Attention-Based Aggregation Graph Networks for Knowledge Graph Information Transfer

Ming Zhao[1,2], Weijia Jia[1,2(✉)], and Yusheng Huang[1,2]

[1] Shanghai Jiao Tong University, Shanghai, China
{vanlightming,huangyusheng}@sjtu.edu.cn
[2] State of Key Lab of Internet of Things for Smart City, University of Macau, Macau, China
jiawj@um.edu.mo

Abstract. Knowledge graph completion (KGC) aims to predict missing information in a knowledge graph. Many existing embedding-based KGC models solve the Out-of-knowledge-graph (OOKG) entity problem (also known as zero-shot entity problem) by utilizing textual information resources such as descriptions and types. However, few works utilize the extra structural information to generate embeddings. In this paper, we propose a new zero-shot scenario: how to acquire the embedding vector of a relation that is not observed at training time. Our work uses a convolutional transition and attention-based aggregation graph neural network to solve both the OOKG entity problem and the new OOKG relation problem without retraining, regarding the structural neighbors as the auxiliary information. The experimental results show the effectiveness of our proposed models in solving the OOKG relation problem. For the OOKG entity problem, our model performs better than the previous GNN-based model by 23.9% in NELL-995-Tail dataset.

Keywords: Knowledge graph · Zero-shot learning · Graph Neural Network · Graph Attention Network

1 Introduction

Knowledge graphs (KGs) have been used in many applications such as information retrieval, question answering and text understanding. Standard KGs are collections of triplet (h, r, t), where h and t represent head entity and tail entity respectively, and r stands for the relation from h to t. Nowadays, large-scale KGs such as Freebase [2], WordNet [9] and DBpedia [1] have been constructed and maintained for many practical tasks. Although a knowledge graph contains millions of triplets, there are incompleteness problems, which are divided into two types: sparsity and poor scalability. Therefore knowledge graph completion (KGC) has become the most important task in knowledge graph constructions.

One of the significant issues for knowledge graph constructions is the poor scalability of KGs. Some KGE models are proposed to extend KGs automatically with OOKG entities (also called zero-shot scenario), proposed by [17].

H. W. Lauw et al. (Eds.): PAKDD 2020, LNAI 12085, pp. 542–554, 2020.
https://doi.org/10.1007/978-3-030-47436-2_41

The zero-shot scenario is summarized as that every testing triplet contains at least one OOKG entity when doing KGC tasks. These zero-shot KGE models use additional textual information, such as descriptions and types, to generate the embedding vectors for OOKG entities.

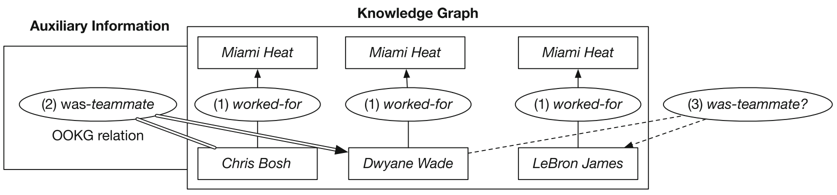

Fig. 1. OOKG relation problem.

Some KGE models [6,15] utilize additional structural information rather than textual information at testing time to solve certain OOKG entity. They generate its embedding vector based on some triplets (called auxiliary triples), and each of them contains one new entity, another in-KG entity and in-KG relation. However, the Graph Neural Network (GNN) based model [6] is effective merely on datasets that contain a small number of relations such as WordNet11 and Freebase13. Besides, this model can only handle the problem of OOKG entities but cannot deal with the problem of new relations, which is a new zero-shot scenario for OOKG relations firstly proposed in our work.

This new scenario is important because the OOKG relation is likely to be added to the knowledge graph to expand the scale and strengthen the connections between entities. As shown in Fig. 1, we want to define the new relation *"was-teammate"* between basketball players. And we know the extra information (*"Chris Bosh"*, *"was-teammate"*, *"Dwyane Wade"*) (auxiliary triplet). Then we want to infer whether there exists the relation *"was-teammate"* between *"Dwyane Wade"* and *"LeBron James"*. And it should help us to estimate that the answer is yes.

To handle the two zero-shot scenarios only using structural information (auxiliary triplets), we propose a convolutional transition and attention-based aggregation graph neural network structure. Our proposed model is inspired by GNN [6] and Graph Attention Networks [14]. We will demonstrate the whole framework in Sect. 3.

