Received December 29, 2020, accepted January 21, 2021, date of publication January 29, 2021, date of current version February 5, 2021.

Digital Object Identifier 10.1109/ACCESS.2021.3055529

RAGAT: Relation Aware Graph Attention Network for Knowledge Graph Completion

XIYANG LIU, HUOBIN TAN, QINGHONG CHEN, AND GUANGYAN LIN
School of Software, Beihang University, Beijing 100191, China

Corresponding author: Huobin Tan (thbin@buaa.edu.cn)

This work was supported by the National Key Research and Development Program of China under Grant 2018YFB1402600.

ABSTRACT Knowledge graph completion (KGC) is the task of predicting missing links based on known triples for knowledge graphs. Several recent works suggest that Graph Neural Networks (GNN) that exploit graph structures achieve promising performance on KGC. These models learn information called messages from neighboring entities and relations and then aggregate messages to update central entity representations. The drawback of existing GNN based models lies in that they tend to treat relations equally and learn fixed network parameters, overlooking the distinction of each relational information. In this work, we propose a **R**elation **A**ware **G**raph **AT**tention network (RAGAT) that constructs separate message functions for different relations, which aims at exploiting the heterogeneous characteristics of knowledge graphs. Specifically, we introduce relation specific parameters to augment the expressive capability of message functions, which enables the model to extract relational information in parameter space. To validate the effect of relation aware mechanism, RAGAT is implemented with a variety of relation aware message functions. Experiments show RAGAT outperforms state-of-the-art link prediction baselines on standard FB15k-237 and WN18RR datasets.

INDEX TERMS Knowledge graph completion, knowledge graph embedding, graph attention networks.

I. INTRODUCTION

Since Google Knowledge Graph [1] was proposed in 2012, knowledge graphs (KGs), a.k.a. knowledge bases, have aroused considerable research interest. The structured knowledge called facts in KGs is organized in triples (subject entity, relation, object entity) or short (s, r, o). Some real-world knowledge graphs like Freebase [2], WordNet [3], YAGO [4], DBpedia [5] have been utilized in a wide range of applications, such as question answering [6], recommender systems [7], and dialog systems [8]. However, most KGs suffer from incompleteness [9], which motivates the task of predicting missing links called Knowledge Graph Completion (KGC, also referred to as link prediction).

A mainstream approach for KGC is known to be Knowledge Graph Embedding (KGE) based methods. In general, they embed entities and relations to low-dimensional distributed representations based on existing triples in KGs. Entity embeddings and relation embeddings are obtained by optimizing a scoring function defined on each fact (s,r,o) to measure its plausibility. These models can be broadly classified into three categories: translational distance models [10]–[13], bilinear models [14]–[17], and neural networks based [9], [18], [19]. Translational distance models treat relations as translations from subject entity s to object entity o [10]. Bilinear models utilize product-based score functions to match latent semantics of entity and relation embeddings [14].

Shallow models of translational distance methods and bilinear methods like TransE [10] and RESCAL [14] suffer from learning less expressive features. As noted in [20], one way to alleviate this issue is to increase the size of embeddings, which is impractical to the large scale KGs with numerous entities and relations. To increase the expressiveness of models while keeping the embedding size, neural network architectures have been used to learn knowledge graph embeddings like NTN [18].

Graph Neural Networks (GNN) have recently been applied to obtain knowledge graph embeddings [21], which achieve significant performance improvement. A GNN based model performs as an encoder used to capture graph information, and thereafter, a Convolutional Neural Networks (CNN)

The associate editor coordinating the review of this manuscript and approving it for publication was Chang Choi.

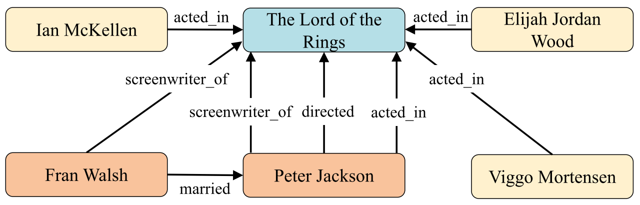

FIGURE 1. A subgraph of a knowledge graph. Different entities are linked to a central entity with different relations.

TABLE 1. Summary of message functions utilized in GNN-based models. Here, e_u, e_r, e_v denotes embeddings of entity *u*, relation *r*, and entity *v*. W_r denotes a separate weight matrix for each relation. α_r is a relation weight scalar. W represents a fixed weight matrix and $W_{dir(r)}$ is defined for different relation directions.

Models	Message Function $\phi(\mathbf{e}_u, \mathbf{e}_r, \mathbf{e}_v)$
R-GCN [21]	$\mathbf{W}_r\mathbf{e}_u$
VR-GCN [24]	$\mathbf{W}[(\mathbf{e}_u - \mathbf{e}_r)\ or\ (\mathbf{e}_u + \mathbf{e}_r)]$
SACN [22]	$\mathbf{W}\alpha_r\mathbf{e}_u$
KBGAT [25]	$\mathbf{W}[\mathbf{e}_u; \mathbf{e}_v; \mathbf{e}_r]$
COMPGCN [26]	$\mathbf{W}_{dir(r)}(\mathbf{e}_v \star \mathbf{e}_r)$

based model is employed as a decoder to predict scores. For instance, SACN [22] takes a Weighted Graph Convolutional Network (W-GCN) as encoder and a convolutional network called Conv-TransE as decoder. Most of these GNN based models can be seen as instantiation of *Message Passing Neural Networks (MPNN)* [23] framework. Message embeddings are constructed from neighboring entity and relation embeddings and then are aggregated to update central entity embeddings.

We notice that neighboring entities under dissimilar relations share distinctive characteristics specific to the relation. Fig. 1 shows a running example of an entity and its neighborhood. Entity *Peter Jackson* is connected to *The Lord of the Rings* with three different relations *directed*, *screenwriter_of* and *acted_in*. Entity *Peter Jackson* affects the central entity variously when combined with particular relations. Such features are shared with entities under the same relations. Entities like {*Peter Jackson*, *Ian McKellen*, *Viggo Mortensen*}, linked to relation *acted_in* are all actors. *Fran Walsh* and *Peter Jackson* are connected to *The Lord of the Rings* with relation *screenwriter_of*, in which they write scripts. The role of actors and screenwriters is different in *The Lord of the Rings*.

Most existing GNN methods learn fixed weight matrices for all entities in graphs, as summarized in Table 1, which makes models fail to capture such relation specific features. For example, VR-GCN [24] obtains messages by applying $\mathbf{W}(\mathbf{e}_u - \mathbf{e}_r)$ or $\mathbf{W}(\mathbf{e}_u + \mathbf{e}_r)$ when entity u takes the role of tail or head regarding relation r. $\mathbf{W}$ is sharing across different locations and local structures, which can filter out various topological structures' common characteristics. Furthermore, $\mathbf{W}$ is fixed for every entity regardless of what type of relations are linked, making the model imperceptive to relations. Though R-GCN [21] defines a separate weight matrix for each relation, which results in R-GCN prone to overfitting, it has no relation representations involved. Besides, R-GCN performs even worse than CNN based models like ConvE for link prediction, as reported in [20].

We propose a new graph neural network, named **R**elation **A**ware **G**raph **AT**tention network (RAGAT), to alleviate the problems mentioned above. The key idea is to construct relation aware message functions. Concretely, beyond weight matrices shared across diverse relations, we define relation specific network parameters to extract relational information from neighboring entities in parameter space. To validate our hypothesis, we leverage relation specific parameters to enhance the expressive power of several existing message functions. Besides, we explore a new way to construct messages based on *interaction embeddings* proposed by CrossE [27]. We further give a new insight of CrossE in an algebraic perspective. RAGAT also follows MPNN framework: (1) RAGAT first combines entity and relation embeddings grouped with identical relation to generate relation-entity hidden embeddings. Thereafter weight matrices shared over the graph are used to transform relation-entity embeddings into messages. (2) Then, we adopt multi-head attention with different learned network parameters to aggregate messages, which allows the model to attend to information from different representation subspaces jointly. After these two phases, the representation for each entity can be updated.

