large enough for most node pairs. The value of k is a function of the number of edges and nodes in the complete graph.

$$k \approx \left\lceil \frac{2|E|}{|V|} \left(1 + \frac{2|E|}{|V|(|V|-1)}\right) \right\rceil \tag{7}$$

The theoretical analysis of GraIL determined that any logical rule R derived from the topology of a knowledge graph uniquely corresponds to a set of nodes connected through a sequence of relations, and that GraIL can learn this rule if the nodes and relations are present in the graph neural network.

To examine the differences between fixed and variable size sub graph, we constructed our feature matrix and sent it to GNN.

Topology Information in Variable Sized Subgraphs: In order to have a feature vector of constant size, we take a statistical analysis of each heuristic feature, across all of the nodes in the subgraph. For each heuristic feature $R \in R^5$, we can take the mean, median, standard deviation, minimum, maximum and variance across all of the nodes in the subgraph. In other words, we can apply the statistical functions to the rows of the topology matrix P_x.

Let $F_x = Stat(P_x)$, where $Stat(P_x)$ replaces each row of P_x with $\langle Mean(r), Median(r), Variance(r), Min(r), Max(r), Std(r)\rangle$, So the resulting feature matrix has 30 elements,

$$F_x = \begin{bmatrix} Mean(CN) & Med(CN) & Var(CN) & Min(CN) & Max(CN) & STD(CN) \\ Mean(AA) & Med(AA) & Var(AA) & Min(AA) & Max(AA) & STD(AA) \\ Mean(JC) & Med(JC) & Var(JC) & Min(JC) & Max(JC) & STD(JC) \\ Mean(RA) & Med(RA) & Var(RA) & Min(RA) & Max(RA) & STD(RA) \\ Mean(PA) & Med(PA) & Var(PA) & Min(PA) & Max(PA) & STD(PA) \end{bmatrix}_{5\times 6}$$

ICCS Camera Ready Version 2022
To cite this paper please use the final published version:
DOI: 10.1007/978-3-031-08757-8_60

Therefore, F_x sees the rows of P_x replaced by the statistical results which are rows of fixed size 5, since we consider five heuristic values and columns of fixed size 6 since there are 6 statistical functions. F_x will always have a total of 30 elements, suitable to be encoded into a node x's feature vector for training in the Graph Neural Network. We simply list all 30 elements as components of the final feature vector.

Topology Information in Fixed Sized Subgraphs: As PLACN used in it's architecture, the topological feature matrix P_x of subgraphs is fixed for a given KG. The columns correspond to the fixed subgraph size and the rows correspond to each heuristic function. Therefore the size of P_x is always $5 \times |S|$. Thus, for fixed sized subgraphs, we can directly encode P_x,

$$F_x = P_x \tag{8}$$

In practice, this has led to very large vectors, when the fixed size of the subgraphs is large.

3.4 Scoring subgraph using GNN

This section explains the importance of GNN in our framework.

GNN Message Passing: In a GNN, a hidden embedding h_u^k for each node $u \in V$ is updated on each message-passing iteration based on information gathered from u's graph neighbor $\Gamma(u)$. In other terms, the representation of the node u is iteratively updated by aggregating its neighbors' representations [32]. So basically, GNN works based on two functions: Aggregation function passes information from $\Gamma(u)$ to u, and update function update features of u based on the information to form an embedded representation.

In AGIL model, each enclosed subgraph has a network of k-hop neighborhood nodes. So, after aggregating for k iteration, the kth layer of GNN is represented as,

$$m_u^k = \text{AGGREGATE}^k(\{h_v^{k-1} : v \in \Gamma(u)\} \tag{9}$$

$$h_u^k = \text{UPDATE}^k(h_u^{k-1}, m_u^k) \tag{10}$$

where h_u^k is the feature vector of node u at kth iteration, Initially, $h_u^0 = X_u$, and m_u^k is the message aggregated from $\Gamma(u)$ at kth iteration.