Our main contributions can be summarized as follows: (1) We propose a new approach for generating embedding vectors of Out-of-KG relations; (2) We develop a Convolutional Transition Function to transfer information for Out-of-KG entities and Out-of-KG relations in parallel; (3) We propose a Graph Attention-based Aggregation Function to merge the embeddings effectively; and (4) We verify the effectiveness of our approaches in several different datasets with different experiment settings.

2 Related Work

Knowledge graph embedding (KGE) aims to represent entities and relations into embedding vectors. Typical KGE models include TransE [3], ConvE [5], TransD [7] and etc. There are also some methods utilizing relation paths, such as TransE-NMM [10], ProjE [11]. Some other methods utilize extra textual information to represent entities and relations, for example, DKRL [17], Open-world KGC [12]. And MetaR [4] concentrate on few-shot link prediction in knowledge graph. In this problem, few triplets are given at training time, while in OOKG entities and relations problems, auxiliary triplets are given at testing time.

Recently, some KGE models have involved GNNs for the representation of OOKG entities, and they are directly related with our work. Hamaguchi et al. [6] employ GNN to build embeddings based on the knowledge transfer between neighbor entities. But Basic-GNN [6] defines two transition networks for each relation. For a knowledge graph with many relations, the triplets are not enough to fit the transition functions for the relation with few instances. The work proposed by [15] provides a logic attention network as the aggregation function in GNN. They mainly compare this LAN aggregation function with *average* method and *LSTM* method. We mainly utilize the graph attention based method as the aggregation function. In the meantime, our proposed models can deal with the OOKG scenario for relations.

3 Methodology

3.1 Notations and Problem Formulation

Firstly, we introduce some notations that are used in this paper. Let $\mathcal{E}$ be the set of *entities* and $\mathcal{R}$ be the set of *relations*. And a typical knowledge graph is denoted by $\mathcal{G} = \{(h, r, t)\} \subseteq \mathcal{E} \times \mathcal{R} \times \mathcal{E}$, where (h, r, t) means the *fact* or the *relation triplet*. Let $\mathcal{G}_{gold}$ be a set of *facts*. Any triplet in $\mathcal{G}_{gold}$ is called positive triplets. Otherwise, it is a negative triplet. Generally, we can see that $\mathcal{G} \subset \mathcal{G}_{gold}$.

Triplet Classification is a typical KGC task [13] and has become a standard benchmark for KGE methods. Let $\mathcal{H} = (\mathcal{E} \times \mathcal{R} \times \mathcal{E})\backslash\mathcal{G}$ be the set of facts that are not in the existing knowledge graph. For each triplet $f \in \mathcal{H}$, it is either a positive triplet (i.e., $f \in \mathcal{G}_{gold}$), or it is a negative triplet (i.e., $f \notin \mathcal{G}_{gold}$). A standard triplet classification limits that $\mathcal{E}$ and $\mathcal{R}$ only appear in $\mathcal{G}$.

One of the new tasks is called the OOKG (out-of-knowledge-graph) entity TC. In addition to the knowledge graph $\mathcal{G}$, new triplets $\mathcal{G}_{aux\_e}$ are given at test time. Each triplet in $\mathcal{G}_{aux\_e}$ contains exactly one OOKG entity from $\mathcal{E}_{OOKG} = \mathcal{E}(\mathcal{G}_{aux\_e})\backslash\mathcal{E}(\mathcal{G})$, one entity from $\mathcal{E}(\mathcal{G})$, where $\mathcal{E}(\mathcal{G}) = \{h|(h, r, t) \in \mathcal{G}\} \cup \{t|(h, r, t) \in \mathcal{G}\}$ and no new relations are involved. $\mathcal{G}_{aux\_e}$ gives the relationship between OOKG entities and old entities in $\mathcal{E}(\mathcal{G})$. In this setting, the task is to correctly identify the test triplets that involve the OOKG entities $\mathcal{E}_{OOKG}$ given the training set $\mathcal{G}$ and auxiliary set $\mathcal{G}_{aux\_e}$. The other new task is called OOKG relation TC. Similar to OOKG entity problem, new triplets $\mathcal{G}_{aux\_r}$ are given at test time. However, each triplet in $\mathcal{G}_{aux\_r}$ contains exactly one OOKG relation and two old entities from $\mathcal{E}(\mathcal{G})$. We denote the OOKG relations as $\mathcal{R}_{OOKG} = \mathcal{R}(\mathcal{G}_{aux\_r}) \backslash \mathcal{R}(\mathcal{G})$, where $\mathcal{R}(\mathcal{G}) = \{r|(h,r,t) \in \mathcal{G}\}$. In this new zero-shot scenario, our task is to correctly identify triplets that involve the OOKG relations $\mathcal{R}_{OOKG}$, given the training set $\mathcal{G}$ and auxiliary set $\mathcal{G}_{aux\_r}$.