The contributions of our work are summarized as follows:

1) We propose RAGAT, a GAT based method that introduces relation specific network parameters to learn information from neighboring entities under different relations adaptively.
2) We implement RAGAT based on existing message functions integrated with relation specific parameters. Further, we present a new message function to learn interaction embeddings, where an intuitive explanation is provided.
3) Our extensive experiments on link prediction task with benchmark datasets show that RAGAT outperforms state-of-the-art KGE methods and demonstrates the effectiveness of constructing relation aware message functions.

II. RELATED WORK

A. NON-NEURAL

Starting with TransE [10], many shallow non-neural approaches have been proposed, which can be classified as translational distance and bilinear models. TransE [10] regards relations as translations from subject entities to object entities. The embedding of object entity $\mathbf{e}_o$ should be close to the embedding of head entity $\mathbf{e}_s$ plus relation embedding $\mathbf{e}_r$ if (s, r, o) holds. Extensions of TransE have been developed like TransH [12], TransG [11], TransR [13]. Bilinear models exploit similarity-based scoring functions. These models include RESCAL [14], DistMult [15], HolE [16], and Complex [17].

CrossE [27] is a shallow model that learns crossover interactions. It simulates the fact that relation affects the information of entities to be selected, and information of entity affects the relation path to be chosen for inferring whether (s, r, o) is valid.

B. NEURAL NETWORK BASED

Neural Tensor Network (NTN) [18] projects entities to their vector embeddings in the input layer. Then embeddings of head and tail entity are combined by relation specific tensor and given as input to a non-linear layer for computing scores. Multi-Layer Perceptron (MLP) [9] is another simple model where each relation and entity is associated with a single vector and embeddings are concatenated to feed into a non-linear layer. Neural Association Model (NAM) [19] conducts semantic matching with a deep architecture. Such simple neural network based methods suffer from overfitting. Therefore, a vast number of complex neural networks-based models have been proposed, including Convolutional Neural Networks [20], Recurrent Neural Networks [28], and Graph Neural Networks [21].

C. CONVOLUTION BASED

Convolutional Neural Network has been utilized in KGE for its properties of parameter effective and fast to train. ConvE [20] uses convolutional feature filters over matrix reshaped from subject and relation embeddings. InteractE [29] improves the performance of ConvE by feature permutation, checkered reshaping, and circular convolution. More CNN based approaches include ConvKB [30], ConvR [31], CapsE [32].

D. GRAPH NEURAL NETWORK BASED

Graph Neural Networks are exploited in KGE to address the limitations of conventional neural network architectures like CNN that are constrained to handle only Euclidean data [26]. R-GCN [21] introduces Graph Convolutional Networks (GCN) [33] and develops GCN to handle multi-relational data characteristic of realistic knowledge graphs. SACN [22] utilizes W-GCN which learns weights that adapt the amount of information from neighbors. One most recent GNN based model is COMPGCN [26], a framework that generalizes KipfGCN [33], R-GCN [21], D-GCN [34], and W-GCN [22]. COMPGCN takes composition-based GCN as encoder along with ConvE [20] as decoder.

For GCNs, all neighbors share fixed weights and thus contributing equally during information passing. To address this shortcoming, graph attention networks (GAT) [35] are introduced, assigning varying importance levels. KBGAT [25] is the first model to learn graph attention based embeddings on KGs and is claimed to outperform other existing KGE methods. However, there is a bug in the leakage of test triples during negative sampling and the evaluation protocol of KBGAT is not rigorous, as reported in [36].

III. RAGAT: MODEL DESCRIPTION

In this section, we provide details of our model RAGAT. The overall architecture is depicted in Fig. 3. We denote knowledge graph as $\mathcal{G} = (\mathcal{E}, \mathcal{R}, \mathcal{T})$, where $\mathcal{E}$ denotes the set of entities, $\mathcal{R}$ denotes relations, $\mathcal{T}$ is the set of edges. Each edge (s, r, o) represents the relation $r \in \mathcal{R}$ existing from entity s to o. Following previous works [21], [26], we allow information in directed knowledge graphs flow in three directions: original, inverse, and self-loop. Hence, the sets of edges and relations are extended as

$$\mathcal{T}' = \mathcal{T} \cup \left\{s, r, o^{-1} \mid (s, r, o) \in \{\mathcal{T}\}\right\} \cup \{s, \top, s \mid s \in \mathcal{E}\}$$
$$\mathcal{R}' = \mathcal{R} \cup \mathcal{R}_{inv} \cup \{\top\}$$

where $\mathcal{R}_{inv} = \left\{r^{-1} | r \in \mathcal{R}\right\}$ and $\top$ denotes the inverse and self loop relations, respectively.

As summarized in Table 1, we utilize message function to represent how information from neighboring nodes and edges is learned. If entity u is connected to entity v with relation r, then their distributed representations are combined as

$$\mathbf{c}_{(u,r,v)} = \phi(\mathbf{e}_u, \mathbf{e}_r, \mathbf{e}_v) \tag{1}$$

where $\mathbf{e}_u, \mathbf{e}_v \in \mathbb{R}^{d_0}$ denotes entity embeddings and $\mathbf{e}_r \in \mathbb{R}^{d_0}$ is relation embeddings. $\phi(\cdot)$ is a combining operator utilized to incorporate relation embeddings into entity embeddings. Then network parameters $\boldsymbol{\theta}_g$ shared over the whole graph are learned to generate messages:

$$\mathbf{m}_{(u,r,v)} = M(\mathbf{c}_{(u,r,v)}, \boldsymbol{\theta}_g) \tag{2}$$

Here $m_{(u,r,v)}$ is the message from entity u to v. $\boldsymbol{\theta}_g$ is shared across different locations and local structures, which can filter out the common characteristics on various topological structures. When entities are linked with different relations, it plays different roles to the central entity v, as discussed in Section I. Following this intuition, we propose Relation Aware Graph ATtention network, RAGAT for short. Concretely, RAGAT defines relation aware message functions parameterized by relation specific network parameters $\boldsymbol{\theta}_r$. Fig. 2 depicts an illustration of our core idea. To obtain messages, now we have:

$$\mathbf{c}^r_{(u,r,v)} = \phi_r(\mathbf{e}_u, \mathbf{e}_r, \mathbf{e}_v, \boldsymbol{\theta}_r) \tag{3}$$
$$\mathbf{m}_{(u,r,v)} = M(\mathbf{c}^r_{(u,r,v)}, \boldsymbol{\theta}_g) \tag{4}$$

where $\boldsymbol{\theta}_r$ is learned to extract relation specific characteristics. Neighboring entity embeddings and relation embeddings are grouped and fed into varied message functions.

