Global Graph Attention Embedding Network for Relation Prediction in Knowledge Graphs

Qian Li, Daling Wang, Shi Feng, Cheng Niu, and Yifei Zhang

***Abstract*— The incompleteness of knowledge graphs triggers considerable research interest in relation prediction. As the key to predicting relations among entities, many efforts have been devoted to learning the embeddings of entities and relations by incorporating a variety of neighbors' information which includes not only the information from direct outgoing and incoming neighbors but also the ones from the indirect neighbors on the multihop paths. However, previous models usually consider entity paths of limited length or ignore sequential information of the paths. Either simplification will make the model lack a global understanding of knowledge graphs and may result in the loss of important and indispensable information. In this article, we propose a novel global graph attention embedding network (GGAE) for relation prediction by combining global information from both direct neighbors and multihop neighbors. Concretely, given a knowledge graph, we first introduce the path construction algorithms to obtain meaningful paths, then design path modeling methods to capture the potential long-distance sequential information in the multihop paths, final propose an entity graph attention and a relation graph attention mechanisms to obtain entity embeddings and relation embeddings. Moreover, an entity graph attention mechanism is proposed to calculate the entity embeddings by aggregating direct incoming and outgoing neighbors from: 1) an original knowledge graph with the original entity and relation embeddings and 2) a new knowledge graph constructed by the paths whose embeddings are updated by path modeling methods. for each relation, we construct a new graph with related entities and present a relation graph attention to learn the features. Therefore, our model can encapsulate the information from different distance neighbors, and enable the embeddings of entities and relations to better capture all-sided semantic information. The experimental results on benchmark datasets verify the superiority of our model over the state-of-the-art ones.**

***Index Terms*— Graph attention network, knowledge graphs, path embedding, relation prediction.**

I. Introduction

KNOWLEDGE graphs are directed graph-structured knowledge bases with multiple triples like $(\text{entity}_i, \text{relation}_m, \text{entity}_j)$, and have been widely used in information search [1], [2], question answering [3], [4], dialogue system [5], [6], recommender system [7], [8], automatic decision making [9], and visual question reasoning [10]. However, many valid relations are missing in existing knowledge graphs [11], [12], which probably harms the downstream applications. Finding out missing relations, also called relation prediction, has become a trending research topic in academic communities.

As the key to predicting relations among entities, many efforts have been devoted to learning the embedding representations for entities and relations. Many classical models, such as TransE [13], TransH [14], DistMult [15], ComplEx [16], ConvE [17], and ConvKB [18], could learn the embedding of each triple independently. To more effectively model the rich and latent local information from the proximity of each entity, the attention mechanism has been introduced into the knowledge graph embeddings, and assigns different weights to different entities in the vicinity of a given entity [12], [19]. Nevertheless, the information from indirect neighbors on the multihop paths is also essential and indispensable for a better understanding of the entity [19], [20].

Recent work suggests that multihop paths are conducive to obtain contextually missing information for knowledge graph embeddings, such as PTransE [21], RTransE [22], TransE-COMP [23], PRUNED-PATHS [24], RPE [25], RSN [26], KBGAT [19], and OPTransE [27]. As one of the early classical embedding methods of integrating path modeling, PTransE [21] proposes a path-based representation learning model with multihop paths via a path-constraint resource allocation and semantic composition algorithm. Borrowing the paths constructed by the above PTransE algorithm, OPTransE [27] pays more attention to the order features of relations in the paths by projecting the head entity and the tail entity of each relation into different space and captures nonlinear features of different paths with a pooling strategy. KBGAT [19] proposes an attention-based feature embedding network that introduces an auxiliary relation for a multihop neighbor whose embedding is the summation of all the relations in the paths. HAN [20] introduces a hierarchical attention graph neural network to learn node-level and semantic-level information among different paths.

However, the previous path-based embedding models only consider entity paths of limited length or ignore sequential information of the paths. Either simplification may result in the loss of important and indispensable information. Let us focus on the example in Fig. 1, the path ($\textit{Jonathan Nolan} \xrightarrow{\text{brother\_of}} \textit{Christopher Nolan} \xrightarrow{\text{born\_in}} \textit{London} \xrightarrow{\text{capital\_of}} \textit{U.K.}$) is a path with

Manuscript received June 22, 2020; revised January 31, 2021; accepted May 17, 2021. This work was supported in part by the National Key Research and Development Program of China under Grant 2018YFB1004700, in part by the National Natural Science Foundation of China under Grant 61772122 and Grant 61872074, in part by the National Defense Basic Research Program under Grant JCKY2018205C012, and in part by the Fundamental Research Funds for the Central Universities of China under Grant N2016008. *(Corresponding author: Daling Wang.)*

Qian Li, Daling Wang, Shi Feng, and Yifei Zhang are with the College of Computer Science and Engineering, Northeastern University, Shenyang 110169, China (e-mail: wangdaling@cse.neu.edu.cn).

Cheng Niu is with Tencent Company, Beijing 518000, China.

Color versions of one or more figures in this article are available at https://doi.org/10.1109/TNNLS.2021.3083259.

Digital Object Identifier 10.1109/TNNLS.2021.3083259

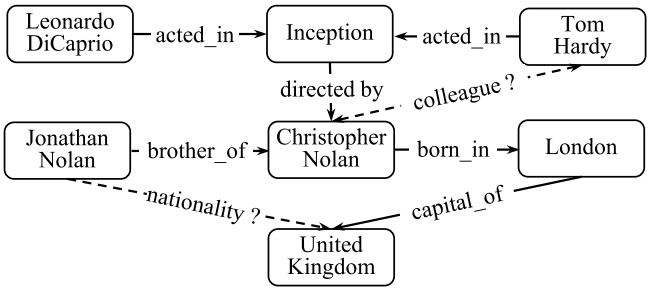


Fig. 1. Subgraph of a knowledge graph, where the rectangles represent entities, the lines represent relations, and the solid lines are actual relations while the dashed lines are inferred relations.

four entities and three relations. By observing closely at the above path, if a person's brother is born in the capital of a nation, then this person should be, with high probability, the citizen of this nation. So, with the help of sequential entity information (*Jonathan Nolan, Christopher Nolan, London, U.K.*) and sequential relation information (brother_of, born_in, capital_of), the model can infer that the *nationality* of *Jonathan Nolan* is *United Kingdom*. If the length of the path above becomes shorter, e.g. the relation path is reduced from (brother_of, born_in, capital_of) to (brother_of, born_in) when only considering 2-hop neighbors, or the order of the path is changed, e.g. the relation path is changed to (brother_of, capital_of, born_in), we will fail to make the inference. Therefore, the model needs to not only incorporate long-distance information from multihop paths but also make use of the path information in a sequential way.

In this article, we propose a novel global graph attention embedding network (GGAE) for relation prediction by combining global information from both direct neighbors and multihop neighbors. The architectures of our proposed model are shown in Fig. 2 (for entity embeddings) and Fig. 3 (for relation embeddings). Concretely, given a knowledge graph $\mathcal{G}$, we first introduce the *path construction* algorithms: *Control paths*, *Random paths*, to obtain meaningful paths, then design *path modeling* methods: *And*, *Or*, *Type*, to capture the potential long-distance sequential information in the multihop paths, finally propose an entity graph attention and a relation graph attention mechanisms to obtain entity embeddings and relation embeddings. Moreover, we propose an entity graph attention mechanism to calculate the entity embeddings by aggregating direct incoming and outgoing neighbors from: 1) an original knowledge graph with original entity and relation embeddings and 2) a new knowledge graph $\mathcal{G}'$ constructed by the paths whose embeddings are updated by *Path Modeling* methods. To obtain the embedding of each relation, we find all related triples, regard each triple as a super entity, and present a relation graph attention to learn the features from the super entities.

Our experimental results on benchmark datasets verify the superiority of our proposed model over the state-of-the-art models. In the experiment section, besides focusing on the overall performance of our model, we conduct a series of comparative experiments on relation properties, path construction methods, path modeling methods, and ablation study.[1] To sum up, the contributions of this article are listed as follows.

[1]https://github.com/feiwangyuzhou/GGAE

TABLE I
MAJOR SYMBOLS

Symbols	Meaning of the Symbols
Control paths	Paths constructed based on the control theory.
Random paths	Paths constructed based on the random theory.
And	Path modeling method to model the entities and relations in the paths together.
Or	Path modeling method to model the entity paths and relation paths separately.
Type	Path modeling method to keep the weights changing along the relation type between each entity pair.
$\mathcal{G}$	A knowledge graph.
$\mathcal{G}'$	A new knowledge graph constructed with the paths from $\mathcal{G}$.
$\mathcal{B}$	A bipartite graph.
$\mathcal{E}$	The set of entities.
$\mathcal{R}$	The set of relations.
$\mathbf{E}$	The initial entity embeddings.
$\mathbf{R}$	The initial relation embeddings.
$\mathbf{E^f}$	The final entity embeddings.
$\mathbf{R^f}$	The final relation embeddings.
e_i	An entity.
r_i	A relation.
h_i	The initial embedding of entity e_i.
s_i	The initial embedding of relation r_i.
$Trip$	The embeddings of triples in $\mathcal{G}'$.
$h_{i,1}^{\leftarrow}$	The incoming embedding of entity e_i in $\mathcal{G}$.
$h_{i,2}^{\leftarrow}$	The incoming embedding of entity e_i in $\mathcal{G}'$.
$h_{i,1}^{\rightarrow}$	The outgoing embedding of entity e_i in $\mathcal{G}$.
$h_{i,2}^{\rightarrow}$	The outgoing embedding of entity e_i in $\mathcal{G}'$.
h_i^f	The final embedding of entity e_i.
s_i^f	The final embedding of relation r_i.
P	$P = \{p_1, p_2, ..., p_N\}$ is the set of paths.
p_i	$p_i = (e_{i,1}, r_{i,1}, ..., r_{i,n-1}, e_{i,n})$ is the ith path in P.
H_i	$H_i = \{x_1, x_2, ..., x_{2n-1}\} = \{h_{i1}, s_{i1}, ..., s_{in-1}, h_{in}\}$ is the initial embeddings of the path p_i.
Y_i	$Y_i = \{y_1, y_2, ..., y_j, ..., y_{2n-1}\}$ is the output embeddings of the path modeling module.