3.2 Framework of Our Proposed Model

We propose a framework that can solve the above two zero-shot scenarios. The designed models can transfer information from $\mathcal{E}(\mathcal{G})$ and $\mathcal{R}(\mathcal{G})$ to $\mathcal{E}_{OOKG}$ and $\mathcal{R}_{OOKG}$. This framework consists of two models, the propagation model and the output model. The next two subsections demonstrate how we design the propagation model to satisfy the information transfer in KGs. The output model defines an objective function according to given tasks using the embeddings of entities and relations. Combining the propagation models and the output model, we can get the complete GNN model. At training time, the propagation models for entities and relations gather the information from entities' and relations' corresponding triplets, to calculate the embeddings of *h,r,t* in (h,r,t). Then the output model takes the embeddings and calculates the absolute-margin objective function. The parameters and embeddings are trained using stochastic gradient descent with backpropagation.

At testing time, for OOKG entity e, we initialize its embeddings randomly. Then we can calculate the embedding of e through the propagation model for entities using auxiliary triplets. The auxiliary triplets define the neighbors of entity e, and the embeddings of neighbors and relations are trained already so they can be used in transition functions. For attention-based aggregation, we can calculate the normalized attention value using the embeddings of neighbors, relations and the randomly initialized embedding of e. For OOKG relation r, we can also calculate the embedding of r through the propagation model for relations, following the same procedure as OOKG entities. That is to say, we can calculate the embeddings of OOKG entities and relations through extra structure information without retraining.

3.3 Propagation Model for Entities

Let $e \in \mathcal{E}(\mathcal{G})$ be an entity, and $\mathbf{v}_e \in \mathbb{R}^d$ be the d-dimensional representation vector of e. We define the propagation model by the following equation:

$$S_{head}(e) = \{T_{head}(\mathbf{v}_h, \mathbf{v}_r; h, r, e)|(h,r,e) \in \mathcal{N}_h(e)\}, \tag{1}$$

$$S_{tail}(e) = \{T_{tail}(\mathbf{v}_t, \mathbf{v}_r; e, r, t)|(e,r,t) \in \mathcal{N}_t(e)\}, \tag{2}$$

$$\mathbf{v}_e = P(S_{head}(e) \cup S_{tail}(e)), \tag{3}$$

where head neighbors $\mathcal{N}_{head}(e) = \{(h,r,e)|(h,r,e) \in \mathcal{G}\}$ and tail neighbors $\mathcal{N}_{tail}(e) = \{(e,r,t)|(e,r,t) \in \mathcal{G}\}$. The T_{head} and T_{tail} are the *transition function*: $\mathbb{R}^d \times \mathbb{R}^d \times \mathcal{E}(\mathcal{G}) \times \mathcal{R}(\mathcal{G}) \times \mathcal{E}(\mathcal{G}) \rightarrow \mathbb{R}^d$. The transition function is used to transform the vector of a node into the vector of its neighbors, and the transition function's parameters depend on the specific node pair and the edge between them. $S_{head}(e)$ contains the embeddings transformed from e's neighbors $\mathcal{N}_{head}(e)$. And $S_{tail}(e)$ contains the embeddings transformed from e's neighbors $\mathcal{N}_{tail}(e)$. The Eq. (3) represents the information aggregation function of the head set and tail set.

