Knowledge Graph Entity Type Prediction with Relational Aggregation Graph Attention Network

Changlong Zou, Jingmin An, and Guanyu Li(✉)

Faculty of Information Science and Technology, Dalian Maritime University, Dalian, Liaoning, China
liguanyu@dlmu.edu.cn

Abstract. Most of the knowledge graph completion methods focus on inferring missing entities or relations between entities in the knowledge graphs. However, many knowledge graphs are missing entity types. The goal of entity type prediction in the knowledge graph is to infer the missing entity types that belong to entities in the knowledge graph, that is, (entity, entity type=?). At present, most knowledge graph entity type prediction models tend to model entities and entity types, which will cause the relations between entities to not be effectively used, and the relations often contain rich semantic information. To utilize the information contained in the relation when performing entity type prediction, we propose a method for entity type prediction based on relational aggregation graph attention network (RACE2T), which consists of an encoder relational aggregation graph attention network (FRGAT) and a decoder (CE2T). The encoder FRGAT uses the scoring function of the knowledge graph completion method to calculate the attention coefficient between entities. This attention coefficient will be used to aggregate the information of relations and entities in the neighborhood of the entity to utilize the information of the relations. The decoder CE2T is designed based on convolutional neural network, which models the entity embeddings output by FRGAT and entity type embeddings, and performs entity type prediction. The experimental results demonstrate that the method proposed in this paper outperforms existing methods. The source code and dataset for RACE2T can be downloaded from: https://github.com/GentlebreezeZ/RACE2T.

Keywords: Knowledge graph · Entity type · Relational aggregation · Attention · Scoring function · Convolutional neural network

1 Introduction

Knowledge graph stores information mainly in the triple [5], denoted as (e_i, r_k, e_j), where e_i is the head entity, e_j is the tail entity, and r_k is the relation between e_i and e_j. Besides the triples, knowledge graphs usually contain many entity type instances in the form of entity-entity type tuples [30] (denoted as (e, t)), indicating that an entity e is of a certain entity type t, for example, (*Chicago*, *Film*) and

P. Groth et al. (Eds.): ESWC 2022, LNCS 13261, pp. 39–55, 2022.
https://doi.org/10.1007/978-3-031-06981-9_3

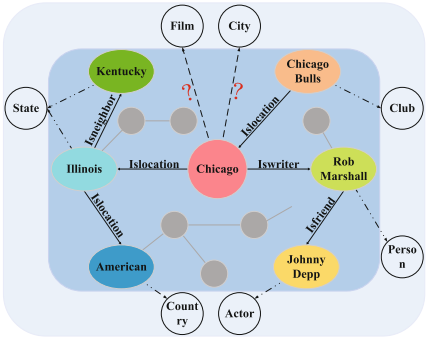


Fig. 1. Knowledge graph triples. The solid shape represents the entity; the hollow shape represents the entity type; the solid line represents the relation; the dotted line points to the entity type.

(*Chicago*, *City*). The entity type information in the knowledge graphs is widely used in various downstream tasks, such as entity alignment [32], entity linking [3], knowledge graph completion [25]. Missing entity types can undermine the effectiveness of algorithms in such tasks. However, knowledge graphs, especially cross-domain knowledge graphs, often face quality problems with incomplete entity type information. For example, 10% of the entities with type */music/artister* in FB15K [2], do not have type */people/person* in Freebase [13].

The missing entity type of knowledge graph can be solved by entity type prediction, which is a subproblem of knowledge graph completion. Early methods for predicting entity types in knowledge graphs are mainly based on probability distributions, such as SDtype [19]. Recently, representation learning [7] has gradually become the basis of knowledge graph-related research. For knowledge graphs, representation learning is the learning low-dimensional embedding vector representations of objects in the knowledge graph by methods such as machine learning. The learned embedding vectors preserve the semantic information of the objects in the knowledge graph and can be used for various downstream tasks based on the knowledge graph. In this paper, we mainly use representation learning to infer the missing entity types of entities in the knowledge graph.

At present, most entity type prediction models based on representation learning directly model entities and entity types, such as ETE [13], and ConnectE [31]. Thus these models have a common drawback of not effectively utilizing the knowledge graph triples. To be precise, the information of relations is not utilized, or these models deal with each entity-entity type tuple independently, without utilizing the semantically rich relations inherent in the neighborhood of entities in the knowledge graph. As shown in Fig. 1, the semantic information of the relations *Islocation* and *Iswriter* helps to infer that the type of entity *Chicago* may include *City* and *Film*.

GAT (Graph Attention Network) [22] is an effective tool for generating knowledge graph entity embeddings. It assigns different weights to entities in the neighborhood of the entity according to their importance. This weight will be used

to aggregate the neighborhood information of the entity. However, GAT cannot utilize relational information.

Based on the above statements, in this work, we introduce a method for entity type prediction based on relational aggregation graph attention network (RACE2T), consisting of an encoder relational aggregation graph attention network (FRGAT) and a decoder CE2T. The encoder FRGAT is designed based on GAT and is mainly used to utilize relation information. FRGAT uses the scoring function of the knowledge graph completion method to calculate the attention between entities and entities in its neighborhood and uses that attention to aggregate information about entities and relations. Meanwhile, considering the limited ability of expressiveness of existing entity type prediction models, we propose a convolutional neural network-based entity type prediction model CE2T as a decoder. CE2T is composed of convolution, projection, and inner product layers. It models the entity embeddings output by FRGAT and entity type embeddings and performs entity type prediction. We demonstrated the effectiveness of our proposed model on the FB15K and FB15KET and YAGO43K and YAGO43KET datasets and obtained advanced results.

2 Related Works

2.1 Knowledge Graph Completion Models

Translation-based models treat relations as translations from head entities to tail entities and use energy-based scoring functions. The basic idea of the TransE [2] model is that if the triple (e_i, r_k, e_j) holds, then the sum of the head entity embedding and the relation embedding should be as close as possible to the tail entity embedding, i.e., $\boldsymbol{e}_i + \boldsymbol{r}_k \approx \boldsymbol{e}_j$. TransH [23] and TransD [6] use a projection strategy and are extensions of TransE.

Semantic matching models use scoring functions based on the similarity of head and tail entities under a given relation. RESCAL [18] follows a relational learning approach based on a tensor factorization model that considers the inherent structure of relational data. DisMult [29] is a simplified version of RESCAL and uses a basic bilinear scoring function to match the underlying semantics of entities and relations in vector space. HOLE [17] combines the expressive ability of RESCAL with the simplicity of DisMult through the use of unique circular correlation operation. In addition to translation models and semantic matching models, there are ConvE [4], ConvKB[1] [15], and CapsE[2] [16], based on convolutional neural networks. Meanwhile, some models use additional information for knowledge graph completion [24].