1) We propose a global graph attention embedding (GGAE) network for relation prediction by encapsulating the long-distance sequential information from multihop paths and the potential semantic information from direct neighbors.
2) We introduce two path construction mechanisms to obtain meaningful paths, and three path modeling methods to fully exploit the long-distance sequential information from multihop paths.
3) We present a new relation graph attention mechanism to learn relation embeddings, which could be of great help to future related research.
4) Our experimental results on benchmark datasets verify the superiority of our proposed model over the state-of-the-art models.

The rest of this article is structured as follows. We introduce the related work in Section II. Then, our GGAE model is described in Section III. We report the experimental results in Section IV and the ablation study in Section V. Finally, this article is concluded in Section VI. In order to better understand the rest of this article, we give a table with the major symbols shown in Table I.

II. RELATED WORK

Knowledge graph embeddings devote to mapping entities and relations into vector representation space [12], [13], [15], [16], which set up the foundation for downstream tasks such as relation prediction [28], node classification [20] and question answering [4], [29]. A classical knowledge graph embedding technique represents entities and relations in a continuous

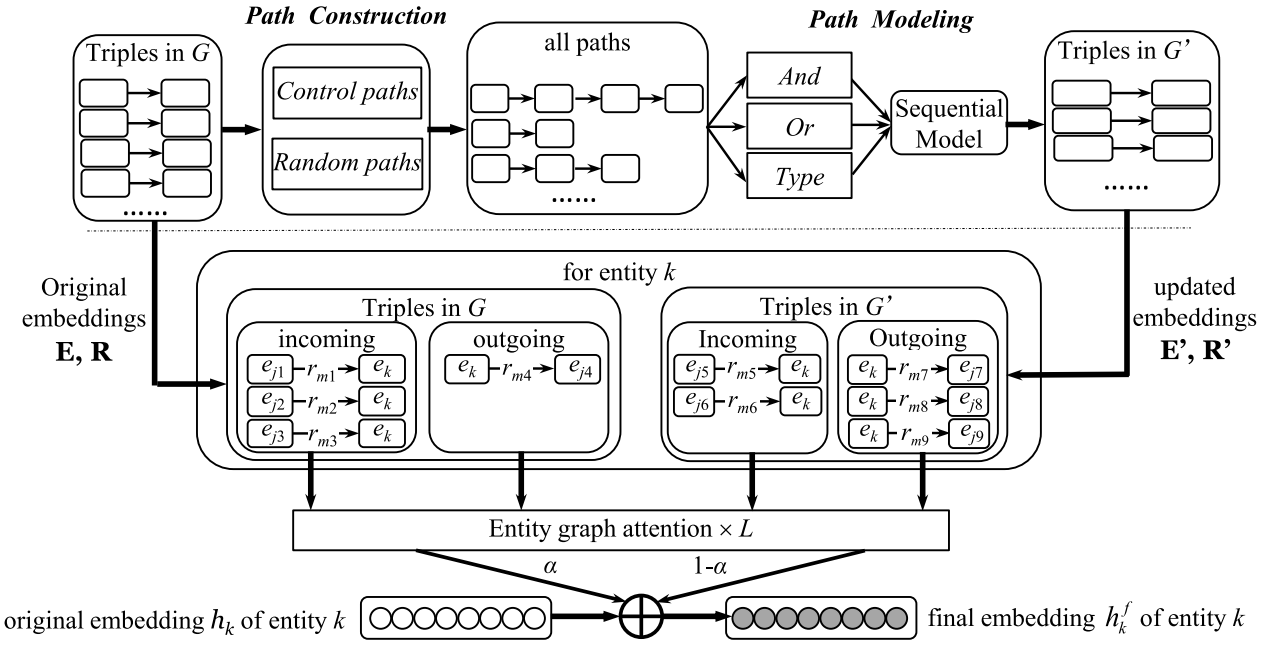


Fig. 2. Architecture of our GGAE model for final entity embeddings. Given a knowledge graph $\mathcal{G}$, we first introduce the *path construction* algorithms: control paths, random paths, to obtain meaningful paths, then design *path modeling* methods: and, or, type, to capture the potential long-distance sequential information in the multihop paths, final propose an entity graph attention to obtain entity embeddings by aggregating direct incoming and outgoing neighbors from: 1) original knowledge graph with original entity and relation embeddings and 2) new knowledge graph $\mathcal{G}'$ constructed by the paths whose embeddings are updated by *Path Modeling* methods.

vector first, and then defines a scoring function to measure the plausibility of each triple, finally learns the representations with some optimization techniques [30].

In some translation-based embedding models, such as TransE [13], TransH [14], TransR [31], TransD [32], DistMult [15], and ComplEx [16], simple linear or bilinear operations are employed to model the relations among embeddings of entities and relations based on the property $h_k + s_m \approx h_j$ for each triple (e_k, r_m, e_j), where h_k, s_m, h_j are embeddings of e_k, r_m, e_j. After that, much work applies the convolutional neural network (CNN) technology to learn nonlinear features to capture complex relationships, such as ConvE [17] and ConvKB [18]. Nevertheless, the aforementioned models learn each triple independently and hence ignore the rich and latent local information from the proximity of each entity in knowledge graphs. To address this issue, the attention mechanism which effectively assigns different weights to different entities in the vicinity of a given entity has been introduced into knowledge graph embeddings [12], [19]. In particular, KBGAT [19] introduces graph attention networks (GATs) [33] into knowledge graph embedding models that incorporate both entity and relation features in any entity's neighborhood.

While much work [12], [19] focuses on direct incoming and outgoing neighbors (local) information for entity and relation embedding, there is a large amount of useful information from indirect neighbors (global) on multihop paths [20], [21], [23], [24]. PTransE [21] proposes a path-based representation learning model with multihop paths via a path-constraint resource allocation and semantic composition algorithm. RTransE [22], TransE-COMP [23], and KBGAT [19] introduce an auxiliary relation for any multihop neighbor whose embedding is the summation of the embeddings of all the relations in the paths. PRUNED-PATHS [24] describes a dynamic incorporation algorithm to efficiently incorporate all paths for knowledge completion. RPE [25] draws inspiration from PTransE and RTransE models to incorporate the path-specific projection between entity pairs in multihop paths. HAN [20] proposes a heterogeneous graph neural network based on hierarchical attention to learn node-level and semantic-level information among different paths for node classification tasks. RSN [26] integrates recurrent neural networks with residual learning to efficiently capture the long-term relational dependencies for knowledge graphs. OPTransE [27] extracts the order features of relations in the paths by projecting the head entity and the tail entity of each relation into different space and captures nonlinear features of different paths with a pooling strategy. And some models transform graph problem into sequential decision problem for some downstream tasks [29], [34]–[36]. Lin *et al.* [29] proposed a reinforcement learning framework for question answering tasks with LSTM to encode search history paths which consist of the sequence of observations and actions.

In comparison, our work focuses on extracting all-sided semantic features for knowledge graph embedding, by leveraging sequential technologies to learn long-distance global information on multihop paths and employing graph attention network to extract local information in the vicinity of a given entity.

III. Proposed GGAE Model

The architectures of our GGAE model are shown in Fig. 2 (for entity embeddings) and Fig. 3 (for relation embeddings). In Fig. 2, given the original triples of a knowledge graph $\mathcal{G}$,

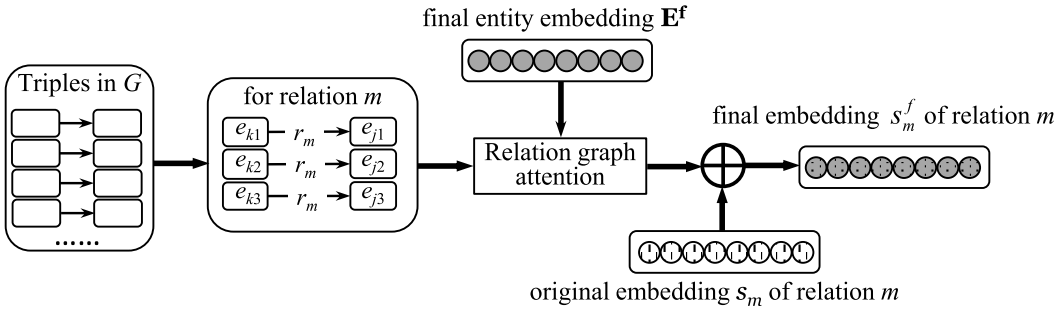


Fig. 3. Architecture of our GGAE model for final relation embeddings. For each relation in a knowledge graph G, we construct a new graph with related entities, and present a relation graph attention to learn the features.

we first apply two *path construction* algorithms to obtain meaningful paths, then design three *path modeling* methods to capture the potential information in the paths. Finally, entity graph attention mechanism is proposed to calculate final entity embeddings by aggregating direct incoming and outgoing neighbors from: 1) an original knowledge graph with the original entity and relation embeddings and 2) a new knowledge graph $\mathcal{G}'$ constructed by the paths whose embeddings are updated by *Path Modeling* methods. In Fig. 3, for each relation, we find all the triples containing the relation and regard each triple as a super entity, then introduce a relation graph attention to learn the features from the super entities. In the following, we first give some definitions, then introduce the path construction algorithms, the path modeling methods, and the embeddings aggregation module, followed by the training procedure.