Transition Function. The purpose of the transition function is to define how to transfer information between the current node and its neighbors. In Hamaguchi's model, they design $2n$ independent transition functions if there are n relations in a KG. For the knowledge graph with many relations, such as NELL-995, the triplets are not enough to fit their transition functions for the relation with few instances. So we define the following two kinds of transition functions to solve this problem:

$$T_{dir}^{conv}(\mathbf{v}_{e_i}, \mathbf{v}_{r_i}; < e, e_i, r_i >) = \mathrm{ReLU}(\mathrm{BN}(Conv^{dir}(\mathbf{v}_{e_i}, \mathbf{v}_{r_i}))), \tag{4}$$

$$T_{dir}^{fcl}(\mathbf{v}_{e_i}, \mathbf{v}_{r_i}; < e, e_i, r_i >) = \mathrm{ReLU}(\mathrm{BN}(FCL^{dir}(\mathbf{v}_{e_i}, \mathbf{v}_{r_i}))), \tag{5}$$

where $dir = head$ if e_i is a head entity or $dir = tail$ if e_i is a tail entity. Given a triplet (h, r, e), T_{head}^{conv} and T_{head}^{fcl} can transform the $\mathbf{v}_h$ and $\mathbf{v}_r$ together into a temporary vector $\mathbf{v}_{h,r\rightarrow e}$. Similarly, given a triplet (e, r, t), the T_{tail}^{conv} and T_{tail}^{fcl} can transform the $\mathbf{v}_t$ and $\mathbf{v}_r$ into a temporary vector $\mathbf{v}_{t,r\rightarrow e}$. The difference between functions proposed by Hamaguchi et al. [6] and (4) (5) is that we substitute the **2n** fully-connected layers with **2** designed convolutional 2D layers or **2** fully-connected layers. Besides, the calculation complexity is lower for (5), but the feature extraction capability is higher for (4), which is tested through experiments.

The FCL^{head} function is a fully-connected layer similar to the functions in [6]. $Conv^{head}$ and $Conv^{tail}$ are two convolutional 2D layers. $Conv^{head}$ takes $\mathbf{v}_h$ and $\mathbf{v}_h$ as inputs. Firstly, we concatenate them along the first dimension into a $n \times 2$ matrix. Then we pad zeros to the matrix along the n-dimension side to get a $(n+1) \times 2$ matrix. After that, we use a 2×2 filter to extract the feature between $\mathbf{v}_h$ and $\mathbf{v}_r$. Then the output of $Conv^{head}$ is an n-dimensional vector. Similarly, $Conv^{tail}$ has the same structure with $Conv^{tail}$ except that it takes the vectors of t and r as inputs. After extracting features through convolutional 2D layers or fully-connected layers, we apply batch normalization and ReLU to the output.

Aggregation Function. Aggregation function P mentioned in Eq. (3) maps a set of vectors to a vector. The purpose of P is to merge information together from a set of vectors. For $S = \{x_i \in \mathbb{R}^d\}_{i=1}^N$, some simple pooling methods are as follows:

$$P_{sum}(S) = \sum_{i=1}^{N} x_i \quad P_{avg}(S) = \frac{1}{N}\sum_{i=1}^{N} x_i \quad P_{max}(S) = max(\{x_i\}_{i=1}^N)$$

where max is the element-wise max function. We apply these three methods separately to our models.

In addition to the simple pooling methods, we also propose an attention-based aggregation network to define the weight value for each temporary information vector inspired by the Graph Attention Network [14].

Each information transfer edge attention value is defined as Eq. (6). Entity e is the target entity. Entity set $\mathcal{E}_{neighbor}(e)$ is the neighbor entities set of e. For each $e_i \in \mathcal{E}_{neighbor}(e)$, we denote the relation between e_i and e as r_i, and e can be head entity or tail entity. $\mathbf{W}_1$ denotes the linear transformation matrix from $3d$ dimensions to d dimensions. $\mathbf{W}_2$ denotes the linear transformation matrix from d dimensions to 1 dimension. $||$ denotes the concatenation operation. The neural network mechanism used in [15] takes transformed vectors as inputs but we take the vectors of entities and relations as inputs

$$value_{<e,e_i,r_i>} = \mathrm{LeakyReLU}(\mathbf{W}_2(\mathbf{W}_1[\mathbf{v}_{e_i}||\mathbf{v}_e||\mathbf{v}_{r_i}])), \tag{6}$$

For each entity e, there are several neighbor entities and corresponding $value$. And we use the softmax function to modify these values as Eq. (7).

$$AttV_{<e,e_i,r_i>} = \frac{\exp(value_{<e,e_i,r_i>})}{\sum\limits_{e_i \in \mathcal{E}_{neighbor}(e)} \exp(value_{<e,e_i,r_i>})}, \tag{7}$$

where $(e, r_i, e_i) \in \mathcal{N}_{head}(e) \cup \mathcal{N}_{tail}(e)$ or $(e_i, r_i, e) \in \mathcal{N}_{head}(e) \cup \mathcal{N}_{tail}(e)$.