In this paper, we will use the scoring functions of TransE [2], TransH [23], TransD [6], and DisMult [29] to calculate the attention between entities.

[1] https://github.com/daiquocnguyen/ConvKB.
[2] https://github.com/daiquocnguyen/CapsE.

2.2 Knowledge Graph Entity Type Prediction Models

CUTE [26] is a cross-language knowledge graph entity type prediction model, which mainly uses cross-language entity links between Chinese and English entities to construct training data. However, CUTE is based on non-representation learning. MuLR [28] learns multi-level embedding representations of entities through character, word, entity descriptions, and entity embeddings and then performs entity type prediction. HMGCN [9] is a knowledge graph entity type prediction model based on GCN [11], which considers relations, entity description information, and Wikipedia categories. Cat2Type [1] is similar to HMGCN in that both use Wikipedia categories for knowledge graph entity type prediction. APE [8] utilizes the attribute, structure, and type information of entities in the knowledge base for entity type prediction and learns entity embeddings through neural networks. FIGMENT [27] is mainly used to judge the types of entities in the corpora, including global models and context models. However, it additionally requires a large annotated corpora.

In short, the most significant difference between RACE2T and MuLR [28], FIGMENT [27], APE [8], HMGCN [9], and Cat2Type [1] is that RACE2T mainly uses entity-entity type tuples (e, t) and triples to learn embedding representations for objects. However, these models do not, and they usually require additional information (e.g., entity description information, corpus). Therefore, the following mainly introduces entity type prediction models that use entity-entity type tuples and triples for modeling.

Ref [14] proposed two knowledge graph entity type prediction models for encoding entity-entity type tuples (e, t), namely linear model and embedding model. The scoring function of the linear model is $\phi(e, t) = \boldsymbol{t}^{\mathrm{T}}\boldsymbol{e}$, where $\boldsymbol{e}$ is the entity embeddings and $\boldsymbol{t}$ is the entity type embeddings. The scoring function of the embedding model is $\phi(e, t) = \boldsymbol{t}^{\mathrm{T}}\boldsymbol{V}\boldsymbol{U}^{\mathrm{T}}\boldsymbol{e}$, where $\boldsymbol{U}$ and $\boldsymbol{V}$ are the projection matrices. However, both models do not use the knowledge graph triples, or they do not use the relations between entities.

The knowledge graph entity type prediction model proposed in the Ref [13] uses an asynchronous approach to learn the embedding representation of entities, relations and entity types. First, using knowledge graph completion methods, such as RESCAL [18], TransE [2], HOLE [17], and ContE [12], learn the entity embedding $\boldsymbol{e}$. Second, keep the entity embeddings $\boldsymbol{e}$ unchanged during training, and update the entity type embeddings by minimizing the distance between the entity embedding and the entity type embedding, namely, RESCAL-ET, HOLE-ET, TransE-ET, and ETE[3]. Their scoring function is $\phi\,(e, t) = \|\boldsymbol{e} - \boldsymbol{t}\|_{\ell 1}$, where $\|\boldsymbol{x}\|_{\ell 1}$ represents the $\ell 1$ norm of the vector $\boldsymbol{x}$. Although these methods use the relations between entities during training, the semantic information the relations is not utilized in making entity type predictions.

ConnectE[4] [31] uses entity-entity type tuples and entity type triples[5] for training and entity type prediction. However, entity type triples are created

[3] https://github.ncsu.edu/cmoon2/kg.
[4] https://github.com/Adam1679/ConnectE.
[5] For details of entity type triples, see Ref [31].

does not consider the semantics that entities represent when they correspond to different types. Meanwhile, entity type triples lead to data leakage in the test set. The scoring function of ConnectE is: $\phi(e,t) = \|\boldsymbol{M} \cdot \boldsymbol{e} - \boldsymbol{t}\|_{\ell 2}$, where $\boldsymbol{M}$ is projection matrix, $\|\boldsymbol{x}\|_{\ell 2}$ represents the $\ell 2$ norm of the vector $\boldsymbol{x}$. ConnectE uses an asynchronous approach to learn embeddings of entities, relations and entity types, and its training process is divided into three stages. Firstly, the model uses TransE [2] to train entity embeddings and relation embeddings. Secondly, the model trains the entity type embeddings and the projection matrix $\boldsymbol{M}$ by minimizing $\phi(e,t)$. Finally, the entity type triples are trained using TransE and only the embedding of the relations is changed, the entity type embedding remains unchanged.

It can be concluded that none of the above entity type prediction models do not effectively utilize the relations in the knowledge graph triple. Although ConnectE [31] utilizes relations through entity type triples, the entity type triples cause the leakage of the test set. In order to effectively use the relation information when predicting entity types, this paper uses the encoder FRGAT to aggregate the informations of entities and relations in the neighborhood of a given entity and uses the decoder CE2T to predict the entity type of the knowledge graph. Meanwhile, these methods mainly adopt an asynchronous way to learn the embeddings of objects (entity, relation and entity type). While the method in this work uses synchronous way to learn the embeddings of objects.

3 Methods

The method (RACE2T) in this paper adopts the form of encoder-decoder. The encoder is FRGAT. The decoder is CE2T, which is designed based on the convolutional neural network and is specially used to predict the entity type of knowledge graph. The overall framework is shown in Fig. 2.

3.1 Problem Definition and Symbol

The goal of knowledge graph entity type prediction is to infer the type t of a given entity e. Entity initial embedding matrix $\boldsymbol{E} \in \mathbb{R}^{|\mathcal{E}| \times D}$, relation initial embedding matrix $\boldsymbol{R} \in \mathbb{R}^{|\mathcal{R}| \times D}$, entity type embedding matrix $\boldsymbol{T} \in \mathbb{R}^{|\mathcal{T}| \times \ell}$, where $\mathcal{E}$, $\mathcal{R}$ and $\mathcal{T}$ respectively represent the collection of all entities, relations, and entity types, $|\mathcal{E}|$, $|\mathcal{R}|$, and $|\mathcal{T}|$ respectively represent the number of entities, relations, and entity types, D is the dimension of the initial embedding vector of entities and relations, and ℓ is the dimension of the embedding vector of entity types. The output matrix of the last layer of FRGAT is $\boldsymbol{E}_0 \in \mathbb{R}^{|\mathcal{E}| \times d}$, and d represents the dimension of FRGAT output embeddings.