A. RELATION AWARE MESSAGE FUNCTIONS

The most intuitive approach to obtain $\boldsymbol{\theta}_r$ is to learn an independent weight matrix for each relation. Considering experiments are conducted with different encoders and decoders, $\boldsymbol{\theta}_r$ is restricted to be a diagonal matrix for low computation complexity.

$$\boldsymbol{\theta}_r = \mathbf{W}_r = diag(\mathbf{w}_r) \tag{5}$$

where $\mathbf{W}_r \in \mathbb{R}^{d_0 \times d_0}$, $\mathbf{w}_r \in \mathbb{R}^{d_0 \times 1}$. $\boldsymbol{\theta}_r$ can be extended to more parameterized forms as discussed in Section VI. This work shows how $\boldsymbol{\theta}_r$ can improve existing message functions even

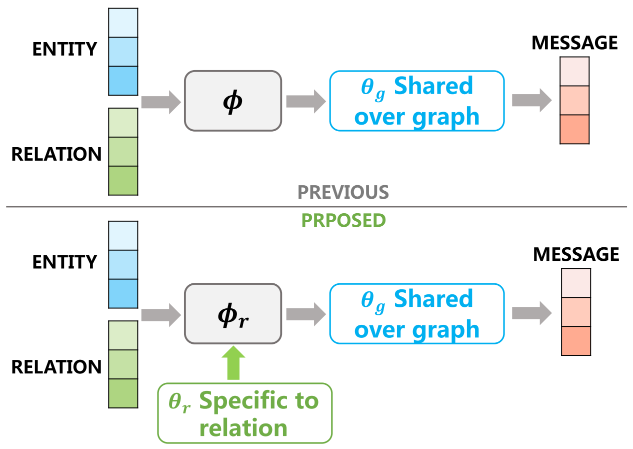
ENTITY
RELATION
ϕ
θ_g Shared over graph
MESSAGE
PREVIOUS
PRPOSED
ENTITY
RELATION
ϕ_r
θ_g Shared over graph
MESSAGE
θ_r Specific to relation

FIGURE 2. The core idea behind RAGAT. θ_r is introduced to reflect relational information in parameter space.

with a simple definition of diagonal matrix. We expect more complex and flexible formulations for $\boldsymbol{\theta}_r$ can be researched to obtain more performance improvements.

Now we present how to apply $\mathbf{W}_r$ into current message functions. To capture the common features associated with specific relation, the transformation operation over entities under the same relation r is performed as following:

$$\mathbf{e}_{(ru)} = \mathbf{W}_r \mathbf{e}_u \tag{6}$$

where $\mathbf{e}_{(ru)}$ represents the transformed hidden embedding of relation r to entity u. Then, to deal with relation embedding $\mathbf{e}_r$, we experiment with several enhanced variants of existing message functions.

1) SUB-GAT

Inspired by TransE, several models learn relational information from entity u with **subtraction** operation like VR-GCN [24], TransGCN [37] defined as:

$$\mathbf{c}_{(u,r,v)} = \mathbf{e}_u - \mathbf{e}_r \tag{7}$$

where operator $-$ is modeling feature interaction between entity embedding and relation embedding. We replace $\mathbf{e}_u$ with $\mathbf{e}_{(ru)}$:

$$\mathbf{c}^r_{(u,r,v)} = \mathbf{W}_r \mathbf{e}_u - \mathbf{e}_r \tag{8}$$

It results in $\mathbf{e}_u$ having multiple representations of features corresponding to relation r. Every entry u_i in $\mathbf{e}_u$, is multiplied by w_i and subtracted with r_i, which is more expressive than previous method.

2) MULT-GAT

Inspired by DistMult, **multiplication** has been experimented in COMPGCN to get messages defined as:

$$\mathbf{c}_{(u,r,v)} = \mathbf{e}_u \circ \mathbf{e}_r \tag{9}$$

$\circ$ denotes Hadamard product. In Mult-GAT, $\mathbf{c}^r_{(u,r,v)}$ is obtained analogously to Sub-GAT:

$$\mathbf{c}^r_{(u,r,v)} = (\mathbf{W}_r \mathbf{e}_u) \circ \mathbf{e}_r \tag{10}$$

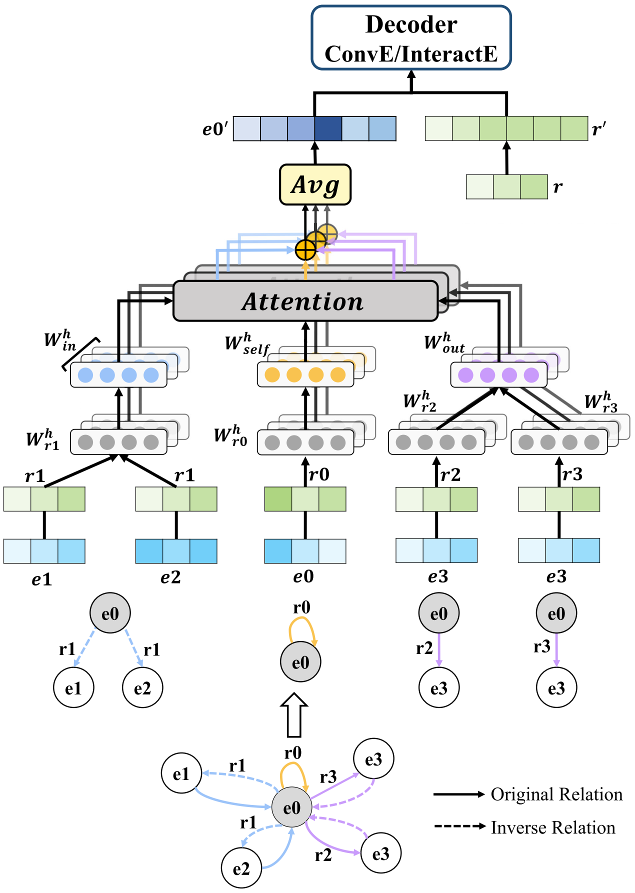

FIGURE 3. Overview of RAGAT with a single graph attention layer along with 3 heads. Updated embeddings of entities and relations are then given into decoder to compute scores.

3) CORR-GAT

COMPGCN finds that the model with **circular correlation** proposed by HolE [16] achieves the best performance for link prediction task. The corresponding enhanced formulation is given as:

$$\mathbf{c}^r_{(u,r,v)} = (\mathbf{W}_r \mathbf{e}_u) \star \mathbf{e}_r \tag{11}$$

Circular correlation can be interpreted as a compression of the tensor product, hence Corr-GAT has a much higher computational complexity than Sub-GAT and Mult-GAT.

4) CONCAT-GAT

A variety of GNNs concatenate entity embeddings and relation embeddings as input and then feed them into neural networks [25], [38], [39]. The number of parameters for model with **concatenation** is twice as many as the aforementioned Sub-GAT, Mult-GAT, and Corr-GAT. No feature interaction between $\mathbf{e}_u$ and $\mathbf{e}_r$ is captured explicitly. We take $[\mathbf{e}_u; \mathbf{e}_r]$ as a whole feature input:

$$\mathbf{c}^r_{(u,r,v)} = \mathbf{W}_r [\mathbf{e}_u; \mathbf{e}_r] \tag{12}$$

5) CROSS-GAT

Further, inspired by model CrossE [27] which aims at simulating *crossover interaction* between entities and relations, we propose a new message function termed **Cross**. CrossE learns multiple triple specific embeddings called *interaction embeddings* which are generated via relation specific variable $\mathbf{c}_r \in \mathbb{R}^{d_0}$. The representation of interaction for entity and relation is written as:

$$\mathbf{u}_I = \mathbf{c}_r \circ \mathbf{e}_u \tag{13}$$

$$\mathbf{r}_I = \mathbf{u}_I \circ \mathbf{e}_r \tag{14}$$

where $\mathbf{u}_I$ is called interaction embedding from relation r to entity s and $\mathbf{r}_I$ is interaction embedding from entity s to relation r. Now, the combined representation of crossover interaction is given as:

$$\mathbf{q}_{ur} = \mathbf{u}_I + \mathbf{r}_I \tag{15}$$

$$= \mathbf{c}_r \circ \mathbf{e}_u + \mathbf{c}_r \circ \mathbf{e}_u \circ \mathbf{e}_r \tag{16}$$

Following this pattern, we replace interaction variable $\mathbf{c}_r$ with $\mathbf{W}_r$, which results in the form:

$$\mathbf{c}^r_{(u,r,v)} = \mathbf{W}_r \mathbf{e}_u + \mathbf{W}_r(\mathbf{e}_u \circ \mathbf{e}_r) \tag{17}$$

Interestingly, this formulation provides us a new explanation of CrossE.