A. Definition

First, we give some notations and definitions. A knowledge graph $\mathcal{G} = (\mathcal{E}, \mathcal{R})$ is a set of directed triples like (e_k, r_m, e_j), where $e_k, e_j \in \mathcal{E}$ represent entities, $r_m \in \mathcal{R}$ represents the relationship between entity e_k and e_j. Note that there may be multiple relations between two entities. The aim of embedding models is to learn the representation of entities and relations carrying as much key information as possible. The initial entity embeddings and relation embeddings are represented by $\mathbf{E} \in \mathbb{R}^{N_e \times T}$ and $\mathbf{R} \in \mathbb{R}^{N_r \times T}$ respectively, where N_e, N_r are the number of entities and relations, and T is the feature dimension. $h_k \in \mathbf{E}$ is the original embedding of entity e_k and $s_m \in \mathbf{R}$ is the original embedding of relation r_m. With a similar definition, the final entity embeddings and relation embeddings from our model are represented as $\mathbf{E^f} \in \mathbb{R}^{N_e \times T^f}$ and $\mathbf{R^f} \in \mathbb{R}^{N_r \times T^f}$. $h_k^f \in \mathbf{E^f}$ is the final embedding of entity e_k, $s_m^f \in \mathbf{R^f}$ is the final embedding of relation r_m, and T^f is the final feature dimension.

B. Path Construction

In this section, we explore how to obtain meaningful paths. For simplicity, let $P = \{p_1, p_2, \ldots, p_N\}$ represents the set of paths, where N is the number of the paths. Let $p_i = (e_{i,1}, r_{i,1}, e_{i,2}, r_{i,2}, \ldots, r_{i,n-1}, e_{i,n})$ represents the ith path in P, where n denotes the number of entities in path p_i, $1 \leq i \leq N$. For the problem of path redundancy and multiplicity, it is unrealistic and meaningless to find all possible paths. Thus, we study how to obtain meaningful paths that cover entities and relations as much as possible. Two kinds of paths are introduced here: control paths based on control theory, and random paths based on a random strategy.

1) Control Paths: The control paths are constructed based on control theory [37]. A graph network is controllable if it can be driven from an initial state to a desired final state with several countable input signals in finite time [38]. A control path is a path that the signal passes from the input entity to the final entity through a series of connected relations [39]. Liu *et al.* [37] provided an efficient method to find input entities to fully control a network based on the maximum matching algorithm [40]. Therefore, we can use the maximum matching method to find disjoint control paths. Next, we describe the theory of maximum matching.

Let us transfer the original knowledge graph $\mathcal{G} = (\mathcal{E}, \mathcal{R})$ to a bipartite graph $\mathcal{B} = (U, V, \mathcal{R})$, where $U \equiv V \equiv \mathcal{E}$, and every relation in $\mathcal{R}$ is direct which connects a source entity in U to a target entity in V. In a bipartite graph, a matching is a set of relations whose source entities in U share no common entity and whose target entities in V share no common entity too. A maximum matching is matching with the maximum number of relations. An entity in V is called an input entity if it is not connected to any relationship in the maximum matching. Fig. 4(a) shows a maximum matching of a bipartite graph with Fig. 1 as the original knowledge graph, where the gray entities are input entities and the bold relations are matching relations. After capturing the maximum matching, our control path can be constructed by starting from any input entity and propagate along with the matching relations. Fig. 4(b) gives two control paths based on the matching in Fig. 4(a). In particular, a graph may have more than one maximum matching with the same size, where different maximum matchings represent different control schemes. Fig. 4(c) describes another example of the control paths based on a different maximum matching from (b). We ignore the isolated entities without relations when constructing control paths. Noteworthy, an entity in a graph may refer to itself through a path (called a cycle). However, all entities in a cycle are connected to the matching relations. If we want to control a cycle, at least one entity in this cycle should be treated as an input entity.

Based on the above theory, Algorithm 1 gives the construction algorithm of control paths. The inputs of Algorithm 1 are a knowledge graph $\mathcal{G}$ and the number C_N of the maximum matchings. For the algorithm, we first transfer the knowledge

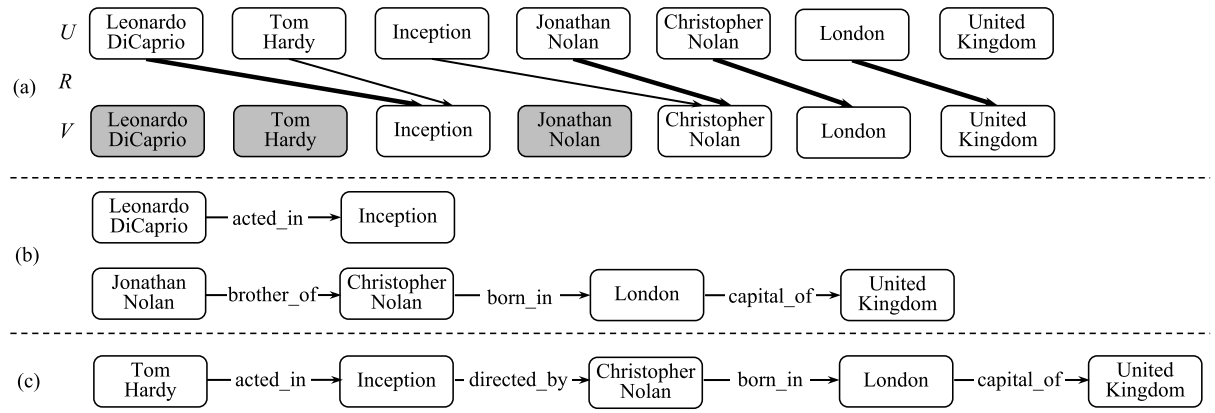


Fig. 4. Examples of control paths with Fig. 1 as the original knowledge graph. (a) Bipartite graph where the gray entities are input entities, the bold relations are matching relations. (b) Two control paths corresponding to the maximum matching in (a). (c) Another example of the control paths whose maximum matching is different from ones in (b).

Algorithm 1 Construction Algorithm of Control Paths

Input: A knowledge graph $\mathcal{G}$, the number C_N of maximum matchings.
Output: A set P of control paths.

Transfer the knowledge graph $\mathcal{G}$ to a bipartite graph $\mathcal{B}$. Let $P = \{\}$, $n = 1$.
for $n <= C_N$ **do**
  M, I = FindMaximumMatching($\mathcal{B}$);
  for each entity $i \in I$ **do**
    Let $T = i$, $p_i = \{i\}$, remove the entity i from I;
    while r, t = FindRelation(M, T) **do**
      Let $p_i = p_i + r + t$, $T = t$;
      Remove the relation r from M;
    end while
    if Length(p_i) > 1 and p_i not in P **then**
      Let $P = P + p_i$.
    end if
  end for
  if M is not None **then**
    for each relation $r \in M$ **do**
      Put the source entity of relation r to I;
    end for
    go Step 4.
  end if
  $n = n + 1$
end for
return P

graph $\mathcal{G}$ to a bipartite graph $\mathcal{B}$ and find a maximum matching with Hopcroft-Karp algorithm [40] to get a set M of matching relations and a set I of input entities. Then we construct a set P of control paths starting from any input entity and propagating along with the matching relations. The relations and entities on each path p_i are stored in order. Next, if there are some relations that remained in the matching set M, the control paths for cycles are constructed starting from any matched entity connected to a remaining relation and propagating along with the remaining matching relations, until no relations in matching set M. The control paths for cycles are also added to the set P. We run the above steps C_N times to extract features from different control schemes, and return the final set P of control paths.

2) Random Paths: The random paths are constructed based on a random strategy. The starting entity of each random path is randomly selected from the entities which are not visited. We use the depth-first search strategy to construct a random path p_i, starting from a random entity and searching along with the relations until the current entity does not have any outgoing relation. Loop through the above step until all entities have been visited. The isolated nodes without relations are ignored similar to the control path construction method. We also run the above steps C_N times to extract features from different random schemes and return the final set P of random paths.

Note that only paths whose length is greater than 1 are preserved. When running the constructing algorithms of control paths or random paths multiple times, duplicate paths may appear which should be discarded.

Intuitively, the model using control paths can perform better than one using random paths. First, we infer that the sequential information on control paths based on control theory is more important and valuable. Second, the control paths will be longer than random paths which facilitates the acquisition of long-distance information, because the control theory ensures that the input nodes are as few as possible so that the control paths are longer to some extent. Third, under the support of control theory, the control paths can pick out more important entities and relations, while the random paths blindly search all entities and relations. Finally, the control paths can cover as many entities and relations as possible than random paths. In the experiment part, we will design some experiments to verify the above intuition.