3.2 Encoder: FRGAT

For a triple (e_i, r_k, e_j), its entity and relation embeddings are $\boldsymbol{e}_i$, $\boldsymbol{r}_k$ and $\boldsymbol{e}_j$, respectively. To make the model obtain sufficient expressive ability, the input

entity and relation embeddings are converted into a higher-dimensional embedding using projection operation: $\boldsymbol{h}_i = \boldsymbol{e}_i \boldsymbol{W}_1$, $\boldsymbol{z}_k = \boldsymbol{r}_k \boldsymbol{W}_2$, and $\boldsymbol{h}_j = \boldsymbol{e}_j \boldsymbol{W}_1$, where $\boldsymbol{e}_i \in \boldsymbol{E}$, $\boldsymbol{r}_k \in \boldsymbol{R}$, $\boldsymbol{e}_j \in \boldsymbol{E}$, $\boldsymbol{W}_1 \in \mathbb{R}^{D \times D_1}$, $\boldsymbol{W}_2 \in \mathbb{R}^{D \times D_1}$, $D_1 > D$. At the same time, $\boldsymbol{W}_1 = \boldsymbol{W}_2$ will be restricted to ensure that entity embedding and relation embedding are in the same semantic space after projection. Then, in FRGAT, the calculation equation for the attention between the entity e_i and the entity e_j is as follows:

$$a_{ijk} = \sigma\left(f\left(\boldsymbol{h}_i, \boldsymbol{z}_k, \boldsymbol{h}_j\right)\right) \tag{1}$$

where $\sigma(\cdot)$ represents the activation function, a_{ijk} represents the importance (attention) of e_j to e_i, $f(\cdot,\cdot,\cdot)$ represents the scoring function in the knowledge graph completion method. Since there may be more than one other entities in the first-order neighborhood of entity e_i, we use *softmax* to normalize a_{ijk}:

$$\alpha_{ijk} = soft\max\left(a_{ijk}\right) = \frac{\exp\left(a_{ijk}\right)}{\sum_{n \in \mathcal{N}_i} \sum_{r \in \mathcal{R}_{in}} \exp\left(a_{inr}\right)} \tag{2}$$

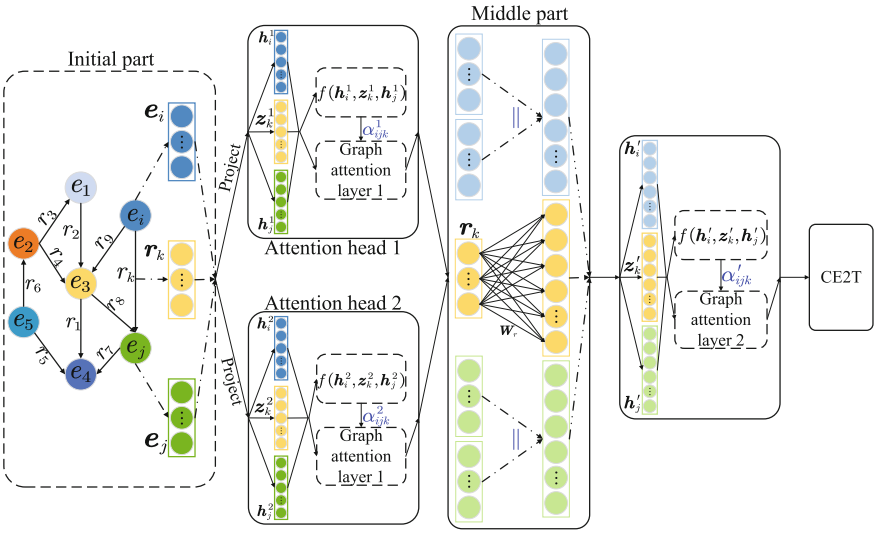


Fig. 2. The overall framework of RACE2T. The FRGAT in this figure includes two graph attention layers. The first layer includes two attention heads, and the second layer includes one.

where $\mathcal{N}_i$ is the entity set contained in the first-order neighborhood of entity e_i, $\mathcal{R}_{in}$ is the set of relations between linked entity e_i and the entities contained in $\mathcal{N}_i$, and α_{inr} represents the importance of entity e_n to entity e_i. To aggregate the information of entities and relations, we use the message propagation mechanism proposed in Ref [21] to perform a combination operation on the embedding vector

of the entity and the relation in the form of $\mathbb{R}^{\mathrm{D}} \times \mathbb{R}^{\mathrm{D}} \rightarrow \mathbb{R}^{\mathrm{D}}$. In this case, the new embedding of the entity e_i is expressed as:

$$\boldsymbol{h}_i^{'} = \sigma\left(\sum_{j \in \mathcal{N}_i} \sum_{k \in \mathcal{R}_{ij}} \alpha_{ijk} \psi\left(\boldsymbol{h}_j, \boldsymbol{z}_k\right)\right) \tag{3}$$

where $\psi(\cdot, \cdot)$ represents combination operation.

Similar to GAT [22], a connected and independent multi-head attention mechanism is used to stabilize the learning process of the model and enhance the generalization ability of the model. Under the multi-head attention mechanism: $\boldsymbol{h}_i^m = \boldsymbol{e}_i \boldsymbol{W}_1^m$, $\boldsymbol{z}_k^m = \boldsymbol{r}_k \boldsymbol{W}_2^m$, $\boldsymbol{h}_j^m = \boldsymbol{e}_j \boldsymbol{W}_1^m$, and $\alpha_{ijk}^m = soft\max\left(\sigma\left(f\left(\boldsymbol{h}_i^m, \boldsymbol{z}_k^m, \boldsymbol{h}_j^m\right)\right)\right)$, where $\boldsymbol{W}_1^m \in \mathbb{R}^{D \times D_1}$, $\boldsymbol{W}_2^m \in \mathbb{R}^{D \times D_1}$, m represents the m-th attention head, as shown in the *attention head 1 and attention head 2* of Fig. 2. The new embedding of entity e_i is expressed as:

$$\boldsymbol{h}_i^{'} = \Big\Vert_{m=1}^{M} \sigma\left(\sum_{j \in \mathcal{N}_i} \sum_{k \in \mathcal{R}_{ij}} \alpha_{ijk}^m \psi\left(\boldsymbol{h}_j^m, \boldsymbol{z}_k^m\right)\right) \tag{4}$$

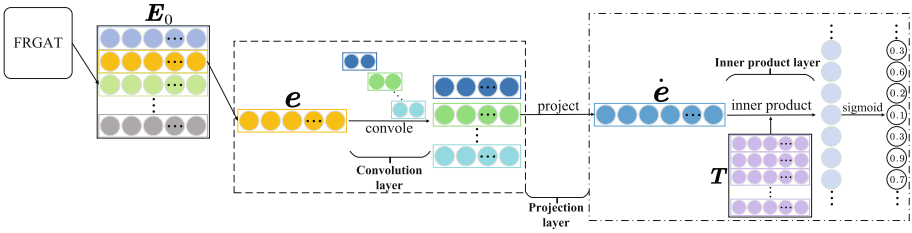


Fig. 3. The framework of CE2T.