6) AN ALGEBRAIC PERSPECTIVE ON CrossE

One issue about CrossE is that little intuition behind $\mathbf{c}_r$ is explained in original paper and no independent experiments regarding $\mathbf{c}_r$ have been carried out. The distinction between $\mathbf{c}_r$ from $\mathbf{e}_r$ is ambiguous. The score function used in CrossE for calculating the probability of triple (s, r, o) is defined as:

$$f(s, r, o) = \sigma(\mathbf{e}_t^T \, tanh(\mathbf{c}_r \circ \mathbf{e}_s + \mathbf{c}_r \circ \mathbf{e}_s \circ \mathbf{e}_r)) \tag{18}$$

Inspired by 17, we treat $\mathbf{c}_r$ as diagonal matrix $diag(\mathbf{c}_r) \in \mathbb{R}^{d_0 \times d_0}$. Similar to TransR [13], $diag(\mathbf{c}_r)$ projects $\mathbf{e}_s$ and $\mathbf{e}_s \circ \mathbf{e}_r$ to same relation space:

$$\mathbf{v}_s = diag(\mathbf{c}_r)\mathbf{e}_s \tag{19}$$

$$\mathbf{v}_{rs} = diag(\mathbf{c}_r)(\mathbf{e}_s \circ \mathbf{e}_r) \tag{20}$$

$\mathbf{c}_r$ is used to represent relation specific space, and $\mathbf{e}_r$ is performed as a scaling transformation for $\mathbf{e}_s$. The entity-relation combined embedding called crossover interaction in CrossE is obtained by adding two projected vectors $\mathbf{v}_s$ and $\mathbf{v}_{rs}$. Now the combined embedding and target entity embedding $\mathbf{e}_o$ is not in the same space, in which case non-linear function *tanh* is used to ensure the combined representation share the same distribution interval with entity representation. Output score is computed by measuring the similarity between combined embedding and target entity embedding. The process of generating the representation of crossover interaction is illustrated in Fig. 4.

Since original, inverse and self-loop relations are three types of edges with different direction, similar to [26], we define separate filters for each of them:

$$\mathbf{m}_{(u,r,v)} = \mathbf{W}_{dir(r)} \mathbf{c}^r_{(u,r,v)} \tag{21}$$

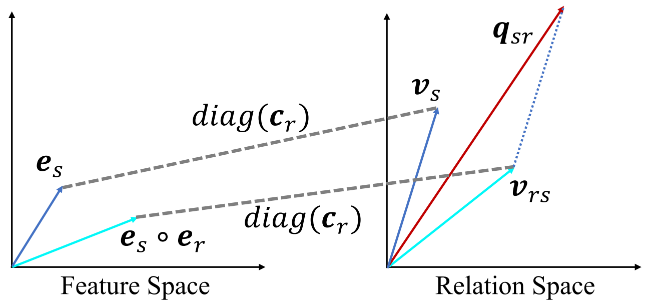

FIGURE 4. The process of generating the representation of crossover interaction applied in CrossE.

where relation-type specific weight $\mathbf{W}_{dir(r)} \in \mathbb{R}^{d_1} \times \mathbb{R}^{d_0}$ is defined as follows:

$$\mathbf{W}_{dir(r)} = \begin{cases} \mathbf{W}_O & \text{if } r \in \mathcal{R} \\ \mathbf{W}_I & \text{if } r \in \mathcal{R}_{inv} \\ \mathbf{W}_S & \text{if } r \in \{\top\} \end{cases} \tag{22}$$

Here, O, I, and S denote original, inverse, and self-loop directions. Thus, RAGAT learns neighboring information in three directions.

B. ATTENTION-BASED INFORMATION AGGREGATION

In order to obtain the new embedding for node v, the update function applied in GCN based models can be written as:

$$\mathbf{e}'_v = f\left(\sum_{(u,r)\in\mathcal{N}(v)} a\mathbf{m}_{(u,r,v)}\right) \tag{23}$$

where $\mathcal{N}(v)$ is the set of immediate neighboring entities and relations of central entity v, f is a non-linear activation function. All messages are summed with fixed importance coefficient a like $|\mathcal{N}(v)|$ applied in R-GCN. It is sensible to assign a varying level of importance to messages. As suggested by GAT [35], the attention mechanism applied in our approach is a single layer feedforward neural network parameterized by weight matrix $\mathbf{W}_{att} \in \mathbb{R}^{1\times d_1}$ and applying LeakyReLU non-linearity.

$$b_{u,r} = LeakyReLU(\mathbf{W}_{att}\mathbf{m}_{(u,r,v)}) \tag{24}$$

$b_{u,r}$ denotes absolute attention coefficient of each message $\mathbf{e}_{u\to v}$. To get relative attention value, softmax is applied over $b_{u,r}$:

$$\alpha_{u,r} = softmax(b_{u,r}) = \frac{exp(b_{u,r})}{\sum_{i\in\mathcal{N}_v}\sum_{r\in\mathcal{R}_{i,u}} exp(b_{i,r})} \tag{25}$$

where $\mathcal{R}_{i,u}$ represents the set of relations connecting entity u and i. Then, normalized attention coefficients are used to compute a linear combination of all messages corresponding to them, to obtain updated output for entities:

$$\mathbf{e}'_v = f\left(\sum_{(u,r)\in\mathcal{N}(v)} \alpha_{u,r}\mathbf{m}_{(u,r,v)}\right) \tag{26}$$

Multi-head attention has been employed in [25], [40] to stabilize the learning process and boost performance. In this work, we adopt multi-head attention to allow the model to attend to information from different relation parameter subspaces jointly. Now, the final message embedding in attention head h is computed as

$$\mathbf{m}^h_{(u,r,v)} = \mathbf{W}^h_{dir(r)}\mathbf{c}^h_r \tag{27}$$

Considering the large size of parameters, we employ *averaging* instead of *concatenation* applied in [35] to M independent attention heads, resulting in the following output representation of entity v:

$$\mathbf{e}'_v = f\left(\frac{1}{H}\sum_{h=1}^{H}\sum_{(u,r)\in\mathcal{N}_v}\alpha^h_{u,r}\mathbf{m}^h_{(u,r,v)}\right) \tag{28}$$

Function f is chosen to be *tanh*.

Further, relation embeddings are also transformed to allow relation embeddings to have the uniform embedding size as $\mathbf{e}'_v$.

$$\mathbf{e}'_r = \mathbf{W}_{rel}\mathbf{e}_r \tag{29}$$

where $\mathbf{W}_{rel} \in \mathbb{R}^{d_1\times d_0}$ is a learnable weight matrix that projects relations to the same embedding space as entities.

C. DECODER

In this work, we utilize two different decoders to validate our model's effectiveness: ConvE [20] and InteractE [29]. ConvE is one of the most commonly used decoders to estimate probabilities for triples. ConvE models the interactions between input entities and relations by convolutional and fully-connected layers. Given (s, r, o) triple, ConvE first reshapes the embedding of s and r into 2D tensors and then applies standard convolution operation on the reshaped tensors to compute triples scores. In ConvE, the triple is scored as:

$$p_{(s,r,o)} = ReLU\left(vec\left(ReLU([\mathbf{e}_s; \mathbf{e}_r] * \boldsymbol{\omega})\right)\mathbf{W}\right)\mathbf{e}_o \tag{30}$$

InteractE augments the expressive power of ConvE through three key ideas: feature permutation, checkered feature reshaping, and circular convolution. For input $(\mathbf{e}_s, \mathbf{e}_r)$, t-random permutations are generated first.

$$\mathcal{P}_t = \left[(\mathbf{e}^1_s, \mathbf{e}^1_r), \ldots, (\mathbf{e}^t_s, \mathbf{e}^t_r)\right] \tag{31}$$

Next, InteractE employs checked reshaping operation $\phi_{chk}(\mathbf{e}_s, \mathbf{e}_r)$, $\forall i \in \{1, \cdots, t\}$.

$$\phi_{chk}(\mathcal{P}_t) = \left[\phi_{chk}(\mathbf{e}^1_s, \mathbf{e}^1_r), \ldots, \phi_{chk}(\mathbf{e}^t_s, \mathbf{e}^t_r)\right] \tag{32}$$

The method of estimating probability for triple (s, r, o) implemented in InteractE can be written formally as:

$$p_{(s,r,o)} = g\left(vec\left(f\left(\phi_{chk}(\mathcal{P}_t) \circledast \boldsymbol{\omega}\right)\right)\mathbf{W}\right)\mathbf{e}_o \tag{33}$$

where $vec(\cdot)$ denotes flattening tensor into vector and $\circledast$ denotes depth-wise circular convolution. $\boldsymbol{\omega}$ represents convolutional filters. $\mathbf{W}$ is a weight matrix. f and g are chosen to be ReLU and sigmoid, respectively.