In equation 9, there are various approaches proposed for message AGGREGATE function. Motivated by these architectures, GraIL adopts the method proposed by [20]. The following function defines the message aggregated function in a relational multi-graph:

$$h_u^k = \sigma\left(\sum_{r \in R} \sum_{v \in \Gamma_i^r} \alpha_{rr_uvu} W_r^{k-1} h_v^{k-1} + W_0^{k-1} h_u^{k-1}\right) \tag{11}$$

where k is the current layer of the neural network, u is the node being aggregated, R is the set of relationship types, α_{rr_uvu} is the attention value for layer k, r is

ICCS Camera Ready Version 2022
To cite this paper please use the final published version:
DOI: 10.1007/978-3-031-08757-8_60

		WN18RR			FB15k-237			NELL-995		
		# R	# V	# E	# R	# V	# E	# R	# V	# E
v1	train	9	2746	6678	183	2000	5226	14	10915	5540
	ind-test	9	992	1991	146	1500	2404	14	225	1034
v2	train	10	6954	18968	203	3000	12085	88	2564	10109
	ind-test	10	2923	4863	176	2000	5092	79	4937	5521
v3	train	11	12078	32150	218	4000	22394	142	4647	20117
	ind-test	11	5084	7470	187	3000	9137	122	4921	9668
v4	train	9	3861	9842	222	5000	33916	77	2092	9289
	ind-test	9	7208	15157	204	3500	14554	61	3294	8520

Table 1. The statistical information of datasets for inductive setting, where R, V and E are relations, vertices and edges respectively.

any relationship , r_t is a target relationship between nodes v and u, W_r^k is the transformation matrix for r and layer k, and h_v^k is the feature vector of the node v.

The GNN uses an aggregation function to distribute features of nodes into their neighbors, for each layer of the neural network. The aggregation function used by GraIL uses the node's labels as the feature vector. We append the elements of the feature matrix F_x to the h vector used in formulation 11. The feature vector h in the GraIL model uses only the node labels $(L1, L2)$, for example $(1, 0)$, or (25.75). However, the node structured information (i.e, node labels) are limited information for training GNN. Therefore, we extend the feature vector with the 30 elements from F_x. So, in AGIL model, the feature vector h for node x would incorporate the statistical analysis of heuristic features as below;
$\langle L1, L2, Mean(CN), Med(CN), Var(CN), Min(CN), Max(CN), STD(CN), Mean(AA), Med(AA), Var(A\,A), Min(AA), Max(AA), STD(AA), Mean(JC), Med(JC), Var(JC), Min(JC), Max(JC), STD(JC), Mean(RA), Med(RA), Var(RA), Min(RA), Max(RA), STD(RA), Mean(PA), Med(PA), Var(PA), Min(PA), Max(PA), STD(PA)\rangle$.

At each layer, the graph neural network continuously combines feature vectors of nodes with the aggregates of their 1-hop neighborhoods.

4 Experiments

We perform experiments to demonstrate the efficiency and effectiveness of our framework, AGIL. Experiments are carried out on benchmark datasets, WN18RR [9], FB15k-237 [26], and NELL-995 [31] which were originally developed for transductive settings. To conduct inductive relation prediction, we use 4 versions of inductive datasets and 2 versions of transductive datasets, which are prepared by the GraIL authors and identical to the data used in their experiments. They constructed fully-inductive benchmark datasets by sampling disjoint subgraphs from the KGs. These datasets consist of two set of graphs: Train-graph and Ind-test-graph. Table 1 represents the statistical information on how benchmark datasets are split for inductive setting.

ICCS Camera Ready Version 2022
To cite this paper please use the final published version:
DOI: 10.1007/978-3-031-08757-8_60

All the experiments are performed on a Intel(R) Core(TM) i7-3770 CPU computer @3.40GHZ speed and 24 GB of RAM.

	WN18RR				**FB15k-237**				**NELL-995**			
	v1	v2	v3	v4	v1	v2	v3	v4	v1	v2	v3	v4
Neural-LP	86.02	83.78	62.90	82.06	69.64	76.55	73.95	75.74	64.66	83.61	87.58	85.69
DRUM	86.02	84.05	63.20	82.06	69.71	76.44	74.03	76.20	59.86	83.99	87.71	85.94
RuleN	90.26	89.01	76.46	85.75	75.24	88.70	91.24	91.79	84.99	88.40	87.20	80.52
GraIL	94.32	94.18	85.80	92.72	84.69	90.57	91.68	**94.46**	86.05	92.62	93.34	87.50
AGIL (F-Subgraph)	**96.38**	**95.77**	**89.28**	**95.66**	73.5	84.56	76.4	NA	90.56	93.7	94.18	NA
AGIL (V-Subgraph)	94.76	94.92	86.46	93.65	**87.42**	**91.20**	**93.44**	93.52	**91.21**	**96.84**	**97.04**	**95.42**

Table 2. Inductive Setting Experimental Result (AUC-PR)

4.1 Inductive Relation Prediction

We test our model, AGIL on inductive datasets to determine if it can generalise relations when the entities aren't visible during GNN training. AGIL is trained on Train-graph and tested on Ind-test-graph.