where M represents the number of heads in the multi-head attention mechanism, and $\|$ represents the vector connect operation, as shown in the *middle part* of Fig. 2. At the same time, in order to use the embedding vector of the relation in the following aggregation process, the conversion matrix $\boldsymbol{W}_r \in \mathbb{R}^{D \times M \cdot D_1}$ is used to perform a linear transformation on the embedding vector of the input relation. The relation embedding after transformation is expressed as:

$$\boldsymbol{z}_k^{'} = \boldsymbol{r}_k \boldsymbol{W}_r \tag{5}$$

To reduce the number of parameters, we use the average to obtain the final embedding vector of the entity in the last layer of the FRGAT model instead of linking embeddings from multiple attention heads. The final entity embedding vector is used as the input of the decoder (CE2T), which is expressed as follows:

$$\boldsymbol{h}_i^{'} = \sigma\left(\frac{1}{M} \sum_{m=1}^{M} \sum_{j \in \mathcal{N}_i} \sum_{k \in \mathcal{R}_{ij}} \alpha_{ijk}^m \psi\left(\boldsymbol{h}_j^m, \boldsymbol{z}_k^m\right)\right) \tag{6}$$

3.3 Decoder: CE2T

Inspired by the application of convolutional neural network in knowledge graph completion [4,15], we propose a convolutional neural network-based knowledge graph entity type prediction model (CE2T[6]) as the decoder of this paper. The framework of CE2T is shown in Fig. 3.

The scoring function of CE2T is defined as:

$$\phi(e,t) = \sigma\left(vec\left(\sigma\left(\boldsymbol{e} * \boldsymbol{\Omega}\right)\right) \cdot \boldsymbol{M}\right)\boldsymbol{t} \tag{7}$$

where $\boldsymbol{e} \in \mathbb{R}^d$ represents the final embedding of the entity output by FRGAT, $\boldsymbol{t} \in \mathbb{R}^\ell$ represents the embedding vector of the entity type, $\boldsymbol{\Omega}$ represents the set of convolution kernels, $\boldsymbol{M}$ represents the projection matrix, $*$ represents convolution operation, and $vec(\cdot)$ represents vectorization operation.

As shown in Fig. 3, during the forward propagation process, the entity embedding $\boldsymbol{e}$ of the CE2T input needs to be found in the entity embedding matrix $\boldsymbol{E}_0$ of the FRGAT output. Suppose the convolution kernel of the convolution layer $\boldsymbol{\Omega} \in \mathbb{R}^{|\boldsymbol{\Omega}| \times 1 \times f}$, the step size is c_s, where $|\boldsymbol{\Omega}|$ represents the number of convolution kernels, and $1 \times f$ is the size of the convolution kernel. Then the output of the convolutional layer is $\boldsymbol{\mathcal{F}} = \sigma\left(\boldsymbol{e} * \boldsymbol{\Omega}\right) \in \mathbb{R}^{|\boldsymbol{\Omega}| \times a \times b}$, where $a = 1$, $b = (d - f)/c_s + 1$. Then reshape $\boldsymbol{\mathcal{F}}$ into vector $vec(\boldsymbol{\mathcal{F}}) \in \mathbb{R}^{|\boldsymbol{\Omega}| \cdot a \cdot b}$ as the input of the projection layer.

The function of the projection layer is to project the features of the entity embedding vector extracted by the convolutional layer into the ℓ-dimensional space. Both the CE2T model and the ConnectE model use a similar projection strategy, but the ConnectE [31] model projects the embedding vector of the entity, while the CE2T model projects the feature vector of the entity output by the convolutional layer. The weight of the projection layer is $\boldsymbol{M} \in \mathbb{R}^{|\boldsymbol{\Omega}| \cdot a \cdot b \times \ell}$. The entity projection vector projected to the ℓ-dimensional space is: $\dot{\boldsymbol{e}} = \sigma\left(vec\left(\boldsymbol{\mathcal{F}}\right) \cdot \boldsymbol{M}\right) \in \mathbb{R}^\ell$. Finally, entity e and entity type t similarity score is calculated by the inner product operation between the entity projection vector $\dot{\boldsymbol{e}}$ and the entity type embedding $\boldsymbol{t}$.

3.4 Training

To accelerate the training of the RACE2T model, we use the 1-N scoring proposed in the Ref [4], i.e., we score both the projection vector of an entity and the embedding vector of all entity types, as shown in the *inner product layer* of Fig. 3. At the same time, to minimize the cross-entropy loss to train the parameters of RACE2T, we use the sigmoid function to normalize $\phi(e,t)$ to between 0 and 1, which is $\boldsymbol{p} = sigmoid\left(\phi(e,t)\right)$. The loss function of RACE2T is defined as:

$$\mathcal{L} = -\frac{1}{|\mathcal{T}|}\sum_{i=1}^{|\mathcal{T}|} \boldsymbol{y}_i \log\left(\boldsymbol{p}_i\right) + \left(1 - \boldsymbol{y}_i\right)\log\left(1 - \boldsymbol{p}_i\right) \tag{8}$$

[6] https://github.com/GentlebreezeZ/CE2T.

where $\boldsymbol{y} \in \mathbb{R}^{|\mathcal{T}|}$ is the binary label vector. If (e, t) is true, the corresponding position of $\boldsymbol{y}$ is 1. Otherwise, it is 0. The optimizer uses Adam [10].

3.5 Performing

For each entity that appears in the test set, the entity type predicted by the RACE2T model is:

$$\hat{t} = \arg\max_{t \in \mathcal{T}} sigmoid\left(\phi\left(e, t\right)\right) \tag{9}$$

4 Experiments

4.1 Datasets

The datasets used for knowledge graph entity type prediction in this paper are FB15KET [13], and YAGO43KET [13] (the form is: (entity, entity type)), and the corresponding knowledge graph triple datasets are FB15K [2], and YAGO43K [13]. The entity types in FB15KET, and YAGO43KET are mapped to the entities in FB15K, and YAGO43K, respectively. The statistics of datasets are shown in Table 1. FRGAT uses triple datasets to utilize the information about the relation, and CE2T uses entity-entity type tuple datasets to learn the embedding of entity types and perform entity type prediction.

Table 1. Statistics of datasets.