The attention-based aggregation function is defined as Eq. (8). The $\mathbf{v}_{trans}(e_i)$ is the vector calculated by the transition function.

$$P_{att}(S_{head}(e) \cup S_{tail}(e)) = \sum_{e_i \in \mathcal{E}_{neighbor}(e)} AttV_{<e,e_i,r_i>} * \mathbf{v}_{trans}(e_i), \tag{8}$$

Using the above transition functions and aggregation functions, we construct the information propagation model for entities as shown in Eq. (1)–(3). We gather the entity e's neighbors and extract their features along with specific relation r. Then we merge the information using different aggregation functions to get the embedding of e.

3.4 Propagation Model for Relations

The second propagation model is designed for relation's information transfer. We can transfer the information in $\mathcal{G}$ into $\mathcal{R}_{OOKG}$ by this model. Intuitively, we have a basic assumption that the relation's embedding vector is only related to the triplets where it shows. That is to say, given the triplets containing relation r and their corresponding entities embedding, we can represent r's embedding. We propose a propagation function based on the above assumption:

$$S_{rel}(r) = \{T_{rel}^{conv}(\mathbf{v}_h, \mathbf{v}_t; h, r, t) | (h, r, t) \in \mathcal{N}_{rel}(r)\}, \quad \mathbf{v}_r = P(S_{rel}(r)), \tag{9}$$

where $S_{rel}(r)$ contains the embeddings that are transformed from r's corresponding triplets $\mathcal{N}_{rel}(r)$. And $\mathcal{N}_{rel}(r)$ stands for the set of triplets that contains relation r. The function P is the simple pooling function mentioned in **Aggregation Function**.

The T_{rel}^{conv} transition function: $\mathbb{R}^d \times \mathbb{R}^d \times \mathcal{E}(\mathcal{G}) \times \mathcal{R}(\mathcal{G}) \times \mathcal{E}(\mathcal{G}) \rightarrow \mathbb{R}^d$ transforms the embeddings of entity h and t into a temporary embedding of relation r. The specific form is as follows:

$$T_{rel}^{conv}(\mathbf{v}_h, \mathbf{v}_t; h, r, t) = ReLU(BN(Conv^{rel}(\mathbf{v}_h, \mathbf{v}_t))), \tag{10}$$

where $Conv^{rel}$ is a convolutional 2D layer similar to $Conv^{head}$. The inputs are $\mathbf{v}_h$ and $\mathbf{v}_t$ and the parameters are independent of $Conv^{head}$. For a specific relation r in $\mathcal{R}(\mathcal{G})$, we first collect the triplets that contain r. Then we gather information from the corresponding entity pairs set $\{(h_i, t_i)\}$, using the transition function T_{rel}^{conv}. Then we mix together these temporary embeddings to get the embedding $\mathbf{v}_r$ of r.

3.5 Output Model

We use the TransE [3] based objective function as the output model. Our architecture is not limited to TransE. We can also use other KGE models like ConvE [5], for the output model.

Score Function. The score function measures the correctness of a triplet (h,r,t). Smaller scores mean that the triplet is more likely to be true. We use the same score function as in TransE:

$$f(h, r, t) = \|\mathbf{v}_h + \mathbf{v}_r - \mathbf{v}_t\|, \tag{11}$$

where $\mathbf{v}_h$, $\mathbf{v}_r$, $\mathbf{v}_t$ are the embedding vectors of the head, relation and tail, respectively.

Absolute-Margin Objective Function. We utilize the following objective function called the absolute-margin objective function [6]:

$$\mathcal{L} = \sum_{i=1}^{N} f(h_i, r_i, t_i) + [\tau - f(h_i^{'}, r_i, t_i^{'})]_+ \tag{12}$$

where τ is a hyperparameter, called the margin. $[]_+$ means when the expression is less than zero, we eliminate it. This absolute-margin objective function aims to train the scores of positive triplets towards zero, whereas the scores of negative triplets are at least τ.

Table 1. Specifications of the triplet classification datasets.

	Relations	Entities	Training triplets	Validation triplets	Test triplets
WordNet11	11	38696	112581	5218	21088
NELL-995	200	75492	149678	1086	7984

Table 2. Standard TC Result. Method with prefix * is our proposed model.