C. Path Modeling

The path modeling module aims to learn long-distance sequential information of the entities and relations in multihop paths with the control paths or random paths constructed in path construction section. For a path $p_i = (e_{i,1}, r_{i,1}, e_{i,2}, r_{i,2}, \ldots, r_{i,n-1}, e_{i,n})$, let the initial embeddings be $H_i = \{x_1, x_2, \ldots, x_{2n-1}\} = \{h_{i1}, s_{i1}, \ldots, s_{\text{in}-1}, h_{\text{in}}\}$, where $h_{\text{iv}} \in \mathbf{E}, s_{\text{iv}} \in \mathbf{R}$ are the original embeddings of

entity $e_{i,v}$ and relation $r_{i,v}$, $v = 1, 2, \ldots, n$. We denote the output embeddings of the path modeling module as $Y_i = \{y_1, y_2, \ldots, y_j, \ldots, y_{2n-1}\}$, where y_j denotes the output embedding of an entity if j is odd, otherwise one of a relation.

Each path $p_i \in P$ can be regarded as a sequence. We can use sequential (RNN) models, such as LSTM [41] and GRU [42], to learn long-distance dependencies in the sequences (paths). Both the incoming and the outgoing information is important to entity embeddings, so we consider both the forward and backward directions of the paths by utilizing bidirectional RNN (BiRNN) models. In order to fully exploit the sequential information of entities and relations in the paths, we propose three path modeling methods.

1) And: Each path $p_i = (e_{i,1}, r_{i,1}, e_{i,2}, r_{i,2}, \ldots, r_{i,n-1}, e_{i,n})$ is fed directly into RNN models to learn sequential information by treating entities and relations equally. Since the initial embeddings of entities and relations are obtained with TransE, we can assume that both embeddings are in the same feature space. In detail, at each time step t

$$y_{t,+} = \text{RNN}(x_t, y_{t-1}; \theta_+) \tag{1}$$

$$y_{t,-} = \text{RNN}(x_t, y_{t+1}; \theta_-) \tag{2}$$

$$y_t = [y_{t,+}, y_{t,-}] \tag{3}$$

where θ_+, θ_- are the learnable parameters for forward and backward RNN modules, $1 \leqslant t \leqslant 2n-1$. For simplicity, we regard the Formula (1)–(3) as: $Y_i = (\text{Bi})\text{RNN}(H_i)$.

2) Or: Each path $p_i = (e_{i,1}, r_{i,1}, e_{i,2}, r_{i,2}, \ldots, r_{i,n-1}, e_{i,n})$ is divided into an entity path $p_i^e = (e_{i,1}, e_{i,2}, \ldots, e_{i,n})$ and a relation path $p_i^r = (r_{i,1}, r_{i,2}, \ldots, r_{i,n-1})$. The entity path and relation path are fed into RNN models separately. The initial embedding of entity path p_i^e and relation path p_i^r are $H_i^e = \{h_{i1}, \ldots, h_{\text{in}}\}$ and $H_i^r = \{s_{i1}, \ldots, s_{\text{in}-1}\}$. The output embeddings for p_i^e and p_i^r are denoted with $Y_i^e = \{y_1^e, \ldots, y_n^e\}$ and $Y_i^r = \{y_1^r, \ldots, y_{n-1}^r\}$, respectively,

$$Y_i^e = (\text{Bi})\text{RNN}\big(H_i^e\big) \tag{4}$$

$$Y_i^r = (\text{Bi})\text{RNN}\big(H_i^r\big). \tag{5}$$

Here, the output embeddings after the or model of path p_i are: $Y_i = \{y_1, y_2, \ldots, y_{2n-1}\} = \{y_1^e, y_1^r, \ldots, y_{n-1}^e, y_{n-1}^r, y_n^e\}$

3) Type: We also split each path into an entity path and a relation path similar to the ones described in the or method. However, instead of modeling the entity path using one RNN model, we keep the weights for RNN cells changing along the entity paths, according to the relation type between each entity pair. For instance, let us look back at the example we show in Fig. 4. The path (*Jonathan Nolan* $\xrightarrow{\text{brother\_of}}$ *Christopher Nolan* $\xrightarrow{\text{born\_in}}$ *London* $\xrightarrow{\text{capital\_of}}$ *U.K.*) contains three types of relations, so we use three different RNN cells to model the hidden state for entity path (*Jonathan Nolan*, *Christopher Nolan*, *London*, *U.K.*). Concretely, at each time step t for entity path p_i^e

$$y_{t,+} = \text{RNN}(x_t, y_{t-1}; \theta_{r_t,+}) \tag{6}$$

$$y_{t,-} = \text{RNN}(x_t, y_{t+1}; \theta_{r_t,-}) \tag{7}$$

$$y_t = [y_{t,+}, y_{t,-}] \tag{8}$$

where θ_{r_t} represents the learnable RNN cell parameters which correspond to specific relation r_t.

In the type method, the embeddings of relation path p_i^r are not updated. Therefore, the output embeddings after the Relation model of path p_i should be: $Y_i = \{y_1, y_2, \ldots, y_{2n-1}\} = \{y_1^e, s_{i1}, \ldots, y_{n-1}^e, s_{\text{in}-1}, y_n^e\}$.

So far, the embedding of each entity or relation on the paths has carried long-distance sequential information. We construct a new graph $\mathcal{G}' = (\mathcal{E}', \mathcal{R}')$ with the triples in all the paths. For example, the path $p_i = (e_{i,1}, r_{i,1}, e_{i,2}, r_{i,2}, \ldots, r_{i,n-1}, e_{i,n})$ can be divided into $n-1$ triples $\{(e_{i,1}, r_{i,1}, e_{i,2}), (e_{i,2}, r_{i,2}, e_{i,3}), \ldots, (e_{i,n-1}, r_{i,n-1}, e_{i,n})\}$, and the embeddings of the triples are $\{[y_1; y_3; y_2], \ldots, [y_{2n-3}; y_{2n-1}; y_{2n-2}]\}$. Here, we denote the embeddings of triples in graph $\mathcal{G}'$ as Trip. Notably, there might be several paths connecting the same entity or the same relation. In the next section, the embeddings from different paths are aggregated through attention mechanisms.

It is worth noting that the three path modeling methods have their own advantages and disadvantages. From the perspective of segmentation, the and method with undivided paths can capture the potential connection information between entities and relations, while the or method modeling entity paths and relation paths separately could pay more attention to mining sequence information between entities or relations. The length of input for the and method is always longer than one for the or method, which may affect the performance of the algorithm. From the perspective of parameters, the type method designs different weights for each relation type which can fully exploit the potential connection between relation types and entity pairs. However, the number of parameters of the type method increases with the growth of relation types, which will add pressure to the training environment.

D. Embeddings Aggregation

In this section, the entity embeddings are computed by aggregating direct incoming and outgoing neighbors from: 1) original knowledge graph with the original entity and relation embeddings, in order to obtain potential features from direct neighbors and 2) new knowledge graph $\mathcal{G}'$ constructed by the multihop paths whose embeddings are updated by *path modeling* methods, in order to exploit sequential information from multihop neighbors.

In either part, we believe that both the incoming and outgoing neighbors of an entity should be considered when computing its entity embeddings. For example, the entity *Christopher Nolan* in Fig. 1 has two incoming triples (Jonathan Nolan $\xrightarrow{\text{brother\_of}}$ *Christopher Nolan*, *Inception* $\xrightarrow{\text{directed\_by}}$ *Christopher Nolan*) and one outgoing triple (*Christopher Nolan* $\xrightarrow{\text{born\_in}}$ *London*), which all provide important information for a better understanding of the entity *Christopher Nolan*. Therefore, we propose graph attention to encapsulate entity and relation features from incoming and outgoing neighbors.

Now, we first present how the entity embeddings are modeled by considering direct neighbors in the original graph $\mathcal{G}$. For an entity e_k, the embeddings of each incoming triple (e_i, r_m, e_k) and each outgoing triple (e_k, r_n, e_o) associated with e_k should be taken into account. We learn the representations of incoming triples and outgoing triples separately with two

sets of parameters and concatenate them finally. For any triple, the representation is computed by applying linear transformation over the concatenation of the entity and relation features

$$c^{\leftarrow}_{\text{kim}} = W^{\leftarrow}_{c}[h_i][h_k][s_m] \tag{9}$$

$$c^{\rightarrow}_{\text{kon}} = W^{\rightarrow}_{c}[h_k][h_o][s_n] \tag{10}$$

where $c^{\leftarrow}_{\text{kim}}$ is the vector representation of the incoming triple (e_i, r_m, e_k), while $c^{\rightarrow}_{\text{kon}}$ is the vector representation of the outgoing triple (e_k, r_n, e_o). Vectors $h_k, h_i, h_o \in \mathbf{E}$ denote the original embeddings of entities e_k, e_i, e_o, and vectors $s_m, s_n \in \mathbf{R}$ are the original embeddings of relations r_m, r_n. $W^{\leftarrow}_{c}$, $W^{\rightarrow}_{c}$ denote the linear matrix.