Dataset	$\lvert\mathcal{E}\rvert$	$\lvert\mathcal{T}\rvert$	#Train	#Valid	#Test
FB15KET	14951	3851	136618	15749	15780
YAGO43KET	41723	45182	375853	42739	42750
Dataset	$\lvert\mathcal{E}\rvert$	$\lvert\mathcal{R}\rvert$	#Train	#Valid	#Test
FB15K	14951	1345	483142	–	–
YAGO43K	42335	37	331687	–	–

4.2 Experimental Setup

Evaluation Metrics: Use ranking-based metric for evaluation [2]. First, for each tested entity-entity type tuple, we remove the entity type. Then, the ranking of the entity types predicted by RACE2T is calculated according to Eq. (9). Finally, its exact rank is obtained by the correct entity type. Two metrics are used for evaluation: the mean reciprocal rank (MRR) [2] and the proportion of correct entity types that predict the top k (HITS@k, k = 1, 3, 10) [2].

Model Parameters Setting: The optimal parameters of the RACE2T model are determined by grid search. Specifically: the embedding dimension of entity and relation is adjusted in $\{50, 100\}$, the embedding dimension of entity type is adjusted in $\{100, 200, 300\}$, the dimension of hidden layer of FRGAT is adjusted in $\{200, 300\}$, the output dimension of FRGAT is adjusted in $\{200, 400, 600\}$, the batch size is adjusted in $\{128, 256, 512\}$, the number of layer is adjusted in $\{1, 2, 3\}$, the number of attention head is adjusted in $\{1, 2, 3, 4\}$, the learning rate is adjusted in $\{0.0001, 0.0005, 0.00\ 1, 0.01\}$, the number of convolution kernels is adjusted in $\{10, 32, 64, 128\}$, the size of the convolution kernel is adjusted in $\{1 \times 2, 1 \times 4, 1 \times 8\}$, and the stride size of convolution operation is adjusted in $\{1, 2, 4, 8\}$. We use Xavier to initialize model parameters. Detailed parameter settings can be found on our GitHub[7].

4.3 Entity Type Prediction Experiments

The baselines choose RESCAL [18], RESCAL-ET [13], HOLE [17], HOLE-ET [13], TransE [2], TransE-ET [13], ConvKB [15], CapsE [16], ETE [13], ConnectE (E2T)[8] [31] and ConnectE(E2T+TRT)[9] [31]. The experimental results are shown in Table 2.

From Table 2, we can see that RACE2T outperforms existing baselines on FB15kET and YAGO43kET. We attribute these results to the fact that RACE2T reasonably aggregate the informations of entities and relations in a given entity neighborhood, and that information about entities and relations helps to infer the types to which entities belong. Without using the encoder FRGAT, the decoder CE2T proposed in this paper also achieved good performance, reflecting that the

Table 2. Entity type prediction results. The baseline results are taken from Ref [31].

Dataset	FB15KET				YAGO43KET			
Model	MRR	HITS@1	HITS@3	HITS@10	MRR	HITS@1	HITS@3	HITS@10
RESCAL [18]	0.19	0.0971	0.1958	0.3758	0.08	0.0424	0.0831	0.1531
RESCAL-ET [13]	0.24	0.1217	0.2792	0.5072	0.09	0.0432	0.0962	0.1940
HOLE [17]	0.22	0.1329	0.2335	0.3816	0.16	0.0902	0.1728	0.2925
HOLE-ET [13]	0.42	0.2940	0.4804	0.6673	0.18	0.1028	0.2013	0.3490
TransE [2]	0.45	0.3151	0.5145	0.7393	0.21	0.1263	0.2324	0.3893
TransE-ET [13]	0.46	0.3356	0.5296	0.7116	0.18	0.0919	0.1941	0.3558
ConvKB [15]	0.45	0.3365	0.5180	0.7462	0.19	0.1136	0.2481	0.3897
CapsE [16]	0.46	0.3461	0.5279	0.7320	0.21	0.1263	0.2498	0.3946
ETE [13]	0.50	0.3851	0.5533	0.7193	0.23	0.1373	0.2628	0.4218
ConnectE(E2T) [31]	0.57	0.4554	0.6231	0.7812	0.24	0.1354	0.2620	0.4451
ConnectE(E2T+TRT) [31]	0.59	0.4955	0.6432	0.7992	0.28	0.1601	0.3085	0.4792
CE2T	0.57	0.4681	0.6354	0.7834	0.29	0.2131	0.3225	0.4472
RACE2T	**0.64**	**0.5607**	**0.6884**	**0.8172**	**0.34**	**0.2482**	**0.3762**	**0.5230**

[7] https://github.com/GentlebreezeZ/RACE2T.
[8] ConnectE (E2T) represents that entity type triples are not used.
[9] ConnectE (E2T+TRT) represents that entity type triples are used.

Table 3. 1-1 and 1-N entity type prediction results on FB15KET and YAGO43KET.

1-1								
Dataset	FB15KET				YAGO43KET			
Model	MRR	HITS@1	HITS@3	HITS@10	MRR	HITS@1	HITS@3	HITS@10
ETE	0.57	0.4655	0.6358	0.7596	0.27	0.1958	0.3289	0.4701
ConnectE(E2T)	0.63	0.5396	0.6959	0.8042	0.31	0.1950	0.3382	0.4965
ConnectE(E2T+TRT)	0.64	0.5502	0.7027	0.8146	0.32	0.2062	0.3560	0.5104
CE2T	0.64	0.5436	0.7039	0.8135	0.35	0.2706	0.3876	0.5053
RACE2T	**0.71**	**0.6347**	**0.7548**	**0.8620**	**0.39**	**0.3091**	**0.4322**	**0.5623**
1-N								
Dataset	FB15KET				YAGO43KET			
Model	MRR	HITS@1	HITS@3	HITS@10	MRR	HITS@1	HITS@3	HITS@10
ETE	0.48	0.3652	0.5462	0.6971	0.19	0.1188	0.2129	0.3971
ConnectE(E2T)	0.53	0.4201	0.5982	0.7619	0.22	0.1292	0.2493	0.4185
ConnectE(E2T+TRT)	0.54	0.4521	0.6110	0.7712	0.25	0.1501	0.2802	0.4375
CE2T	0.54	0.4378	0.6078	0.7677	0.27	0.1902	0.3001	0.4267
RACE2T	**0.61**	**0.5312**	**0.6619**	**0.7996**	**0.32**	**0.2298**	**0.3533**	**0.4987**

improvement of model expression ability can improve the performance of entity type prediction.

In the knowledge graph, an entity often has multiple entity types (1-N). As shown in Fig. 1, the type of *Chicago* can be either *City* or *Film*. Since RACE2T aggregates information in entity neighborhoods, especially information about relations, hence RACE2T can be suitable for modeling the 1-N case. To verify that RACE2T can model the 1-N case better, we divided the test sets of FB15KET and YAGO43KET into two parts, one part is 1-1 and the other part is 1-N. Then, using ETE [13], ConnectE [31], CE2T, and RACE2T for entity type prediction, respectively. The experimental results are shown in Table 3.