To train the model, standard cross entropy loss with label smoothing is optimized:

$$\mathcal{L} = -\frac{1}{N}\sum_i (t_i \cdot log(p_i) + (1-t_i)\cdot log(1-p_i)) \tag{34}$$

where t_i is the label of triple i and p_i is the corresponding score.

IV. EXPERIMENTAL SETUP

A. DATASETS

FB15k and WN18 are the two generally used datasets for link prediction introduced by TransE [10]. Nevertheless, previous works [20], [41] suggest that a simple reversal rule-based model can achieve state-of-the-art results on these two datasets. To this end, following [20], we use two corresponding improved datasets: FB15k-237 [41] and WN18RR [20]. The details of these datasets are shown in Table 3.

- **FB15k-237**: A subset of FB15k dataset [10] where all inverse relations are removed to resolve the reversible relation problem.
- **WN18RR**: WN18RR is created from WN18 dataset [10] with deleted relations similar to FB15k-237, a dataset featuring lexical relations between words.

B. IMPLEMENTATION DETAILS

We implement our model using Pytorch [43] with Adam [44] optimizer. Final parameters of RAGAT are determined according to the mean reciprocal rank (MRR) evaluated on validation set. The hyperparameters we find work well are as follows: learning rate 0.001, label smoothing 0.1, 1 layer of GNN, 2 graph attention heads, initial embedding size 100, output embedding size 200, and batch size 1024 for FB15k-237, batch size 256 for WN18RR.

The source code of RAGAT have been made available at https://github.com/liuxiyang641/RAGAT.

C. EVALUATION PROTOCOL

The link prediction evaluation follows the same protocol as previous works [20], in which for each test triple (h, r, t), h and t are replaced by all entities in dataset to calculate scores. Then, following [10], we apply the *filter* setting where valid triples already existing in train, valid, and test set are filtered before ranking. Our evaluation protocol is similar to RANDOM evaluation protocol proposed by [36], which is rigorous and fair for knowledge graph completion task to deal with triples with same scores. Three standard metrics are reported to evaluate performance, mean reciprocal rank (MRR), mean rank (MR), and the proportion of ranking scores within N of all test triples (Hits@N) for N = 1, 3, and 10.

D. BASELINES

To evaluate RAGAT, we compare a variety of non-neural and neural baselines.

- **Non-neural**: Methods that use translation distance based or semantic matching based score functions. For

TABLE 2. Link prediction results of RAGAT and various models on FB15k-237 and WN18RR. We find that RAGAT outperforms all other baselines 4 out of 5 metrics on FB15k-237 and 3 out of 5 metrics on WN18RR. Please refer to Section V-A for more details.

	FB15k-237					WN18RR				
	MRR	**MR**	**Hit@1**	**Hit@3**	**Hit@10**	**MRR**	**MR**	**Hit@1**	**Hit@3**	**Hit@10**
TransE [10]	0.294	357	-	-	0.465	0.226	3384	-	-	0.501
DistMult [15]	0.241	254	0.155	0.263	0.419	0.43	5110	0.39	0.44	0.49
ComplEx [17]	0.247	339	0.158	0.275	0.428	0.44	5261	0.41	0.46	0.51
ConvE [20]	0.325	244	0.237	0.356	0.501	0.43	4187	0.40	0.44	0.52
RotatE [42]	0.338	177	0.241	0.375	0.533	0.476	3340	0.428	0.492	**0.571**
ConvR [31]	0.35	-	0.261	0.385	0.528	0.475	-	0.443	0.489	0.537
R-GCN [21]	0.248	-	-	-	0.417	-	-	-	0.137	-
VR-GCN [24]	0.248	-	0.159	0.272	0.432	-	-	-	-	-
SACN [22]	0.35	-	0.26	0.39	0.54	0.47	-	0.43	0.48	0.54
CrossE [27]	0.299	-	0.211	0.331	0.474	-	-	-	-	-
KBGAT [25]	0.157	270	-	-	0.331	0.412	**1921**	-	-	0.554
InteractE [29]	0.354	**172**	0.263	-	0.535	0.463	5202	0.43	-	0.528
TransE-GCN [37]	0.315	-	0.229	0.324	0.477	0.233	-	0.203	0.338	0.508
G2SKGE [39]	0.342	-	0.253	0.374	0.515	0.447	-	0.424	0.467	0.493
A2N [38]	0.317	-	0.232	0.348	0.486	0.45	-	0.42	0.46	0.51
COMPGCN [26]	0.355	197	0.264	0.39	0.535	0.479	3533	0.443	0.494	0.546
RAGAT	**0.365**	199	**0.273**	**0.401**	**0.547**	**0.489**	2390	**0.452**	**0.503**	0.562

TABLE 3. Datasets statistics of FB15k-237 and WN18RR.

Dataset	$\|\mathcal{E}\|$	$\|\mathcal{R}\|$	Train Set	Valid Set	Test Set
FB15k-237	14,541	237	272,115	17,535	20,466
WN18RR	40,943	11	86,835	3,034	3,134

instance, TransE [10], DistMult [15], ComplEx [17], RotatE [42], CrossE [27].

- **Neural**: Methods that leverage non-linear neural networks like convolutional neural networks and graph neural networks to estimate scores. CNN based method includes ConvE [20], ConvR [31], InteractE [29]. For a fair comparison, we compare against several recent GNN baselines: R-GCN [21], SACN [22], VR-GCN [24], A2N [38], KBGAT [25], G2SKGE [39], TransE-GCN [37] and COMPGCN [26].

V. RESULTS

A. PERFORMANCE COMPARISON

The results of RAGAT compared against existing knowledge graph embedding methods are summarized in Table 2, where the best performance of Cross-GAT with InteractE as decoder is reported. The scores of all baselines reported are directly taken from previous papers [26], [27], [29], [38], [39]. Sun *et al.* [36] investigated the inappropriate evaluation problem that occurred in KBGAT. Hence, we take the results from [36] for KBGAT.

GNN based methods generally achieve better performance than conventional models like TransE and RAGAT improves upon CrossE's MRR by a margin of 22%, Hit@10 by a margin of 15.1% on FB15k-237, which shows the effect of leveraging graph structures. Compared to other baselines, RAGAT outperforms all other methods 4 out of 5 metrics on FB15k-237 and 3 out of 5 metrics on WN18RR, which indicates the whole effectiveness of our model. Compared with KBGAT, A2N, and G2SKGE which also utilize attention to aggregate messages, the improvement of RAGAT demonstrates the performance of our proposed enhanced message functions.

B. COMPARISON OF MESSAGE FUNCTION VARIANTS

Next, we evaluate the effectiveness of RAGAT with its five message construction variants discussed in Section III-A. In our results, $\mathbf{X} + \mathbf{Y}$ denotes that method $\mathbf{X}$ is used as an encoder and $\mathbf{Y}$ is performed as a scoring function. The statistic results are given in Table 4. The results of variants with InteractE as scoring function are generally better than utilizing ConvE to predict scores. InteractE has the same input, training strategy, and output as ConvE, in which it can be easily adapted to previous works like COMPGCN to augments the expressive power.