Since the modeling of the incoming and outgoing triples are similar, here we only introduce the modeling method of incoming triples in detail. We get attention value with LeakyRelu nonlinearity and softmax functions for each incoming triple

$$b^{\leftarrow}_{\text{kim}} = \text{LeakyRelu}\left(W^{\leftarrow}_{b}, c^{\leftarrow}_{\text{kim}}\right) \tag{11}$$

$$a^{\leftarrow}_{\text{kim}} = \frac{\exp\left(b^{\leftarrow}_{\text{kim}}\right)}{\Sigma_{j\in E^{\leftarrow}_{k}}\Sigma_{u\in R^{\leftarrow}_{\text{jk}}}\exp\left(b^{\leftarrow}_{\text{kju}}\right)} \tag{12}$$

where $W^{\leftarrow}_{b}$ is the linear matrix, $E^{\leftarrow}_{k}$ denotes the incoming entities of entity e_k and $R^{\leftarrow}_{\text{jk}}$ denotes the incoming relations from entity e_j to entity e_k in the original graph $\mathcal{G}$. Notice that $R^{\leftarrow}_{\text{jk}}$ is a set because there are probably multiple types of relations between two entities. The incoming embedding of the entity e_k is the sum of each incoming triple representation weighted by their attention values

$$h^{\leftarrow}_{k,1} = \sigma\left(\sum_{j\in E^{\leftarrow}_{k}}\sum_{u\in R^{\leftarrow}_{\text{jk}}} a^{\leftarrow}_{\text{kju}} c^{\leftarrow}_{\text{kju}}\right). \tag{13}$$

To stabilize the learning process, we calculate L times of the above attention mechanism to obtain L independent embeddings [19], [33] with the idea of multihead attention

$$h^{\leftarrow}_{k,1} = \|^{L}_{l=1}\ \sigma\left(\sum_{j\in E^{\leftarrow}_{k}}\sum_{u\in R_{\text{jk}}} a^{\leftarrow,l}_{\text{kju}} c^{\leftarrow,l}_{\text{kju}}\right) \tag{14}$$

where $\|^{L}_{l=1}$ denotes the concatenation of the L embeddings.

Besides the contribution from direct neighbors, the triples from the multihop paths are also aggregated for entity embedding modeling. A new knowledge graph $\mathcal{G}' = (\mathcal{E}', \mathcal{R}')$ is constructed with the triples in all the multihop paths. And the embeddings of entities and relations in a new graph $\mathcal{G}'$ have been updated by the path modeling module. For entity e_k in the new graph $\mathcal{G}'$, the incoming embedding is the sum of the multiply of the attention weight $a^{p\leftarrow}_{\text{kju}}$ with the vector representation $c^{p\leftarrow}_{\text{kju}}$ of incoming triples

$$h^{\leftarrow}_{k,2} = \|^{L}_{l=1}\ \sigma\left(\sum_{j\in E^{p\leftarrow}_{k}}\sum_{u\in R^{p\leftarrow}_{\text{jk}}} a^{p\leftarrow,l}_{\text{kju}} c^{p\leftarrow,l}_{\text{kju}}\right) \tag{15}$$

where $E^{p\leftarrow}_{k}$ denotes the incoming neighbors of entity e_k and $R^{p\leftarrow}_{\text{jk}}$ is the set of relations from entity e_j to entity e_k in the graph $\mathcal{G}'$. The same multiplication operation in formula (9) and attention operation in formula (11) and (12) are used to calculate $c^{p\leftarrow}_{\text{kju}}$ and $a^{p\leftarrow}_{\text{kju}}$ except for using the embeddings Trip from the path modeling module as the input. Then we calculate the outgoing embeddings $h^{\rightarrow}_{k,1}$ and $h^{\rightarrow}_{k,2}$ with the same method as above.

1) Final Entity Embeddings: Here, we introduce how to obtain the final entity embeddings shown in Fig. 2. A switch gate network is presented to assign different value to direct neighbors and multihop neighbors. The embeddings of direct neighbors are concatnated by the incoming embeddings $h^{\leftarrow}_{k,1}$ and outgoing embeddings $h^{\rightarrow}_{k,1}$, as well as the multihop neighbors

$$\alpha = \sigma\left(W_e\left[h^{\leftarrow}_{k,1}; h^{\rightarrow}_{k,1}\right] + W_p\left[h^{\leftarrow}_{k,2}; h^{\rightarrow}_{k,2}\right] + b\right) \tag{16}$$

where $\alpha \in [0, 1]$ denotes the weight value for direct neighbors, σ is the Sigmoid activation function, W_e, W_p denote the linear matrix, and b is the bias vector.

The learned entity embedding of entity e_k can be obtained by summing the embeddings of direct neighbors and multihop neighbors with the switch gate network and a linear matrix W_h

$$h'_k = W_h\left[\alpha * \left[h^{\leftarrow}_{k,1}; h^{\rightarrow}_{k,1}\right] + (1-\alpha) * \left[h^{\leftarrow}_{k,2}; h^{\rightarrow}_{k,2}\right]\right]. \tag{17}$$

Moreover, the initial entity embeddings are important but lost while learning new embeddings. So we add the initial entity embeddings with a linear matrix W_f and the learned entity embeddings as our final entity embeddings

$$h^{f}_{k} = h'_k + W_f h_k \tag{18}$$

where $h^{f}_{k} \in \mathbf{E^f}$ is the final embeddings of entity e_k, $h_k \in \mathbf{E}$ is the initial entity embeddings,

2) Final Relation Embeddings: Here, we introduce how to obtain the final relation embeddings shown in Fig. 3. Borrowing the translational assumption from TransE [13], for each triple (e_k, r_m, e_j), let $h_k + s_m \approx h_j$, where h_k, s_m, h_j are embeddings of e_k, r_m, e_j. In other words, the embedding s_m of the relation r_m is about equal to the difference between the head entity embedding h_k and the tail entity embedding h_j. However, there are more than one triples related to a relation. How to ensure that all the related triples satisfy the above theorem is particularly worth studying when calculating the embeddings of relations. Therefore, we present a relation attention mechanism to obtain the final relation embeddings from their related entity embeddings.

For each relation r_m, we first to get the set $\Upsilon_m = \{(e_{k1}, r_m, e_{j1}), (e_{k2}, r_m, e_{j2}), \ldots\}$ of triples containing the relation r_m and regard each triple as a super entity, then to calculate the difference value between the head entity embedding and the tail entity embedding in each super entity, final to apply a relation graph attention mechanism to learn the features from super entities

$$\beta(h_{\text{ki}}, h_{\text{ji}}) = \text{softmax}\left(\text{LeakyRelu}\left(W_r, h^{f}_{\text{ji}} - h^{f}_{\text{ki}}\right)\right) \tag{19}$$

$$s'_m = \sum_{(h_{\text{ki}}, h_{\text{ji}})\in\Upsilon_m} \beta(h_{\text{ki}}, h_{\text{ji}})\left(h^{f}_{\text{ji}} - h^{f}_{\text{ki}}\right) \tag{20}$$

where $h^{f}_{\text{ki}}, h^{f}_{\text{ji}} \in \mathbf{E^f}$ are the final embeddings of entity $e_{\text{ki}}, e_{\text{ji}} \in \Upsilon_m$. The final relation embeddings are summed by the

above attention embeddings and the initial relation embeddings with a linear transformation matrix W_s

$$s_m^f = s'_m + W_s s_m \tag{21}$$

where $s_m^f \in \mathbf{R^f}$ is the final relation embeddings of relation r_m, $s_m \in \mathbf{R}$ is the initial relation embeddings.

E. Training Procedure

The training procedure for relation prediction includes an encoder and a decoder. We first train the encoder and then the decoder. The encoder is to encode the information of both entities and relations from direct neighbors and multihop neighbors mentioned above. Our encoder tries to minimize the L1-norm dissimilarity measure $d_{t_{\text{kmj}}} = ||h_k + s_m - h_j||_1$ for a valid triple $t_{\text{kmj}} = (e_k, r_m, e_j)$. Therefore, the training objective of our encoder is as follows:

$$\mathcal{L}_e = \sum_{t_{\text{kmj}} \in S} \sum_{t'_{\text{kmj}} \in S'} \max \left\{ d_{t'_{\text{kmj}}} - d_{t_{\text{kmj}}} + \gamma_e, 0 \right\} \tag{22}$$

where $\gamma_e > 0$ is a margin parameter, S is the valid triples and S' is the invalid triples.

The decoder is to predict the valid scores for the triples according to the global embeddings properties. We use ConvKB as a decoder borrowing the idea from [19]. The score function can be

$$f(t_{\text{kmj}}) = \left(\|_{q=1}^{Q} \text{ReLU}([h_k, s_m, h_j] * \omega_q \right) W_d \tag{23}$$

where Q is the number of filters, ω_q is the dth filter, $*$ denotes a convolution operator, W_d represents a linear transformation matirx. The soft-margin loss of our decoder is as follows, where α is 1 if $t_{\text{kmj}} \in S$ otherwise 0

$$\mathcal{L}_d = \sum_{t_{\text{kmj}} \in \{S \cup S'\}} \log(\exp(f(t_{\text{kmj}}) \cdot \alpha + 1)) + \frac{\gamma_d}{2} ||W_d||_2^2. \tag{24}$$

IV. Experiment

A. Datasets

To evaluate our proposed method, we use three benchmark datasets: FB15k-237 [19], WN18RR [17] and Kinship [29], as shown in Table II. The number of entities in the FB15k-237 and WN18RR datasets is relatively larger than the one in the Kinship dataset, and the number of relations in the FB15k-237 dataset is relatively larger than ones in WN18RR and Kinship datasets. From the mean in-degree and out-degree, the Kinship dataset is denser than the FB15k-237 and WN18RR datasets. We count the average lengths of control paths and random paths, which are located in the ControlP and RandomP column separately. The average lengths of control paths in all datasets are longer than ones in random paths. And the average length of control paths in the Kinship dataset is longer than that in FB15k-237 and WN18RR datasets.