To evaluate the performance, we compare AGIL against the following state-of-the-art methods.

1. NeuralLP [34]: an end-to-end differentiable model for inductive relation prediction.
2. DRUM [18]: a scalable and differentiable approach for mining first-order logical rules from KG.
3. RuleN [15]: statistical rule mining method, and the current state-of-the-art in inductive relation prediction on KGs.
4. GraIL [25]: inductive relation prediction by subgraph reasoning, and highly similar to AGIL.

We use the original source code by the authors for the implementation of above methods, NeuralLP[1], DRUM[2], RuleN[3] and GraIL[4]. For AGIL framework, the implementation is built upon the Python code base provided by [25] in their GraIL implementation. It uses the Deep Graph Learning library to implement a graph neural network.

Results and Discussion: The performance of the experimental setup for AGIL is represented in Table 2 against baseline methods. The Precision Recall Area Under Curve (AUC-PR) is used to evaluate the model's accuracy. The AGIL model is tested based on fixed sized subgraph (F-Subgraph) as proposed in PLACN, and variable sized subgraph (V-Subgraph). We observed that the model with fixed sized subgraph fails to perform better in some dataset such as v4-FB15k-237 and v4-NELL-995. The poor performance might be due to the absence of critical nodes and relations in the subgraph with fixed size neighbours. But, AGIL model with variable sized subgraph outperforms most of the

[1] https://github.com/fanyangxyz/Neural-LP
[2] https://github.com/alisadeghian/DRUM
[3] https://web.informatik.uni-mannheim.de/RuleN/
[4] https://github.com/kkteru/grail

ICCS Camera Ready Version 2022
To cite this paper please use the final published version:
DOI: 10.1007/978-3-031-08757-8_60

standard baseline methods. In the NELL-995 dataset, the improvement is most significant compared to the other datasets. In WN18RR dataset, AGIL performs significantly better when we use fixed sized subgraph extraction. This indicates that for any knowledge graph of realistic size, fixed sized subgraphs are not always suitable for Inductive Graph Neural Network models.

If k value is sufficiently large enough, it may include all connecting paths. The recommended calculation to derive k by PLACN was shown to be too low for certain data sets, such as FB15k-237. If a knowledge graph has a high number of cycles, there may be many alternative paths between target nodes. Due to the truncation of the subgraph size to k, only a subset of possible paths will be analyzed by the neural network. Therefore, only a subset of the possible inductive rules will be learned by the GNN. When those inductive rules are applied to link prediction, they fail to produce accurate results. Both AGIL and GraIL provide a limiting factor to prevent excessively large subgraphs. It limits the number of hops from each node to a maximum, in all experiments, this maximum was 3 hops.

Moreover, GraIL outperforms on v4 of FB15k-237. However, the performance of AGIL is still close to GraIL on this dataset.

4.2 Transductive Relation Prediction

Most existing embedding based KG completion methods consider transductive setting for the prediction. Basically, all the existing KGs including WN18RR, FB15k-237, and NELL-995 are originally developed for the transductive setting. We test AGIL on transductive setting to determine it can predict the links accurately. We then compare AGIL against GraIL and RuleN.

	WN18RR		**FB15k-237**		**NELL-995**	
	v1	v2	v1	v2	v1	v2
RuleN	81.79	83.97	87.07	92.49	80.16	87.87
GraIL	89.00	90.66	88.97	93.78	83.95	92.73
AGIL	**92.77**	**92.80**	**90.03**	**95.56**	**92.44**	**93.84**

Table 3. Transductive Setting Experimental Result (AUC-PR)

Results and Discussion: The experimental results on transductive setting is represented in Table 3, which compares AGIL with GraIL and state-of-the-art method RuleN. In all the cases, AGIL outperforms the other two methods. For the time being, we could not compare AGIL with other embedded-based methods. We will compare this in the future.

During the experiments it was shown that use of feature vectors would cause the model to overfit the training data, and loose some generality when applied to the test triples. To resolve this, we utilized the NodeNorm function [36] to normalize the feature vector. This gives the effect of making each feature vector have the save variance. Zhou et.al [36] have observed that GNNs perform poorly when the variance of features of nodes is very high. The normalization replaces

ICCS Camera Ready Version 2022
To cite this paper please use the final published version:
DOI: 10.1007/978-3-031-08757-8_60

each component in the feature vector with the difference from the mean divided by the variance.
The code is available in the GitHub link:
https://anonymous.4open.science/r/agil2021/README.md.