Method	WordNet11	NELL-995
TransE	67.6	74.8
TransD	69.4	71.4
NMM	82.4	84.4
Basic-GNN-max	<u>87.4</u>	<u>87.4</u>
*ConvEntity-avg	85.6	79.7
*FCLEntity-Att	**87.9**	**93.2**
*ConvEntity-ConvRelation-avg	85.9	86.5

4 Experiment

4.1 Experimental Setup

We evaluate the effectiveness of our model on two datasets WordNet11 and NELL-995. To shorten the information transfer time, we sample the neighbor entities randomly when an entity has too many neighbors, and we also sample the corresponding entity pairs randomly when a relation is related to too many of entity pairs. The maximum number of corresponding information sources is set to be 64. The parameters are trained by the stochastic gradient descent with backpropagation. Specifically, we use the Adam optimization method. The step size of Adam is $\alpha_1/(\alpha_2 \cdot k + 1.0)$, where k indicates the number of epochs performed, $\alpha_1 = 0.01$, and $\alpha_2 = 0.0001$. The mini-batch size is 5000 and the number of training epochs is 150 in every experiment. Moreover, the dimension of the embedding space is 200. We implement our models using the neural network library Chainer http://chainer.org/.

In the experiment part, we define three versions of our model. The first version *ConvEntity-PoolingMethods* means that only entity embeddings are obtained based on the convolutional transition functions and simple pooling methods. The second version *FCLEntity-Att* means that only entity embeddings are obtained based on the fully-connected transition function and attention-based aggregation function. We use the pretrained TransE embeddings to initialize the entity and relation embeddings. The third version is *ConvEntity-ConvRelation-PoolingMethods*, which means the embeddings of entities and relations are all obtained based on the convolutional transition functions and simple pooling methods.

4.2 Standard Triplet Classification

Firstly, we compare our models with some baselines in the standard setting. There are no OOKG entities and relations involved at testing time.

Datasets. We use WordNet11, NELL-995 for evaluation. We use *corrupted* triplets generated from positive triplets as the negative triplets of validation and test sets to evaluate *Triplet Classification*. The specifications on WordNet11 and NELL-995 are shown in Table 1. Half of the validation and test sets are negative triplets, and they are included in the numbers of validation and test triplets. During the training time, we also use corrupted triplets as negative samples. From a positive triplet (h, r, t) in knowledge graph $\mathcal{G}$, a corrupted triplet is generated by substituting a random entity sampled from $\mathcal{E}(\mathcal{G})$ for h or t with the 'Bernoulli' trick [16].

Table 3. Specifications of the OOKG entity datasets.

	WN11			NELL-995		
	Head	Tail	Both	Head	Tail	Both
Training triplets	99963	78763	71097	137846	138487	128956
Validation triplets	4108	3122	2759	838	485	411
OOKG entities	1034	2627	3319	2178	116	2183
Test triplets	2969	2880	2708	2202	2391	2089
Auxiliary entities	6791	16193	19218	5210	6263	9825
Auxiliary triplets	12376	31770	38285	9900	7853	15068

Table 4. Results of the OOKG entity experiment.

Method	WN11			NELL-995		
	Head	Tail	Both	Head	Tail	Both
Basic-GNN-avg	82.1	<u>73.5</u>	66.1	54.7	59.6	54.8
LAN	**84.5**	**75.6**	**72.6**	<u>92.8</u>	64.4	<u>65.2</u>
*ConvEntity-avg	71.1	69.5	61.8	80.7	<u>70.2</u>	65.0
*FCLEntity-Att	<u>82.6</u>	72.1	<u>68.6</u>	**93.0**	**88.3**	**87.3**

Results. The results are shown in Table 2. As the table shows, in both WordNet11 and NELL-995 datasets, our proposed model *FCLEntity-Att* achieves the best accuracy.

4.3 OOKG Entity Experiment

In this part, we compare our model *ConvEntity*, *FCLEntity-Att* with Basic-GNN [6] and LAN [15]. Our proposed models *ConvEntity* and *FCLEntity* can solve the scenario where there are only OOKG entities.

Datasets. We process NELL-995 dataset to construct several datasets for our OOKG entity experiment following a similar protocol used in [6]. Also, we directly use the datasets WN11 released by [6].