We also analyze how the types of entities and relations are changed when the number of path constructions increases on all the benchmark datasets, the analysis results are shown in Fig. 5. In the FB15k-237 dataset, the proportions of entity types increase and gradually approach 90% in control paths when the number of path constructions increases, while ones approach 65% in random paths. And the proportions of relation types are larger in control paths than in random paths at the start, increase and gradually approach 95% in control paths and random paths when the number of path constructions increases. In the WN18RR dataset, the proportions of entity types in control paths (80%) are always larger than in random paths (50%), and both control paths and random paths can cover all types of relations. In the Kinship dataset, the proportions in all paths increase and gradually approach 100% (for entity types) and 95% (for relation types) when the number of path constructions increases, and the last two graphs show that we can obtain some better results when the number of path constructions is between 4 and 6. As can be seen from all the graphs in Fig. 5, we have two inferences: the first is that the results with control paths in FB15k-237 and Kinship datasets could have better performance than ones in the WN18RR dataset because the proportions of entity and relation types are larger than 90% in the front two datasets. And the second is that the models with control paths could have better results than the models with random paths because the proportions of entity and relation types in control paths are always greater than or equal to ones in random paths.

B. Settings

Here, the experiment settings are introduced in detail. Following [13], we construct two sets of invalid triples (h', r, t) and (h, r, t') for each valid triple (h, r, t), where $h \neq h', t \neq t'$. To ensure robust performance, the same amount of triples are randomly sampled from these two sets. Then we compute a score for each triple and sort them in ascending order to get the rank of any correct triple (h, r, t). The evaluation metrics are: the proportion of valid test triples ranking in the top N predictions (Hits@N), mean reciprocal rank (MRR), and mean rank (MR). A model with better performance should have higher Hits@N, higher MRR, and lower MR. For more practicality, we evaluate all triples in the test data with the filter setting, even if the triple for the test is a fact in the KBs which may affect the final rank and lower the evaluation scores than filtered ones. The original entity embeddings E and relation embeddings R are initialized by TransE [13]. We use Adam optimizer with a 0.001-learning rate to optimize all the parameters for both the encoder and the decoder model. For the parameters of the encoder, the number L of independent attention is set to 2, the dropout probability value is set to 0.3, the LeakyRelu alphas is set to 0.2, and the final dimension is set to 200. For the parameters of the decoder, we set the weight decay value of 0.00001, the epochs of 200, and the negative ratio of 40. Other hyperparameters are shown in Table III.

C. Compared Models

The baseline models for relation prediction we compared here consist of traditional models and path-based embedding models. We use the grid search technique to select the optimal values of hyperparameters for the baseline models. We first give some traditional models without considering paths.

TABLE II
DATASET STATISTICS

Name	Entities	Relations	InDegree	OutDegree	ControlP	RandomP	Training	Validation	Test	Total
FB15K-237	14,541	237	18.4	17.8	6.5	3.0	272,115	17,535	20,466	310,116
WN18RR	40,943	11	2.7	2.2	3.0	2.9	86,835	3034	3134	93,003
Kinship	104	25	82.2	82.2	27.5	5.0	8544	1068	1074	10,686

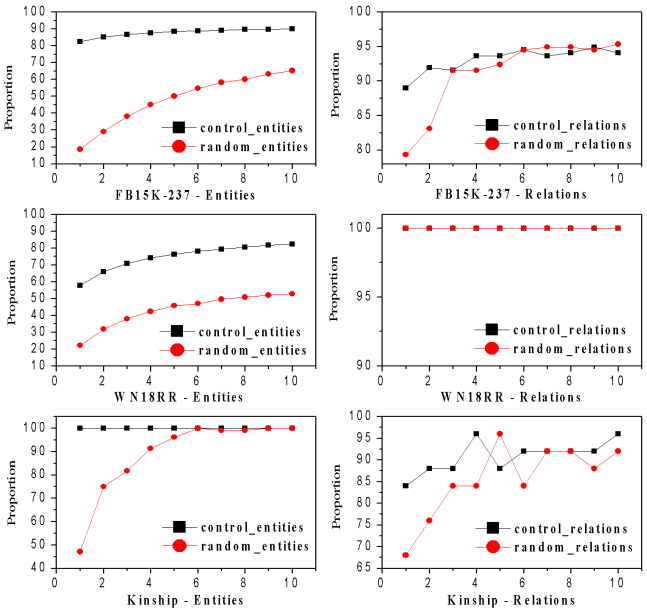


Fig. 5. Proportions of entity or relation types vary with the number of path constructions on benchmark datasets.

TransE [13] is a typical translation-based model to capture the entity and relation embeddings based on the property $h_k + s_m \approx h_j$. The lists of the hyperparameters include: embedding size ∈ {50 100}, learning rate ∈ {1e-4, 5e-4, 1e-3, 5e-3}, margin ∈ {1, 3, 5, 7}, dissimilarity metric ∈ {L1, L2}. After our rigorous experiments with the public code,[2] the optimal values of hyperparameters are 100 for embedding size, 1e-4 for learning rate and L1 for dissimilarity metric on all datasets. The optimal margin is 1 on FB15K-237 and WN18RR datasets while 3 on Kinship dataset.

DistMult [15] is a bilinear diagonal model which uses weighted element-wise dot products to learning embeddings. The lists of the hyperparameters include: embedding size ∈ {50, 100, 200}, learning rate ∈ {0.01, 0.1, 0.5}, ratio of negative triples to positive triples ∈ {1, 2, 5, 10}. After our rigorous experiments with the public code,[3] the optimal values of hyperparameters are 200 for the embedding size and 1 for the ratio of negative triples to positive triples on all datasets. The optimal learning rate is 0.01 on the FB15K-237 dataset, 0.5 on the WN18RR dataset, and 0.1 on the Kinship dataset.

ComplEx [16] is a complex-valued embedding model that can handle a large variety of binary relations, among them symmetric and antisymmetric relations. The lists of the hyperparameters are the same as that of the DistMult model. After our rigorous experiments with the public code,[4] the optimal values of hyperparameters for the ComplEx model are [200, 0.01, 1] on the FB15K-237 dataset, [200, 0.5, 1] on the WN18RR dataset, and [100, 0.1, 1] on Kinship dataset, where [*,*,*] represents the embedding size, learning rate, and ratio of negative triples to positive triples separately.

TABLE III
OPTIMAL VALUES OF HYPER-PARAMETERS FOR OUR MODEL

Data	Encoder				Decoder	
	W-decay	Epochs	Margin	Path-C	Dropout	Filters
FB15K-237	1e-5	3000	1	2	0.3	50
WN18RR	5e-6	3600	5	2	0.0	500
Kinship	5e-6	3600	5	10	0.0	500

ConvE [17] uses 2-D convolution operation to capture global features for each triple. The lists of the hyperparameters include: embedding dropout ∈ {0.0, 0.1, 0.2}, feature map dropout ∈ {0.0, 0.1, 0.2, 0.3}, projection layer dropout ∈ {0.0, 0.1, 0.3, 0.5}, embedding size ∈ {100, 200}, learning rate ∈ {0.001, 0.003}, label smoothing ∈ {0.0, 0.1, 0.2, 0.3}. After our rigorous experiments with the public code,[5] the optimal values of hyperparameters are [0.2, 0.3, 0.2, 200, 0.001, 0.1] on FB15K-237 dataset, [0.2, 0.3, 0.2, 200, 0.003, 0.1] on WN18RR dataset, and [0.0, 0.3, 0.3, 200, 0.003, 0.2] on Kinship dataset.

ConvKB [18] applies the convolutional neural network technology to learn nonlinear features. The lists of the hyperparameters include: embedding size ∈ {50, 100, 200}, learning rate ∈ {5e-6, 1e-5, 5e-5, 1e-4, 5e-4}, the number of filters ∈ {50, 100, 200, 400, 500}. After our rigorous experiments with the public code,[6] the optimal values of hyperparameters are [100, 5e-6, 50] on FB15K-237 dataset, [50, 1e-4, 500] on WN18RR dataset, and [200, 1e-4, 500] on Kinship dataset.

In addition to the above traditional models, we also introduce some latest path-based embedding models as follows.