From Table 3, it is not difficult to see that RACE2T perform better than ETE [13], ConnectE [31] and CE2T on 1-1 and 1-N. Since RACE2T reasonably aggregates entity and relation information in entity neighborhoods, and this information helps RACE2T capture the differences in entities when they correspond to different types. Therefore, RACE2T can model the 1-N case, while ETE, ConnectE, and CE2T cannot do well.

Table 4. Entity type prediction results on FB15KET and YAGO43KET.

Dataset	FB15KET				YAGO43KET			
Model	MRR	HITS@1	HITS@3	HITS@10	MRR	HITS@1	HITS@3	HITS@10
GAT+CE2T	0.6102	0.5151	0.6582	0.7939	0.3257	0.2352	0.3589	0.5051
RACE2T(TansE)	**0.6212**	**0.5291**	**0.6709**	**0.8030**	0.3301	0.2394	0.3651	0.5098
RACE2T(TansH)	0.6196	0.5269	0.6683	0.8020	0.3305	0.2387	0.3647	**0.5122**
RACE2T(TansD)	0.6195	0.5273	0.6687	0.8029	0.3266	0.2360	0.3603	0.5073
RACE2T(DisMult)	0.6202	0.5280	0.6685	0.8020	**0.3308**	**0.2397**	**0.3659**	0.5115

From Tables 2 and 3, we found that: the RACE2T did not perform as well on YAGO43KET relative to CE2T as they did on FB15KET relative to CE2T. This phenomenon may be related to the number of different relations connected to the entities. FB15K used to train RACE2T contains 1345 relations, while YAGO43K contains only 37 relations, which means that RACE2T has less relation information available on YAGO43K, leading to a decrease in RACE2T performance. More detailed experiments about the above statements will be given in Sect. 4.6.

4.4 Attention Calculation Function Analysis

For a triple (e_i, r_k, e_j), its entity and relation embedding are $\boldsymbol{e}_i$, $\boldsymbol{r}_k$ and $\boldsymbol{e}_j$, respectively. We choose the scoring function of the following knowledge graph completion method to calculate the attention between entities:

1) TransE [2]: For $\boldsymbol{e}_i \in \mathbb{R}^l$, $\boldsymbol{r}_k \in \mathbb{R}^l$, $\boldsymbol{e}_j \in \mathbb{R}^l$. Scoring function: $f(e_i, r_k, e_j) = -\|\boldsymbol{e}_i + \boldsymbol{r}_k - \boldsymbol{e}_j\|_{\ell 2}$.
2) TransH [23]: For $\boldsymbol{e}_i \in \mathbb{R}^l$, $\boldsymbol{r}_k \in \mathbb{R}^l$, $\boldsymbol{e}_j \in \mathbb{R}^l$, $\boldsymbol{W}_{r_k} \in \mathbb{R}^l$. Scoring function: $f(e_i, r_k, e_j) = -\|\hat{\boldsymbol{e}}_i + \boldsymbol{r}_k - \hat{\boldsymbol{e}}_j\|_{\ell 2}$, where $\hat{\boldsymbol{e}}_i = \boldsymbol{e}_i - \boldsymbol{W}_{r_k}^{\mathrm{T}} \boldsymbol{e}_i \boldsymbol{W}_{r_k}$, $\hat{\boldsymbol{e}}_j = \boldsymbol{e}_j - \boldsymbol{W}_{r_k}^{\mathrm{T}} \boldsymbol{e}_j \boldsymbol{W}_{r_k}$.
3) TransD [6]: For $\boldsymbol{e}_i \in \mathbb{R}^l$, $\boldsymbol{r}_k \in \mathbb{R}^l$, $\boldsymbol{e}_j \in \mathbb{R}^l$, $\boldsymbol{e}_i^p \in \mathbb{R}^l$, $\boldsymbol{e}_j^p \in \mathbb{R}^l$, $\boldsymbol{r}_k^p \in \mathbb{R}^l$. Scoring function: $f(e_i, r_k, e_j) = -\|\hat{\boldsymbol{e}}_i + \boldsymbol{r}_k - \hat{\boldsymbol{e}}_j\|_{\ell 2}$, where $\hat{\boldsymbol{e}}_i = \boldsymbol{e}_i + \left(\boldsymbol{e}_i^p\right)^{\mathrm{T}} \boldsymbol{e}_i \boldsymbol{r}_k^p$, $\hat{\boldsymbol{e}}_j = \boldsymbol{e}_j + \left(\boldsymbol{e}_j^p\right)^{\mathrm{T}} \boldsymbol{e}_j \boldsymbol{r}_k^p$.
4) DisMult [29]: For $\boldsymbol{e}_i \in \mathbb{R}^l$, $\boldsymbol{r}_k \in \mathbb{R}^l$, $\boldsymbol{e}_j \in \mathbb{R}^l$. Scoring function: $f(e_i, r_k, e_j) = \boldsymbol{e}_i^{\mathrm{T}} \mathrm{diag}(\boldsymbol{r}_k) \boldsymbol{e}_j$.

Take GAT[10] [22] as the comparison object, and CE2T as the decoder. For a more objective comparison, $\psi(\boldsymbol{u}, \boldsymbol{v}) = \boldsymbol{u}$ (the mode used by GAT) is selected for the combination mode of FRGAT entity and relation embedding vectors, which means that the information of relations is not aggregated in aggregating information. The results of entity type prediction are shown in Table 4.

As can be seen from Table 4, the results of entity type prediction for RACE2T improve about 1% over GAT+CE2T for all evaluation metrics. It indicates that the way to calculate the attention between entities using the scoring function of the knowledge graph completion method is slightly better than the traditional way and illustrates the feasibility of FRGAT. At the same time, using the scoring function of the knowledge graph completion method to calculate the attention coefficient will not generate additional space overhead. They directly use the embedding of entities and relations for calculation, such as TransE [2] and DisMult [29].

4.5 Combination Mode Analysis

Inspired by the Ref [2,20,29], we choose the following ways to combine the embedding vectors of entities and relations:

[10] https://github.com/Diego999/pyGAT.

- void: $\psi(\boldsymbol{u}, \boldsymbol{v}) = \boldsymbol{u}$
- sub: $\psi(\boldsymbol{u}, \boldsymbol{v}) = \boldsymbol{u} - \boldsymbol{v}$
- mult: $\psi(\boldsymbol{u}, \boldsymbol{v}) = \boldsymbol{u} \circ \boldsymbol{v}$
- rotate: $\psi(\boldsymbol{u}, \boldsymbol{v}) = [\boldsymbol{u}_1 \circ \boldsymbol{v}_1 - \boldsymbol{u}_2 \circ \boldsymbol{v}_2; \boldsymbol{u}_1 \circ \boldsymbol{v}_2 + \boldsymbol{u}_2 \circ \boldsymbol{v}_1]$

where $\circ$ represents Hadamard product, $\boldsymbol{x}_1$ represents the first half of vector $\boldsymbol{x}$, $\boldsymbol{x}_2$ represents the second half of vector $\boldsymbol{x}$, $[\cdot;\cdot]$ represents the vector connect operation. The experiment is performed on FB15K and FB15KET, and results are shown in Table 5.