We take the NELL-995 as an example to explain the process, which consists of two steps: choosing OOKG entities, filtering and splitting triplets. The details of the generated OOKG entity datasets are shown in Table 3.

1. *Choosing OOKG entities.* We first randomly select 3000 triplets from the NELL-995 test file. For these 3000 triplets, we choose the initial OOKG candidates in three different ways (thereby yielding three datasets in total), called Head, Tail, and Both settings. For the Head setting, we choose the head entities of the 3000 triplets as candidates. The Tail setting is similar, but with tail entities regarded as candidates. In the Both setting, all the head and tail entities are seen as candidates. We consider the candidates set as $\mathcal{T}_c$ and the final OOKG entity set as $\mathcal{T}$. For every candidate $e \in \mathcal{T}_c$, if it does not have any neighbor in the original training set, such an entity is filtered out, yielding the final $\mathcal{T}$.

Table 5. Specifications of the OOKG relation datasets.

	NELL-995		
	1000	1500	2000
Training triplets	147088	147546	146422
Validation triplets	653	732	576
Auxiliary triplets	2094	1639	267
OOKG relations	3	3	5
Test triplets	819	1202	1629

Table 6. Results of the OOKG relation experiment.

Method	NELL-995		
	1000	1500	2000
Baseline-sum	<u>58.4</u>	<u>61.5</u>	56.1
Baseline-max	41.8	38.8	44.1
Baseline-avg	58.0	58.7	<u>56.4</u>
*ConvEntity-ConvRelation-avg	**84.1**	**84.8**	**73.2**

2. *Filtering and splitting triplets.* After getting the OOKG entity set $\mathcal{T}$, we choose the new training set and auxiliary set. For a triplet in the original training set, if it only contains non-OOKG entities, it is added to the new training set. If it only contains one OOKG entity, it is added to the auxiliary set. For the validation triplets, we remove the triplets containing OOKG entities. For the test triplets, we use the 3000 triplets in the NELL-995 test file in step (1), with discarding the triplets that contain removed OOKG candidates.

Results. The results are shown in Table 4. In three datasets generated from NELL-995, our proposed *FCLEntity-Att* model outperforms other models. In three datasets generated from WN11, LAN proposed by [15] achieves the best results. But our proposed model performs better than the original Basic-GNN method [6]. The results demonstrate that our proposed convolutional transition function and the attention-based aggregation function are effective for the Out-of-KG entity experiments in datasets with many relations. Because of its relation-specific transition function and logic attention mechanism, LAN [15] has better generalization performance in WN11.

4.4 OOKG Relation Experiment

This part is to verify the effectiveness of our relation information transfer model, *ConvEntity-ConvRelation-PoolingMethods*, which can generate zero-shot relation embeddings.

Datasets. We process the NELL-995 dataset to construct three datasets for our OOKG relation experiment with different quantities of OOKG relations. The process consists of two steps: choosing OOKG relations, filtering and splitting triplets. The details of the generated OOKG datasets are shown in Table 5.

1. *Choosing OOKG relations candidates.* To decide OOKG relations, we first randomly select N = 1000, 1500, 2000 triplets from the NELL-995 test file. For these N triplets, we choose the relations as the initial OOKG relation candidates set called $\mathcal{K}_c$.
2. *Filtering and splitting triplets.* After getting the OOKG relations candidates set $\mathcal{K}_c$, we divide the original training set into two parts. The first part contains the triplets whose relation is not in $\mathcal{K}_c$, called the new training set. The second part contains the triplets whose relation is in $\mathcal{K}_c$, called auxiliary set, with discarding the triplet which contains entities not shown in the new training set. And for the N test triplets mentioned before, we discard these triplets which contain entities not shown in the new training set. For the original validation set, we only choose triplets which only contain entities and relations shown in the new training set as new validation triplets.

Results. The results are shown in Table 6. We use the following method as the baseline in this experiment since this zero-shot scenario is proposed firstly in our work. For an OOKG relation r, based on the auxiliary set, we use the basic assumption $h + r = t$ to compute the temporary relation embeddings set $\{r_i | r_i = t_i - h_i\}$. And we apply the simple pooling function to this set to get the representation vector of r. We use the hyperparameter and settings of TransE in the work of [8]. As the table shows, our proposed model *ConvEntity-ConvRelation-avg* outperforms the baseline methods in all three datasets generated from NELL-995. This experiment shows that the convolutional transition function is also effective in the relation information transfer.