PTransE [21] is a path-based representation learning model with multihop paths via a path-constraint resource allocation and semantic composition algorithm, where ADD, MUL and RNN are three ways of learning path information. The lists of the hyperparameters include: embedding size ∈ {50, 100}, learning rate ∈ {1e-4, 5e-4, 1e-3, 5e-3}, margin ∈ {1, 3, 5, 7}, dissimilarity metric ∈ {L1, L2}. After our rigorous experiments with the public code,[7] the optimal values of hyperparameters for PTransE(ADD) model are [100, 1e-4, 1, L1] on FB15K-237 dataset, [100, 1e-4, 5, L1] on WN18RR dataset, and [100, 5e-3, 5, L1] on Kinship dataset. The optimal values of hyperparameters for PTransE(MUL) model are [50, 1e-4, 5, L1] on FB15K-237 dataset, [50, 5e-4, 5, L1] on WN18RR dataset, and [50, 1e-4, 3, L1] on Kinship dataset. The optimal values of hyperparameters for PTransE(RNN)

[2]https://github.com/thunlp/KB2E
[3]https://github.com/thunlp/OpenKE
[4]https://github.com/thunlp/OpenKE
[5]https://github.com/TimDettmers/ConvE
[6]https://github.com/daiquocnguyen/ConvKB
[7]https://github.com/thunlp/KB2E

TABLE IV

EXPERIMENTAL RESULTS ON THE FB15K-237 AND WN18RR TEST SETS. THE RESULTS IN THE FIRST GROUP ARE FOR THE TRADITIONAL MODELS, IN THE MIDDLE GROUP ARE FOR THE MODELS WITH PATHS, AND THE LAST GROUP INCLUDES OUR MODELS

Model	FB15K-237					WN18RR				
	$Hits@10\uparrow$	$Hits@3\uparrow$	$Hits@1\uparrow$	$MRR\uparrow$	$MR\downarrow$	$Hits@10\uparrow$	$Hits@3\uparrow$	$Hits@1\uparrow$	$MRR\uparrow$	$MR\downarrow$
TransE	42.6	29.5	19.1	27.0	339	42.8	37.0	2.23	20.0	6175
DistMult	39.4	25.9	16.2	23.9	352	46.8	36.6	23.6	31.7	3915
ComplEx	44.1	29.8	18.7	27.1	395	47.2	42.2	33.3	38.8	4790
ConvE	49.4	36.7	28.8	30.5	260	49.6	44.5	**41.5**	**42.4**	4917
ConvKB	52.7	42.0	34.9	40.7	246	53.1	42.1	5.84	25.4	2794
PTransE(ADD)	28.5	22.7	17.4	21.2	200	21.9	20.2	16.6	18.7	2824
PTransE(MUL)	31.8	27.0	21.3	24.9	152	24.3	22.6	20.9	22.1	2496
PTransE(RNN)	29.8	27.2	22.4	25.2	196	20.8	19.8	19.1	19.6	3746
RSN	45.2	31.6	20.3	28.1	289	41.3	30.2	25.6	35.1	5137
OPTransE	53.0	37.7	25.0	34.2	265	38.7	36.9	34.9	36.4	5944
KBGAT	58.9	47.6	37.7	44.8	214	54.0	44.8	33.0	40.5	2563
GGAE(*Random*)	61.0	49.2	38.5	46.1	188	54.1	44.6	33.1	40.6	2626
GGAE(*Control*)	**61.4**	**50.0**	**39.9**	**47.1**	**183**	**54.5**	**45.3**	33.4	40.9	**2417**

TABLE V

EXPERIMENTAL RESULTS ON THE KINSHIP TEST SET

Model	Kinship				
	$Hits@10\uparrow$	$Hits@3\uparrow$	$Hits@1\uparrow$	$MRR\uparrow$	$MR\downarrow$
TransE	56.3	31.5	11.5	26.4	16.05
DistMult	88.5	61.2	39.9	54.9	4.62
ComplEx	97.0	91.3	74.2	83.3	2.02
ConvE	96.6	89.0	71.7	81.2	2.52
ConvKB	96.3	78.5	48.0	65.2	3.02
PTransE(ADD)	27.3	16.0	7.50	14.1	9.74
PTransE(MUL)	23.9	13.6	5.14	11.5	11.09
PTransE(RNN)	27.3	15.1	6.15	13.1	9.35
RSN	49.2	30.1	15.6	21.9	13.76
OPTransE	58.3	32.8	17.5	30.6	12.87
KBGAT	97.1	91.1	81.8	87.2	2.35
GGAE(*Random*)	97.7	92.4	85.4	89.6	1.97
GGAE(*Control*)	**98.2**	**95.1**	**90.6**	**93.2**	**1.88**

model are [100, 1e-4, 1, L1] on FB15K-237 dataset, [100, 1e-4, 1, L1] on WN18RR dataset, and [50, 5e-4, 3, L1] on Kinship dataset.

RSN [26] is a recurrent skipping network that integrates recurrent neural networks with residual learning to efficiently capture the long-term relational dependencies for knowledge graphs. The lists of the hyperparameters include: learning rate $\in$ {1e-4, 5e-4, 1e-3}, bias value $\in$ {0.5, 0.7}, path lengths $\in$ {5, 7, 10}. After our rigorous experiments with the public code,[8] the optimal value of learning rate are 1e-4 on all datasets. The optimal bias value is 0.5 on Kinship dataset and 0.7 on other datasets. The optimal path lengths is 7 on FB15K-237 dataset, 5 on WN18RR dataset and 10 on Kinship dataset.

OPTransE [27] extracts the order features of relations in the paths by projecting the head entity and the tail entity of each relation into different space, and captures nonlinear features of different paths with a pooling strategy. The lists of the hyperparameters include: embedding size $\in$ {50, 100}, learning rate $\in$ {1e-4, 5e-4, 1e-3, 5e-3}, margin $\gamma \in$ {1, 3, 5, 7}, $\gamma 1 \in$ {1,3,5,7}, $\gamma 2 \in$ {1,3,5,7}. After our rigorous experiments with the public code,[9] the optimal values of hyperparameters are 100 for embedding size and 1e-4 for learning rate. The optimal values of margin are $\gamma = 4.0$, $\gamma 1 = 4.5, \gamma 2 = 5.0$ on FB15K-237 dataset, $\gamma = 4.0$, $\gamma 1 = 5.0, \gamma 2 = 5.5$ on WN18RR dataset, and $\gamma = 5.0$, $\gamma 1 = 5.0$, $\gamma 2 = 5.5$ on Kinship dataset.

KBGAT [19] is an attention-based model to learn entity embeddings from incoming neighbors, and introduces an auxiliary relation for each two-hop neighbor whose embedding is the summation of all the relations in multihop paths. And the KBGAT model does not update the relation embeddings. The hyperparameters include the following two parts. For the encoder, the list of the hyperparameters include: weight decay value $\in$ {1e-5, 5e-5, 5e-6}, negative ratio $\in$ {1, 2, 4}, learning rate $\in$ {1e-4, 1e-3}, dropout probability value $\in$ {0.0, 0.3}, LeakyRelu alphas $\in$ {0.2, 0.4}, multiattention heads $\in$ {1, 2, 4}, embedding size $\in$ {200, 400}, margin $\in$ {1, 3, 5}. For the decoder, the list of the hyperparameters include: the number of filters $\in$ {50, 300, 500}, dropout probability value $\in$ {0.0, 0.3}, weight decay value $\in$ {1e-5, 5e-5}, negative ratio $\in$ {10, 40, 80}, learning rate $\in$ {1e-4, 1e-3}. After our rigorous experiments with the public code,[10] the optimal hyperparameter values for the encoder are [1e-5, 2, 1e-3, 0.3, 0.2, 2, 200, 1] on FB15K-237 dataset, [5e-6, 2, 1e-3, 0.3, 0.2, 2, 200, 5] on WN18RR dataset, [1e-5, 2, 1e-3, 0.3, 0.2, 2, 400, 1] on Kinship dataset. The optimal hyperparameter values for the decoder are [50, 0.3, 1e-5, 40, 1e-3] on FB15K-237 dataset, [500, 0.0, 1e-5, 40, 1e-3] on WN18RR dataset, and [50, 0.3, 1e-5, 10, 1e-3] on Kinship dataset.

In order to fully verify the effectiveness of our GGAE model, we also design two experiments as follows.

GGAE (Random): Our GGAE model by integrating global information from both direct neighbors and multihop neighbors. The paths used in the path modeling module are random paths.

GGAE (Control): Our GGAE model with control paths integrates global information from both direct neighbors and multihop neighbors. The paths used in the path-modeling module are control paths.

D. Results and Analysis

We evaluate the performance of our model in this section. The experiment results on the test sets of all datasets are shown in Tables IV and V. The best score is in bold and the second best score is underlined. For the results of our GGAE models in Tables IV and V, we set the

[8]https://github.com/nju-websoft/RSN

[9]https://github.com/Peter7Yao/OPTransE

[10]https://github.com/deepakn97/relationPrediction

TABLE VI
EXPERIMENTAL RESULTS ON FB15K-237 TEST SET BY MAPPING PROPERTIES OF RELATIONS

Model (Hits@10)	1-to-1			1-to-M			M-to-1			M-to-M		
	Head	Tail	Avg	Head	Tail	Avg	Head	Tail	Avg	Head	Tail	Avg
TransE	53.7	52.1	52.9	57.3	5.18	31.2	7.00	83.3	45.2	34.7	50.8	42.8
DistMult	19.3	18.2	18.8	51.4	3.94	27.7	3.11	79.3	41.2	32.0	48.5	40.3
ComplEx	41.1	41.1	41.1	55.1	4.95	30.0	4.97	81.8	43.4	37.9	53.3	45.6
ConvE	25.0	25.8	25.4	60.3	13.2	**36.8**	14.7	86.5	50.6	42.6	58.1	50.4
ConvKB	52.6	49.5	51.1	58.3	9.67	34.0	48.4	84.3	66.4	47.4	53.7	50.6
PTransE(ADD)	26.0	26.0	26.0	29.5	7.27	18.4	9.57	42.5	26.0	27.2	33.0	30.1
PTransE(MUL)	27.3	27.1	27.2	38.5	13.4	26.0	12.0	44.9	28.5	30.7	35.8	33.3
PTransE(RNN)	32.8	31.3	32.1	40.5	16.1	28.3	12.6	47.0	29.8	33	37.7	35.4
RSN	29.4	27.5	28.5	30.2	13.2	21.7	15.7	52.9	34.3	43.6	44.2	43.9
OPTransE	54.1	52.5	53.3	64	14.2	39.1	15.6	88.7	52.2	46.9	61.6	54.3
KBGAT	41.1	40.1	40.6	39.5	21.9	30.7	59.7	74.2	67.0	58.2	60	59.1
GGAE(*Random*)	55.2	52.6	**53.9**	44.9	23.8	34.4	62.5	78.4	70.5	59.7	61.8	60.8
GGAE(*Control*)	55.7	50.5	53.1	45.7	24.2	35.0	63.0	78.5	**70.8**	60.2	62.1	**61.2**