(**Cross-GAT** + **InteractE**) gives the best performance and (**Concat-GAT** + **InteractE**) as well as (**Corr-GAT** + **InteractE**) achieves sub-optimal results. The results show that more complex combining function $\phi_r(\cdot)$ outperforms or performs comparably to simpler functions. This is consistent with the observations in COMPGCN attributed to the hypothesis that complex operations can provide more expressive power. Besides, comparing the results of **Cross-GAT** with **Concat-GAT** which increases network parameters by just concatenating vector input, we learn that it is reasonable to achieve better performance without introducing new parameters by designing more suitable ways to increase feature interaction between entity and relation embeddings like circular correlation and **Cross**.

C. ANALYSIS OF RELATION SPECIFIC PARAMETERS

To further analyze the impact of learning relation specific parameters, we compare the effect of corresponding RAGAT variants where $\mathbf{W}_r$ is removed.

- $\mathbf{X} + \mathbf{Y}$: RAGAT with relation specific parameter $\mathbf{W}_r$.
- $\mathbf{X} + \mathbf{Y} -$ **w/o** $\mathbf{W}_r$: RAGAT without $\mathbf{W}_r$.

To ensure a fair comparison, the same RAGAT variants are implemented with the same hyperparameters.

TABLE 4. Effect of different ways of message construction and scoring functions evaluated on FB15k-237 dataset. X + Y denotes that method X is used as an encoder and Y is performed as a scoring function. Overall we find that Cross-GAT + InteractE gives the best performance.

Scoring Function (=Y)→	ConvE					InteractE				
Encoders X↓	**MRR**	**MR**	**Hit@1**	**Hit@3**	**Hit@10**	**MRR**	**MR**	**Hit@1**	**Hit@3**	**Hit@10**
Sub-GAT+Y	0.363	163	0.272	0.397	0.545	0.363	163	0.271	0.398	0.546
Mult-GAT+Y	0.362	170	0.271	0.398	0.546	0.363	202	0.272	0.398	0.548
Corr-GAT+Y	0.362	191	0.271	0.395	0.544	0.364	192	0.272	0.400	0.547
Concat-GAT+Y	0.363	161	0.270	0.398	0.546	0.364	194	0.272	0.400	0.547
Cross-GAT+Y	0.363	194	0.271	0.397	0.544	0.365	199	0.273	0.401	0.547

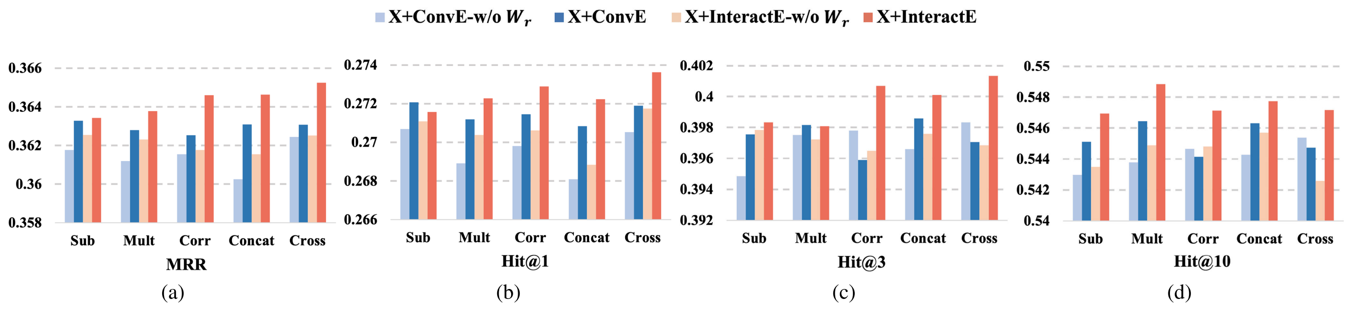

FIGURE 5. The effect of relation specific parameters evaluated on FB15k-237.

TABLE 5. Results on link prediction by relation category on FB15k-237 dataset. Following TransH, relations were classified into four categories based on the average number of tails per head and heads per tail: 1-1, 1-N, N-1, and N-N. We observe that RAGAT is more capable of modeling complex relations. Refer to Section V-D for more details.

		InteractE			RotatE			COMPGCN			RAGAT		
		MRR	MR	H@10	MRR	MR	H@10	MRR	MR	H@10	MRR	MR	H@10
Head Pred	1-1	0.386	**175**	0.547	**0.498**	359	**0.593**	0.457	150	0.604	0.474	197	**0.593**
	1-N	0.106	**573**	0.192	0.092	614	0.174	0.112	604	0.190	**0.131**	772	**0.218**
	N-1	0.466	**69**	0.647	0.471	108	0.674	0.471	99	0.656	**0.478**	89	**0.662**
	N-N	0.276	148	0.476	0.261	**141**	0.476	0.275	179	0.474	**0.287**	156	**0.487**
Tail Pred	1-1	0.368	308	0.547	**0.484**	307	0.578	0.453	193	0.589	0.451	**236**	**0.588**
	1-N	0.777	**27**	0.881	0.749	41	0.674	0.779	34	0.885	**0.790**	**27**	**0.890**
	N-1	0.074	625	0.141	0.074	**578**	0.138	0.076	792	0.151	**0.077**	657	**0.155**
	N-N	0.395	92	0.617	0.364	**90**	0.608	0.395	102	0.616	**0.403**	92	**0.624**

The results are shown in Fig. 5, where MRR and Hit@1, 3, 10 are reported. We can find that our enhanced message functions achieve constant superiority over their corresponding functions without $\mathbf{W}_r$ that is widely used in previous works. This demonstrates the validity of introducing relation specific parameters module into our GNN-based models. Moreover, operations with more computational complexity applied in $\phi(\cdot)$ like **Corr**, **Concat** underperforms simple functions like **Sub** in conventional ways. However, these complex formulations can obtain more performance improvement by learning $\mathbf{W}_r$. It shows that altering conventional complex message functions to be relation aware may improve the expressive capability of models more significantly even though they have more parameters.

D. EVALUATION ON DIFFERENT RELATION CATEGORIES

Further, we analyze the performance of RAGAT on different relation categories of FB15k-237. Following [12], relations are classified into four categories based on the average number of tails per head and heads per tail: one-to-one, one-to-many, many-to-one, and many-to-many. As shown in Table 5, we present the results for different relation types. The results of RotatE, InteractE, and COMPGCN are taken directly from [26], [29], [42]. We notice that both COMPGCN and RAGAT outperform InteractE on all four relation types, which indicates that GNN based encoder helps handle both simple and complex relations. RAGAT is more effective at modeling complex relation categories like one-to-many, many-to-one and many-to-many. However, RotatE captures simple relations like one-to-one better, which can be attributed to the fact that RotatE can infer various relation patterns, including symmetry/antisymmetry, inversion, and composition. The number of edges with relation type of one-to-one is much fewer than other edges with complex relations, explaining why our model's effect is improved.