5 Conclusion

In this paper, we propose the convolutional transition and attention-based aggregation graph neural network structure to solve the two zero-shot scenarios where entities and relations are not involved at training time. In particular, the zero-shot scenario for relations is firstly involved in our work. Through the OOKG entity experiment and the OOKG relation experiment, we evaluate the effectiveness of the convolutional transition function and the graph attention-based aggregation function. Our proposed models outperform baseline models significantly in the three experiment settings.

Acknowledgement. This work is supported by Chinese National Research Fund (NSFC) Key Project No. 61532013 and No. 61872239; 0007/2018/A1, 0060/2019/A1, DCT-MoST Joint-project No. 025/2015/AMJ of Science and Technology Development Fund, Macao S.A.R (FDCT), China, and University of Macau Grant Nos: MYRG2018-00237-FST, CPG2020-00015-IOTSC and SRG2018-00111-FST.

References

1. Auer, S., Bizer, C., Kobilarov, G., Lehmann, J., Cyganiak, R., Ives, Z.: DBpedia: a nucleus for a web of open data. In: Aberer, K., et al. (eds.) ASWC/ISWC -2007. LNCS, vol. 4825, pp. 722–735. Springer, Heidelberg (2007). https://doi.org/10.1007/978-3-540-76298-0_52
2. 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. ACM (2008)
3. Bordes, A., Usunier, N., Garcia-Duran, A., Weston, J., Yakhnenko, O.: Translating embeddings for modeling multi-relational data. In: Advances in Neural Information Processing Systems, pp. 2787–2795 (2013)
4. Chen, M., Zhang, W., Zhang, W., Chen, Q., Chen, H.: Meta relational learning for few-shot link prediction in knowledge graphs. In: EMNLP-IJCNLP, pp. 4208–4217 (2019)
5. Dettmers, T., Minervini, P., Stenetorp, P., Riedel, S.: Convolutional 2D knowledge graph embeddings. In: Thirty-Second AAAI Conference on Artificial Intelligence (2018)
6. Hamaguchi, T., Oiwa, H., Shimbo, M., Matsumoto, Y.: Knowledge transfer for out-of-knowledge-base entities: a graph neural network approach. In: IJCAI, pp. 1024–1034 (2017)
7. Ji, G., He, S., Xu, L., Liu, K., Zhao, J.: Knowledge graph embedding via dynamic mapping matrix. In: Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers), vol. 1, pp. 687–696 (2015)
8. Lin, Y., Liu, Z., Sun, M., Liu, Y., Zhu, X.: Learning entity and relation embeddings for knowledge graph completion. In: Twenty-Ninth AAAI Conference on Artificial Intelligence (2015)
9. Miller, G.A.: WordNet: a lexical database for English. Commun. ACM **38**(11), 39–41 (1995)

10. Nguyen, D.Q., Sirts, K., Qu, L., Johnson, M.: Neighborhood mixture model for knowledge base completion. In: CoNLL, pp. 40–50 (2016)
11. Shi, B., Weninger, T.: ProjE: embedding projection for knowledge graph completion. In: Thirty-First AAAI Conference on Artificial Intelligence (2017)
12. Shi, B., Weninger, T.: Open-world knowledge graph completion. In: Thirty-Second AAAI Conference on Artificial Intelligence (2018)
13. Socher, R., Chen, D., Manning, C.D., Ng, A.: Reasoning with neural tensor networks for knowledge base completion. In: Advances in Neural Information Processing Systems, pp. 926–934 (2013)
14. Veličković, P., Cucurull, G., Casanova, A., Romero, A., Lio, P., Bengio, Y.: Graph attention networks. arXiv preprint arXiv:1710.10903 (2017)
15. Wang, P., Han, J., Li, C., Pan, R.: Logic attention based neighborhood aggregation for inductive knowledge graph embedding. In: Proceedings of the AAAI Conference on Artificial Intelligence, vol. 33, pp. 7152–7159 (2019)
16. Wang, Z., Zhang, J., Feng, J., Chen, Z.: Knowledge graph embedding by translating on hyperplanes. In: Twenty-Eighth AAAI Conference on Artificial Intelligence (2014)
17. Xie, R., Liu, Z., Jia, J., Luan, H., Sun, M.: Representation learning of knowledge graphs with entity descriptions. In: Thirtieth AAAI Conference on Artificial Intelligence (2016)