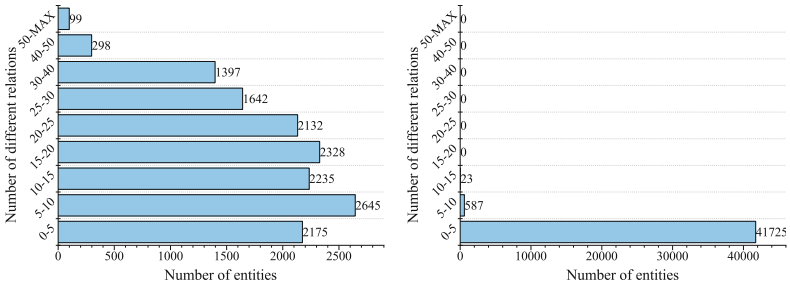
Table 5. The results of entity type prediction on FB15KET.

$f(\cdot,\cdot,\cdot) \rightarrow$	TransE		TransH		TransD		DisMult	
Model ↓	MRR	HITS@1	MRR	HITS@1	MRR	HITS@1	MRR	HITS@1
f+RACE2T(void)	0.6212	0.5291	0.6196	0.5269	0.6195	0.5237	0.6202	0.5280
f+RACE2T(sub)	0.6426	0.5579	0.6402	0.5524	0.6367	0.5484	**0.6456**	**0.5607**
f+RACE2T(mult)	0.6328	0.5448	0.6312	0.5432	0.6304	0.5427	0.6339	0.5451
f+RACE2T(rotate)	0.6412	0.5546	0.6388	0.5547	0.6352	0.5436	0.6427	0.5462

From Table 5, for the RACE2T model, when the combination of entity embedding and relation embedding is: $\psi(\boldsymbol{u}, \boldsymbol{v}) = \boldsymbol{u} - \boldsymbol{v}$, RACE2T achieve the best performance. For different $\psi(\cdot,\cdot)$ except for $\psi(\boldsymbol{u}, \boldsymbol{v}) = \boldsymbol{u}$, the performance gap of RACE2T is not apparent. From the experimental results of entity type prediction in Table 2 (CE2T), Table 4 (GAT+CE2T), and Table 5, we found that using graph networks (for example, GAT) to utilize the neighborhood information of the entity can provide the performance of the model substantial improvement. Meanwhile, from the experimental results of RACE2T (sub/mult/rotate) and RACE2T (void), it can be concluded that integrating the embeddings of relations into the process of aggregating the neighborhood information of the entity can further improve the performance of the model, and it is also verified that the information of relations is helpful to predict the missing entity types of entities.

4.6 the Number of Different Relations Analysis

Figure 4 shows the distribution of the number of different relations connected by entities in the FB15K and YAGO43K datasets. Since the relation distribution in the FB15K dataset is more even than the relation distribution in YAGO43K, this experiment is carried out on FB15K and FB15KET. The experimental results are shown in Table 6.



(a) FB15K (b) YAGO43K

Fig. 4. Distribution of the number of different relations connected by entities.

As shown in Table 6, the performance of RACE2T is always better than that of CE2T, which benefits from the utilization of relations between entities by RACE2T or the information aggregation by RACE2T. Meanwhile, from the results of CE2T and RACE2T in Table 6, we found that more information aggregated is not the better. When an entity connects more different relations, RACE2T aggregates more information about different types of entities and relations in aggregating information. Aggregating more different types of entity and relation information may cause the entity to lose the original information and decrease the performance of RACE2T.

Table 6. The results of entity type prediction on FB15KET.

Distribution	CE2T			RACE2T		
	MRR	HITS@1	HITS@3	MRR	HITS@1	HITS@3
$[0, 5]$	0.4786	0.3836	0.5193	0.5189	0.4241	0.5673
$[5, 10]$	0.5745	0.4777	0.6248	0.6503	0.5691	0.6834
$[10, 15]$	0.5987	0.4912	0.6550	0.6858	0.6025	0.7196
$[15, 20]$	0.5899	0.4753	0.6580	0.6744	0.5909	0.7196
$[20, 25]$	0.5869	0.4703	0.6576	0.6465	0.5525	0.6944
$[25, 30]$	0.5803	0.4671	0.6437	0.6540	0.5624	0.7013
$[30, 40]$	0.5782	0.4746	0.6276	0.6323	0.5447	0.6713
$[40, 50]$	0.5405	0.4437	0.6026	0.5647	0.4794	0.6039
$[50, Max]$	0.4718	0.3596	0.5369	0.5109	0.4137	0.5640

From Table 6, we can observe that when the number of different relations connected by the entity is between 5–40, RACE2T improves about 8% higher than CE2T on MRR, HITS@1, and HITS@3. When the number of different relations connected by the entity is between 0–5, RACE2T improves about 4% higher than CE2T on MRR, HITS@1, and HITS@3. As shown in Fig. 4b, the

number of different relations connected by entities in YAGO43K is mainly concentrated between 0–5, which indirectly leads to RACE2T did not perform as well on YAGO43KET relative to CE2T as they did on FB15KET relative to CE2T.

5 Conclusion and Future Work

This work proposes a method for entity type prediction in knowledge graphs with relational aggregation graph attention network called RACE2T. It includes an encoder and a decoder. The focus of RACE2T is on utilizing relational information when predicting the type of an entity. Therefore, we introduce relational aggregation graph attention network (FRGAT) as an encoder to aggregate the information of entities and relations in the entity neighborhood to utilize the information of relations. Meanwhile, we design a convolutional neural network-based knowledge graph entity type prediction model as the decoder of this paper, called: CE2T. Its role is to measure the similarity between entity embedding and entity type embedding. In addition, we provide various experiments to verify the validity of our model.

Our future research interest is to apply disentangled representation learning (learning disentangled representations of entities) to RACE2T, specifically: using disentangled representation learning for knowledge graph entity type prediction.

Acknowledgements. This work was supported by the National Natural Science Foundation of China (No. 61976032).