5 Conclusions

This paper examines an augmenting graph inductive learning framework based on GNN, named AGIL. Since many real-world KGs evolve with time, training very large networks with GNN is a challenging task. Therefore, we used a common neighbor-based subgraph to solve the scalability issue. Although AGIL is highly similar to the recently proposed model GraIL, AGIL incorporates topological heuristic features as additional information when GNN trains. Experimentally, we can see that the additional feature information gives better accuracy in both transductive and inductive settings. We also proved experimentally that fixed-sized subgraphs are not always suitable for Inductive Graph Neural Network models. Overall, our model, AGIL, outperforms most of the baseline methods. In the future, we are planning to examine the importance of individual topological features for the relation prediction.

References

1. Auer, S., Bizer, C., Kobilarov, G., Lehmann, J., Cyganiak, R., Ives, Z.: Dbpedia: A nucleus for a web of open data. In: The semantic web, pp. 722–735. Springer (2007)
2. Balažević, I., Allen, C., Hospedales, T.M.: Hypernetwork knowledge graph embeddings. In: International Conference on Artificial Neural Networks. pp. 553–565. Springer (2019)
3. Bojchevski, A., Günnemann, S.: Deep gaussian embedding of graphs: Unsupervised inductive learning via ranking. arXiv preprint arXiv:1707.03815 (2017)
4. Bollacker, K., Evans, C., Paritosh, P., Sturge, T., Taylor, J.: Freebase: a collaboratively created graph database for structuring human knowledge. In: Proceedings of the 2008 ACM SIGMOD international conference on Management of data. pp. 1247–1250 (2008)
5. Bordes, A., Usunier, N., Garcia-Duran, A., Weston, J., Yakhnenko, O.: Translating embeddings for modeling multi-relational data. Advances in neural information processing systems **26** (2013)
6. Carlson, A., Betteridge, J., Kisiel, B., Settles, B., Hruschka, E.R., Mitchell, T.M.: Toward an architecture for never-ending language learning. In: Twenty-Fourth AAAI conference on artificial intelligence (2010)
7. De Raedt, L., Džeroski, S.: First-order jk-clausal theories are pac-learnable. Artificial Intelligence **70**(1-2), 375–392 (1994)
8. Demir, C., Ngomo, A.C.N.: Convolutional complex knowledge graph embeddings. In: European Semantic Web Conference. pp. 409–424. Springer (2021)
9. Dettmers, T., Minervini, P., Stenetorp, P., Riedel, S.: Convolutional 2d knowledge graph embeddings. In: Thirty-second AAAI conference on artificial intelligence (2018)

ICCS Camera Ready Version 2022
To cite this paper please use the final published version:
DOI: 10.1007/978-3-031-08757-8_60