VI. DISCUSSION

Comparison with R-GCN First, R-GCN has no vectorized relation embedding involved, limiting the model's expansibility. For instance, relation embeddings can represent additional information like semantics in textual words describing relations [18]. Relation embedding also performs a vital role in KGE based multi-relational network alignment [24]. RAGAT explicitly models relation embeddings, which can be applied in more areas. Second, to limit the total number of parameters, R-GCN introduces two separate regularizing strategies: *basis decomposition* and *block-diagonal decomposition*. Basis decomposition is defined as:

$$\mathbf{W}_r = \sum_{b=1}^{B} a_{rb}\mathbf{V}_b, \quad \mathbf{V}_b \in \mathbb{R}^{d_1 \times d_0} \tag{35}$$

In the block-diagonal decomposition, W_r is defined through the sum over a set of low-dimensional matrices:

$$\mathbf{W}_r = diag(\mathbf{Q}_{1r}, \ldots, \mathbf{Q}_{Br}), \quad \mathbf{Q}_{br} \in \mathbb{R}^{(d_1/B)\times(d_0/B)} \tag{36}$$

In implement of RAGAT, relation specific parameter $\boldsymbol{\theta}_r$ can be seen as an instantiation of block-diagonal decomposition where $\mathbf{Q}_{br}$ has dimension 1 with $B = d$. $\boldsymbol{\theta}_r$ can be further modified with basis decomposition formulation to adapt to more complicated knowledge graphs.

Variants of relation specific parameters $\boldsymbol{\theta}_r$ RAGAT restricts $\boldsymbol{\theta}_r$ to be $\mathbf{W}_r$ for training speed. In fact, $\boldsymbol{\theta}_r$ can be extended to more parameterized forms like multi-Layer perceptron, convolution kernel [31], etc. More interestingly, we find our idea associated with two most recent works, ParamE [45] and CoPER [46]. In ParamE, subject entity embeddings, relation embeddings, and object entity embeddings are regarded as the input, parameters, and output of a neural network, respectively. CoPER proposes a *contextual parameter generator (CPG)* component which takes relation embeddings as input and outputs network parameters to be performed over entity embeddings. Comparing to ParamE, we learn relation embeddings and relation specific parameters simultaneously. Compared with CoPER, there is no straight transformation operation between relation embedding and $\boldsymbol{\theta}_r$. This strategy makes our method more generalized and it can be extended with various modifications towards $\boldsymbol{\theta}_r$. $\boldsymbol{\theta}_r$ can be implemented as the same architecture of ParamE-Gate proposed in ParamE, or it can be generated by relation embedding $\mathbf{e}_r$ as in CoPER, which we defer for future work.

VII. CONCLUSION

In this paper, we propose RAGAT, a novel graph attention based knowledge graph embedding method with relation aware message functions that are perceptive to neighboring relations. We analyze how to integrate relation specific parameters in previous message functions. What's more, inspired by CrossE, we explore a new message construction method to learn interaction embeddings. We further provide a theoretical understanding of CrossE, which can be associated with translational distance methods like TransR. Through empirical studies on knowledge graph completion, we demonstrate the effectiveness of the proposed RAGAT on FB15k-237 and WN18RR datasets. It is worth noting that there are several ways in which our model can be extended. In the future, we intend to integrate more well-designed neural networks into the RAGAT model. Besides, we are going to devise a transition structure between relation embeddings and relation specific parameters.

REFERENCES

[1] A. Singhal, "Introducing the knowledge graph: Things, not strings," Official Google Blog, May 2012, pp. 1–8.

[2] K. Bollacker, C. Evans, P. Paritosh, T. Sturge, and J. Taylor, "Freebase: A collaboratively created graph database for structuring human knowledge," in *Proc. ACM SIGMOD Int. Conf. Manage. Data (SIGMOD)*, 2008, pp. 1247–1250.

[3] G. A. Miller, "WordNet: A lexical database for English," *Commun. ACM*, vol. 38, no. 11, pp. 39–41, 1995.

[4] F. M. Suchanek, G. Kasneci, and G. Weikum, "Yago: A core of semantic knowledge," in *Proc. 16th Int. Conf. World Wide Web*, C. L. Williamson, M. E. Zurko, P. F. Patel-Schneider, and P. J. Shenoy, Eds., 2007, pp. 697–706.

[5] S. Auer, C. Bizer, G. Kobilarov, J. Lehmann, R. Cyganiak, and Z. G. Ives, "DBpedia: A nucleus for a Web of open data," in *Proc. 6th Int. Semantic Web Conf.*, in Lecture Notes in Computer Science, vol. 4825, 2007, pp. 722–735.

[6] A. Bordes, S. Chopra, and J. Weston, "Question answering with subgraph embeddings," in *Proc. Conf. Empirical Methods Natural Lang. Process. (EMNLP)*, 2014, pp. 615–620.

[7] F. Zhang, N. J. Yuan, D. Lian, X. Xie, and W.-Y. Ma, "Collaborative knowledge base embedding for recommender systems," in *Proc. 22nd ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining*, Aug. 2016, pp. 353–362.

[8] Y. Ma, P. A. Crook, R. Sarikaya, and E. Fosler-Lussier, "Knowledge graph inference for spoken dialog systems," in *Proc. IEEE Int. Conf. Acoust., Speech Signal Process. (ICASSP)*, Apr. 2015, pp. 5346–5350.

[9] X. Dong, E. Gabrilovich, G. Heitz, W. Horn, N. Lao, K. Murphy, T. Strohmann, S. Sun, and W. Zhang, "Knowledge vault: A Web-scale approach to probabilistic knowledge fusion," in *Proc. 20th ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining*, Aug. 2014, pp. 601–610.

[10] A. Bordes, N. Usunier, A. García-Durán, J. Weston, and O. Yakhnenko, "Translating embeddings for modeling multi-relational data," in *Proc. Adv. Neural Inf. Process. Syst.*, 2013, pp. 2787–2795.

[11] H. Xiao, M. Huang, and X. Zhu, "TransG: A generative model for knowledge graph embedding," in *Proc. 54th Annu. Meeting Assoc. Comput. Linguistics*, vol. 1. Berlin, Germany: The Association for Computer Linguistics, 2016, pp. 2316–2325.

[12] Z. Wang, J. Zhang, J. Feng, and Z. Chen, "Knowledge graph embedding by translating on hyperplanes," in *Proc. 28th Conf. Artif. Intell.*, 2014, pp. 1112–1119.

[13] Y. Lin, Z. Liu, M. Sun, Y. Liu, and X. Zhu, "Learning entity and relation embeddings for knowledge graph completion," in *Proc. 29th Conf. Artif. Intell.*, 2015, pp. 2181–2187.

[14] M. Nickel, V. Tresp, and H. Kriegel, "A three-way model for collective learning on multi-relational data," in *Proc. 28th Int. Conf. Mach. Learn.*, 2011, pp. 809–816.

[15] B. Yang, W. Yih, X. He, J. Gao, and L. Deng, "Embedding entities and relations for learning and inference in knowledge bases," in *Proc. 3rd Int. Conf. Learn. Represent.*, 2015, pp. 1–12.

[16] M. Nickel, L. Rosasco, and T. A. Poggio, "Holographic embeddings of knowledge graphs," in *Proc. 30th Conf. Artif. Intell.*, 2016, pp. 1955–1961.

[17] T. Trouillon, J. Welbl, S. Riedel, E. G. Gaussier, and G. Bouchard, "Complex embeddings for simple link prediction," in *Proc. 33rd Int. Conf. Int. Conf. Mach. Learn. (ICML)*, vol. 48, Jun. 2016, pp. 2071–2080.

[18] R. Socher, D. Chen, C. D. Manning, and A. Y. Ng, "Reasoning with neural tensor networks for knowledge base completion," in *Proc. Adv. Neural Inf. Process. Syst.*, 2013, pp. 926–934.

[19] Q. Liu, H. Jiang, Z. Ling, S. Wei, and Y. Hu, "Probabilistic reasoning via deep learning: Neural association models," *CoRR*, vol. abs/1603.07704, pp. 1–12, Mar. 2016.

[20] T. Dettmers, P. Minervini, P. Stenetorp, and S. Riedel, "Convolutional 2D knowledge graph embeddings," in *Proc. 32nd Conf. Artif. Intell.*, 2018, pp. 1811–1818.