References

1. Biswas, R., Sofronova, R., Sack, H., Alam, M.: Cat2type: Wikipedia category embeddings for entity typing in knowledge graphs. In: K-CAP, pp. 81–88 (2021). https://doi.org/10.1145/3460210.3493575
2. Bordes, A., Usunier, N., García-Durán, A., Weston, J., Yakhnenko, O.: Translating embeddings for modeling multi-relational data. In: NIPS, pp. 2787–2795 (2013)
3. Chen, S., Wang, J., Jiang, F., Lin, C.: Improving entity linking by modeling latent entity type information. In: AAAI, pp. 7529–7537 (2020)
4. Dettmers, T., Minervini, P., Stenetorp, P., Riedel, S.: Convolutional 2D knowledge graph embeddings. In: AAAI, pp. 1811–1818 (2018)
5. Jain, N., Kalo, J.-C., Balke, W.-T., Krestel, R.: Do embeddings actually capture knowledge graph semantics? In: Verborgh, R., et al. (eds.) ESWC 2021. LNCS, vol. 12731, pp. 143–159. Springer, Cham (2021). https://doi.org/10.1007/978-3-030-77385-4_9
6. Ji, G., He, S., Xu, L., Liu, K., Zhao, J.: Knowledge graph embedding via dynamic mapping matrix. In: ACL, pp. 687–696 (2015). https://doi.org/10.3115/v1/p15-1067
7. Ji, S., Pan, S., Cambria, E., Marttinen, P., Yu, P.S.: A survey on knowledge graphs: representation, acquisition, and applications. IEEE Trans. Neural Netw. Learn. Syst. 1–21 (2021). https://doi.org/10.1109/TNNLS.2021.3070843

8. Jin, H., Hou, L., Li, J., Dong, T.: Attributed and predictive entity embedding for fine-grained entity typing in knowledge bases. In: Proceedings of the 27th International Conference on Computational Linguistics, pp. 282–292 (2018)
9. Jin, H., Hou, L., Li, J., Dong, T.: Fine-grained entity typing via hierarchical multi graph convolutional networks. In: EMNLP, pp. 4968–4977 (2019). https://doi.org/10.18653/v1/D19-1502
10. Kingma, D.P., Ba, J.: Adam: a method for stochastic optimization. In: ICLR, pp. 1–15 (2015)
11. Kipf, T.N., Welling, M.: Semi-supervised classification with graph convolutional networks. In: ICLR, pp. 1–14 (2017)
12. Moon, C., Harenberg, S., Slankas, J., Samatova, N.F.: Learning contextual embeddings for knowledge graph completion. In: PACIS, pp. 248–253 (2017)
13. Moon, C., Jones, P., Samatova, N.F.: Learning entity type embeddings for knowledge graph completion. In: CIKM, pp. 2215–2218. ACM (2017)
14. Neelakantan, A., Chang, M.: Inferring missing entity type instances for knowledge base completion: new dataset and methods. In: NAACL, pp. 515–525 (2015). https://doi.org/10.3115/v1/n15-1054
15. Nguyen, D.Q., Nguyen, T.D., Nguyen, D.Q., Phung, D.Q.: A novel embedding model for knowledge base completion based on convolutional neural network. In: NAACL, pp. 327–333 (2018). https://doi.org/10.18653/v1/n18-2053
16. Nguyen, D.Q., Vu, T., Nguyen, T.D., Nguyen, D.Q., Phung, D.Q.: A capsule network-based embedding model for knowledge graph completion and search personalization. In: NAACL, pp. 2180–2189 (2019). https://doi.org/10.18653/v1/n19-1226
17. Nickel, M., Rosasco, L., Poggio, T.A.: Holographic embeddings of knowledge graphs. In: AAAI, pp. 1955–1961 (2016)
18. Nickel, M., Tresp, V., Kriegel, H.: A three-way model for collective learning on multi-relational data. In: ICML, pp. 809–816 (2011)
19. Paulheim, H., Bizer, C.: Type inference on noisy RDF data. In: Alani, H., et al. (eds.) ISWC 2013. LNCS, vol. 8218, pp. 510–525. Springer, Heidelberg (2013). https://doi.org/10.1007/978-3-642-41335-3_32
20. Sun, Z., Deng, Z., Nie, J., Tang, J.: Rotate: knowledge graph embedding by relational rotation in complex space. In: ICLR, pp. 1–18 (2019)
21. Vashishth, S., Sanyal, S., Nitin, V., Talukdar, P.P.: Composition-based multi-relational graph convolutional networks. In: ICLR, pp. 1–15 (2020)
22. Velickovic, P., Cucurull, G., Casanova, A., Romero, A., Liò, P., Bengio, Y.: Graph attention networks. In: ICLR, pp. 1–12 (2018)
23. Wang, Z., Zhang, J., Feng, J., Chen, Z.: Knowledge graph embedding by translating on hyperplanes. In: AAAI, pp. 1112–1119 (2014)
24. Xie, R., Liu, Z., Jia, J., Luan, H., Sun, M.: Representation learning of knowledge graphs with entity descriptions. In: AAAI, pp. 2659–2665 (2016)
25. Xie, R., Liu, Z., Sun, M.: Representation learning of knowledge graphs with hierarchical types. In: IJCAI, pp. 2965–2971 (2016)
26. Xu, B., Zhang, Y., Liang, J., Xiao, Y., Hwang, S., Wang, W.: Cross-lingual type inference. In: Navathe, S.B., Wu, W., Shekhar, S., Du, X., Wang, X.S., Xiong, H. (eds.) DASFAA 2016. LNCS, vol. 9642, pp. 447–462. Springer, Cham (2016). https://doi.org/10.1007/978-3-319-32025-0_28
27. Yaghoobzadeh, Y., Adel, H., Schütze, H.: Corpus-level fine-grained entity typing. J. Artif. Intell. Res. **61**, 835–862 (2018). https://doi.org/10.1613/jair.5601

28. Yaghoobzadeh, Y., Schütze, H.: Multi-level representations for fine-grained typing of knowledge base entities. In: EACL, pp. 578–589 (2017). https://doi.org/10.18653/v1/e17-1055
29. Yang, B., Yih, W., He, X., Gao, J., Deng, L.: Embedding entities and relations for learning and inference in knowledge bases. In: ICLR, pp. 1–12 (2015)
30. Zhao, Y., Li, Z., Deng, W., Xie, R., Li, Q.: Learning entity type structured embeddings with trustworthiness on noisy knowledge graphs. Knowl. Based Syst. **215**, 106630 (2021). https://doi.org/10.1016/j.knosys.2020.106630
31. Zhao, Y., Zhang, A., Xie, R., Liu, K., Wang, X.: Connecting embeddings for knowledge graph entity typing. In: ACL, pp. 6419–6428 (2020). https://doi.org/10.18653/v1/2020.acl-main.572
32. Zhu, Q., et al.: Collective multi-type entity alignment between knowledge graphs. In: WWW, pp. 2241–2252 (2020)