10. Hachey, B., Radford, W., Nothman, J., Honnibal, M., Curran, J.R.: Evaluating entity linking with wikipedia. Artificial intelligence **194**, 130–150 (2013)
11. Hamilton, W.L., Ying, R., Leskovec, J.: Inductive representation learning on large graphs. In: Proceedings of the 31st International Conference on Neural Information Processing Systems. pp. 1025–1035 (2017)
12. Liben-Nowell, D., Kleinberg, J.: The link-prediction problem for social networks. Journal of the American society for information science and technology **58**(7), 1019–1031 (2007)
13. Liu, S., Chen, H., Ren, Z., Feng, Y., Liu, Q., Yin, D.: Knowledge diffusion for neural dialogue generation. In: Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers). pp. 1489–1498 (2018)
14. Lukovnikov, D., Fischer, A., Lehmann, J., Auer, S.: Neural network-based question answering over knowledge graphs on word and character level. In: Proceedings of the 26th international conference on World Wide Web. pp. 1211–1220 (2017)
15. Meilicke, C., Fink, M., Wang, Y., Ruffinelli, D., Gemulla, R., Stuckenschmidt, H.: Fine-grained evaluation of rule-and embedding-based systems for knowledge graph completion. In: International semantic web conference. pp. 3–20. Springer (2018)
16. Muggleton, S.: Inductive logic programming. New generation computing **8**(4), 295–318 (1991)
17. Ragunathan, K., Selvarajah, K., Kobti, Z.: Link prediction by analyzing common neighbors based subgraphs using convolutional neural network. In: ECAI 2020, pp. 1906–1913. IOS Press (2020)
18. Sadeghian, A., Armandpour, M., Ding, P., Wang, D.Z.: Drum: End-to-end differentiable rule mining on knowledge graphs. arXiv preprint arXiv:1911.00055 (2019)
19. Scarselli, F., Gori, M., Tsoi, A.C., Hagenbuchner, M., Monfardini, G.: The graph neural network model. IEEE transactions on neural networks **20**(1), 61–80 (2008)
20. Schlichtkrull, M., Kipf, T.N., Bloem, P., Van Den Berg, R., Titov, I., Welling, M.: Modeling relational data with graph convolutional networks. In: European semantic web conference. pp. 593–607. Springer (2018)
21. Selvarajah, K., Ragunathan, K., Kobti, Z., Kargar, M.: Dynamic network link prediction by learning effective subgraphs using cnn-lstm. In: 2020 International Joint Conference on Neural Networks (IJCNN). pp. 1–8. IEEE (2020)
22. Speer, R., Chin, J., Havasi, C.: Conceptnet 5.5: An open multilingual graph of general knowledge. In: Thirty-first AAAI conference on artificial intelligence (2017)
23. Suchanek, F.M., Kasneci, G., Weikum, G.: Yago: a core of semantic knowledge. In: Proceedings of the 16th international conference on World Wide Web. pp. 697–706 (2007)
24. Sun, Z., Deng, Z.H., Nie, J.Y., Tang, J.: Rotate: Knowledge graph embedding by relational rotation in complex space. arXiv preprint arXiv:1902.10197 (2019)
25. Teru, K., Denis, E., Hamilton, W.: Inductive relation prediction by subgraph reasoning. In: International Conference on Machine Learning. pp. 9448–9457. PMLR (2020)
26. Toutanova, K., Chen, D., Pantel, P., Poon, H., Choudhury, P., Gamon, M.: Representing text for joint embedding of text and knowledge bases. In: Proceedings of the 2015 conference on empirical methods in natural language processing. pp. 1499–1509 (2015)
27. Trouillon, T., Welbl, J., Riedel, S., Gaussier, É., Bouchard, G.: Complex embeddings for simple link prediction. In: International conference on machine learning. pp. 2071–2080. PMLR (2016)
28. Wang, M., Qiu, L., Wang, X.: A survey on knowledge graph embeddings for link prediction. Symmetry **13**(3), 485 (2021)

ICCS Camera Ready Version 2022
To cite this paper please use the final published version:
DOI: 10.1007/978-3-031-08757-8_60

29. Wang, Q., Mao, Z., Wang, B., Guo, L.: Knowledge graph embedding: A survey of approaches and applications. IEEE Transactions on Knowledge and Data Engineering **29**(12), 2724–2743 (2017)
30. Xiong, C., Callan, J.: Esdrank: Connecting query and documents through external semi-structured data. In: Proceedings of the 24th ACM international on conference on information and knowledge management. pp. 951–960 (2015)
31. Xiong, W., Hoang, T., Wang, W.Y.: Deeppath: A reinforcement learning method for knowledge graph reasoning. arXiv preprint arXiv:1707.06690 (2017)
32. Xu, K., Hu, W., Leskovec, J., Jegelka, S.: How powerful are graph neural networks? arXiv preprint arXiv:1810.00826 (2018)
33. Yang, B., Yih, W.t., He, X., Gao, J., Deng, L.: Embedding entities and relations for learning and inference in knowledge bases. arXiv preprint arXiv:1412.6575 (2014)
34. Yang, F., Yang, Z., Cohen, W.W.: Differentiable learning of logical rules for knowledge base reasoning. arXiv preprint arXiv:1702.08367 (2017)
35. Zhang, M., Chen, Y.: Link prediction based on graph neural networks. Advances in Neural Information Processing Systems **31**, 5165–5175 (2018)
36. Zhou, K., Dong, Y., Wang, K., Lee, W.S., Hooi, B., Xu, H., Feng, J.: Understanding and resolving performance degradation in graph convolutional networks. arXiv preprint arXiv:2006.07107 (2020)
37. Zhu, F., Wang, Y., Chen, C., Liu, G., Orgun, M., Wu, J.: A deep framework for cross-domain and cross-system recommendations. arXiv preprint arXiv:2009.06215 (2020)

ICCS Camera Ready Version 2022
To cite this paper please use the final published version:
DOI: 10.1007/978-3-031-08757-8_60