[21] M. S. Schlichtkrull, T. N. Kipf, P. Bloem, R. van den Berg, I. Titov, and M. Welling, "Modeling relational data with graph convolutional networks," in *Proc. Eur. Semantic Web Conf.*, in Lecture Notes in Computer Science, vol. 10843, 2018, pp. 593–607.

[22] C. Shang, Y. Tang, J. Huang, J. Bi, X. He, and B. Zhou, "End-to-end structure-aware convolutional networks for knowledge base completion," in *Proc. 33rd Conf. Artif. Intell.*, 2019, pp. 3060–3067.

[23] J. Gilmer, S. S. Schoenholz, P. F. Riley, O. Vinyals, and G. E. Dahl, "Neural message passing for quantum chemistry," in *Proc. 34th Int. Conf. Mach. Learn.*, vol. 70, 2017, pp. 1263–1272.

[24] R. Ye, X. Li, Y. Fang, H. Zang, and M. Wang, "A vectorized relational graph convolutional network for multi-relational network alignment," in *Proc. 28th Int. Joint Conf. Artif. Intell.*, Aug. 2019, pp. 4135–4141.

[25] D. Nathani, J. Chauhan, C. Sharma, and M. Kaul, "Learning attention-based embeddings for relation prediction in knowledge graphs," in *Proc. 57th Annu. Meeting Assoc. Comput. Linguistics*, 2019, pp. 4710–4723.

[26] S. Vashishth, S. Sanyal, V. Nitin, and P. P. Talukdar, "Composition-based multi-relational graph convolutional networks," in *Proc. 8th Int. Conf. Learn. Represent.*, 2020, pp. 1–15.

[27] W. Zhang, B. Paudel, W. Zhang, A. Bernstein, and H. Chen, "Interaction embeddings for prediction and explanation in knowledge graphs," in *Proc. 12th ACM Int. Conf. Web Search Data Mining*, Jan. 2019, pp. 96–104.

[28] Y. Lin, Z. Liu, H. Luan, M. Sun, S. Rao, and S. Liu, "Modeling relation paths for representation learning of knowledge bases," in *Proc. Conf. Empirical Methods Natural Lang. Process.*, 2015, pp. 705–714.

[29] S. Vashishth, S. Sanyal, V. Nitin, N. Agrawal, and P. P. Talukdar, "InteractE: Improving convolution-based knowledge graph embeddings by increasing feature interactions," in *Proc. 34th Conf. Artif. Intell.*, 2020, pp. 3009–3016.

[30] D. Q. Nguyen, T. D. Nguyen, D. Q. Nguyen, and D. Phung, "A novel embedding model for knowledge base completion based on convolutional neural network," in *Proc. Conf. North Amer. Chapter Assoc. Comput. Linguistics, Human Lang. Technol.*, vol. 2, 2018, pp. 327–333.

[31] X. Jiang, Q. Wang, and B. Wang, "Adaptive convolution for multi-relational learning," in *Proc. Conf. North Amer. Chapter Assoc. Comput. Linguistics, Hum. Lang. Technol.*, 2019, pp. 978–987.

[32] D. Q. Nguyen, T. Vu, T. D. Nguyen, D. Q. Nguyen, and D. Phung, "A capsule network-based embedding model for knowledge graph completion and search personalization," in *Proc. Conf. North Amer. Chapter Assoc. Comput. Linguistics, Hum. Lang. Technol.*, 2019, pp. 2180–2189.

[33] T. N. Kipf and M. Welling, "Semi-supervised classification with graph convolutional networks," in *Proc. 5th Int. Conf. Learn. Represent.*, 2017, pp. 1–14.

[34] D. Marcheggiani and I. Titov, "Encoding sentences with graph convolutional networks for semantic role labeling," in *Proc. Conf. Empirical Methods Natural Lang. Process.*, 2017, pp. 1506–1515.

[35] P. Velickovic, G. Cucurull, A. Casanova, A. Romero, P. Liò, and Y. Bengio, "Graph attention networks," in *Proc. 6th Int. Conf. Learn. Represent.*, 2018, pp. 1–12.

[36] Z. Sun, S. Vashishth, S. Sanyal, P. Talukdar, and Y. Yang, "A re-evaluation of knowledge graph completion methods," in *Proc. 58th Annu. Meeting Assoc. Comput. Linguistics*, 2020, pp. 5516–5522.

[37] L. Cai, B. Yan, G. Mai, K. Janowicz, and R. Zhu, "TransGCN: Coupling transformation assumptions with graph convolutional networks for link prediction," in *Proc. 10th Int. Conf. Knowl. Capture*. New York, NY, USA: Association for Computing Machinery, Sep. 2019, pp. 131–138.

[38] T. Bansal, D.-C. Juan, S. Ravi, and A. McCallum, "A2n: Attending to neighbors for knowledge graph inference," in *Proc. 57th Annu. Meeting Assoc. Comput. Linguistics*. Florence, Italy: Association for Computational Linguistics, Jul. 2019, pp. 4387–4392.

[39] W. Li, X. Zhang, Y. Wang, Z. Yan, and R. Peng, "Graph2Seq: Fusion embedding learning for knowledge graph completion," *IEEE Access*, vol. 7, pp. 157960–157971, 2019.

[40] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin, "Attention is all you need," in *Proc. Adv. Neural Inf. Process. Syst.*, 2017, pp. 5998–6008.

[41] K. Toutanova, D. Chen, P. Pantel, H. Poon, P. Choudhury, and M. Gamon, "Representing text for joint embedding of text and knowledge bases," in *Proc. Conf. Empirical Methods Natural Lang. Process.*, 2015, pp. 1499–1509.

[42] Z. Sun, Z. Deng, J. Nie, and J. Tang, "Rotate: Knowledge graph embedding by relational rotation in complex space," in *Proc. 7th Int. Conf. Learn. Represent.*, 2019, pp. 1–18.

[43] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, and A. Desmaison, "PyTorch: An imperative style, high-performance deep learning library," in *Proc. Adv. Neural Inf. Process. Syst.*, 2019, pp. 8024–8035.

[44] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," in *Proc. 3rd Int. Conf. Learn. Represent.*, 2015, pp. 1–15.

[45] F. Che, D. Zhang, J. Tao, M. Niu, and B. Zhao, "Parame: Regarding neural network parameters as relation embeddings for knowledge graph completion," in *Proc. 34th Conf. Artif. Intell.* Palo Alto, CA, USA: AAAI Press, Feb. 2020, pp. 2774–2781.

[46] G. Stoica, O. Stretcu, E. A. Platanios, T. M. Mitchell, and B. Póczos, "Contextual parameter generation for knowledge graph link prediction," in *Proc. 34th Conf. Artif. Intell.* Palo Alto, CA, USA: AAAI Press, Feb. 2020, pp. 3000–3008.

XIYANG LIU received the B.S. degree from the School of Software, Beihang University, in 2019, where he is currently pursuing the master's degree. His current research interests include knowledge graph completion, knowledge graph embedding, and graph neural network.

HUOBIN TAN received the B.S. degree in computer science from Shenyang Aerospace University, China, in 2000, and the M.S. and Ph.D. degrees in computer science from Beihang University, China, in 2003 and 2015, respectively. He was a Visiting Scholar with Yale University, USA, from 2017 to 2018. He is currently an Associate Professor with the School of Software, Beihang University. His research interests include intelligent software engineering, big data, and recommender systems.

QINGHONG CHEN received the B.S. degree from the School of Educational Information Technology, Central China Normal University, in 2019. He is currently pursuing the master's degree in software engineering with the School of Software, Beihang University. His research interests include knowledge graph embedding and the applications of knowledge graph, especially the knowledge-based recommender systems.

GUANGYAN LIN received the B.S. and M.S. degrees in computer science from Tiangong University, China, in 1989 and 1992, respectively. She is currently an Associate Professor with the School of Software, Beihang University, China. Her research interests include software engineering and agile development.