TABLE VII
EXPERIMENTAL RESULTS ON WN18RR TEST SET BY MAPPING PROPERTIES OF RELATIONS

Model (Hits@10)	1-to-1			1-to-M			M-to-1			M-to-M		
	Head	Tail	Avg	Head	Tail	Avg	Head	Tail	Avg	Head	Tail	Avg
TransE	97.6	97.6	**97.6**	27.6	6.11	16.9	2.22	19.0	10.6	94.2	94.1	94.2
DistMult	95.2	92.9	94.1	26.9	5.05	16.0	4.71	33.4	19.1	94.8	94.4	94.6
ComplEx	97.6	97.6	**97.6**	28.8	8.63	18.7	5.25	30.9	18.1	95.1	95.0	95.1
ConvE	97.6	97.6	**97.6**	45.1	19.0	32.1	10.7	30.3	20.5	94.7	94.8	94.8
ConvKB	97.6	97.6	**97.6**	47.8	17.1	32.5	16.1	37.6	26.9	94.9	94.6	94.8
PTransE(ADD)	45.2	45.2	45.2	12.4	2.32	7.36	3.06	11.4	7.23	46.5	46.5	46.5
PTransE(MUL)	44.0	44.0	44.0	19.8	5.89	12.8	6.22	16.2	11.2	46.1	45.2	45.7
PTransE(RNN)	45.2	45.2	45.2	9.79	1.26	5.53	0.57	9.78	5.18	46.9	46.6	46.8
RSN	51.7	52	51.9	34.2	2.13	18.2	9.73	30.6	20.2	54.6	53.4	54.0
OPTransE	92.9	90.5	91.7	12.2	2.32	7.26	4.37	10.8	7.59	90.9	90.9	90.9
KBGAT	95.2	97.6	96.4	46.1	24.6	35.4	20.5	34.4	27.5	95.4	95.0	95.2
GGAE(*Random*)	97.6	97.6	**97.6**	45.7	26.7	**36.2**	20.3	33.7	27.0	95.8	95.3	**95.6**
GGAE(*Control*)	97.6	97.6	**97.6**	45.3	26.9	36.1	21.0	35.2	**28.1**	95.4	95.1	95.3

RNN model of the BiLSTM model, and the path modeling method of or method which models entity paths and relation paths separately, other hyperparameters are shown in Table III. The results in Tables IV and V show that our GGAE(Control) model outperforms the baseline models with strong improvement on most metrics in all the datasets. Among that, the Hits@3 metric increases by 2.4 points for FB15k-237 dataset and 0.5 points for the WN18RR dataset, which have relatively shorter control paths and much less incoming and outgoing triples, and by 3.8 points for the Kinship dataset with relatively longer control paths and much denser incoming and outgoing triples. This proves that our model can achieve better results in dense networks with longer control paths. And the results in FB15k-237 and Kinship datasets, which have larger proportions of entity and relation types, could have better performance than the ones in the WN18RR dataset, which is the same as our inference in Section IV-A. Next, we further analyze the results in detail.

1) Path Construction Methods: Tables IV and V also elaborates the comparative experiments on control paths and random paths, where GGAE (random) represents the results of random paths and GGAE(control) represents the results of control paths. The results show that the performance of control paths is much better than that of random paths, which is the same as our inferences in Sections III-B and IV-A. From the above observations, some conclusions can be inferred easily: First, the sequential information on control paths based on control theory is more important and valuable. Second, a longer path facilitates the acquisition of long-distance information, because Table II indicates that the average lengths of control paths are longer than ones of the random paths on all datasets. Third, under the support of control theory, the control paths can pick out more important entities and relations than random paths. So we use control paths in all main experiments.

2) Relation Properties: In this article, we also present a relation graph attention mechanism to learn relation embeddings, so the experiments on relation properties are conducted here. The relations are categorized into four classes [13]: 1-to-1 (if one head of the relation can connect with at most one tail), 1-to-M (if one head of the relation can connect with many tails), M-to-1 (if many heads of the relation can connect with one tail), M-to-M (if many heads of the relation can connect with many tails). According to statistics, the proportions of the four classes in the FB15K-237 dataset are: 7.2% (1-to-1), 11.0% (1-to-M), 34.2% (M-to-1), 47.6% (M-to-M), and in WN18RR are: 18.2% (1-to-1), 36.3% (1-to-M), 27.3% (M-to-1), 18.2% (M-to-M), while the M-to-M relations account for 88% of Kinship dataset, so we do the experiments of the relation properties on FB15K-237 and WN18RR datasets. Tables VI and VII give the results of the Hits@10 metric by mapping properties of relations. Because some baseline models fluctuate greatly in the results of predicting the head and tail entities, we mainly compare the average results. Our GGAE models achieve the highest scores in almost all the sub-tasks. In the FB15K-237 dataset, the GGAE model outperforms the baselines prominently in most of the sub-tasks, while the ConvE model has a better performance than our models in predicting the entities of 1-to-M relations,

TABLE VIII
ABLATION STUDY FOR PATH MODELING METHODS ON THE PATH MODELING MODULE

Method	FB15K-237			WN18RR			Kinship		
	$Hits@3\uparrow$	$MRR\uparrow$	$MR\downarrow$	$Hits@3\uparrow$	$MRR\uparrow$	$MR\downarrow$	$Hits@3\uparrow$	$MRR\uparrow$	$MR\downarrow$
And	48.5	45.7	204	45.2	**41.2**	2518	92.9	89.9	2.03
Or	50.0	47.1	**183**	**45.3**	40.9	**2417**	**95.1**	**93.2**	**1.88**
Type	**50.2**	**47.2**	**183**	44.8	40.2	2579	92.6	90.9	2.08

TABLE IX
ABLATION STUDY FOR RNN TYPES ON THE PATH MODELING MODULE

Type	$Hits@3\uparrow$	$Hits@1\uparrow$	$MRR\uparrow$	$MR\downarrow$
NoRNN	92.2	87.3	90.6	2.10
GRU	92.7	87.9	91.0	2.04
LSTM	92.6	88.3	91.2	1.99
BiGRU	93.7	89.9	92.4	1.99
BiLSTM	**95.1**	**90.6**	**93.2**	**1.88**

but it has lower performance results on other sub-tasks than our models. In the WN18RR dataset, our GGAE models achieve the highest scores in all the sub-tasks. Because the 1-to-1 relations are relatively easy, some baseline models, such as TransE, ComplEx, ConvE, ConvKB, have similar results with our model in predicting the entities of 1-to-1 relations.

V. ABLATION STUDY

In order to fully verify the effectiveness of the components of our model, we give the ablation study here. Note that in each group of experiments, the settings are identical except for the variables of interest. Our model has three main components: path construction module with Control Paths and Random Paths in Section III-B, Path Modeling module with and, or, type in Section III-C, Embedding Aggregation module with relation graph attention and entity graph attention mechanisms in Section III-D. Next, we introduce the ablation experiments for the components which can be deleted or replaced.

A. Ablation Study on Path Construction Module

For the path construction module, the comparative experiments of control paths and random paths are shown in Tables IV–VII and described in above Section IV-D. The experiments on the path construction number are described as follows. The construction method of control paths provides that a graph may have more than one maximum matching with the same size. Different maximum matchings denote different control schemes. By calculating the maximum matching several times, we can obtain a variety of control paths, which can make the model as smooth as possible. The proportions of entity and relation types increase and stabilize as the number of path constructions increases (see Fig. 5). And we design experiments to increase the number of path constructions (maximum matching) from 1 to 10 and observe the changes in model performance. Fig. 6 shows the experimental results on the Kinship dataset. The three evaluation indicators Hits@1, Hits@3, Hits@10 increase with the number of path constructions and tend to be stable. Therefore, the appropriate number of path constructions is very important to achieve better results.

B. Ablation Study on Path Modeling Module

1) Path Modeling Methods: The experimental results of three path modeling methods: and, or, type are shown

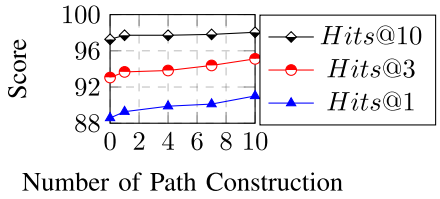


Fig. 6. Ablation study for path construction number on the path construction module.

in Table VIII. It can be observed that the three path modeling methods have their own advantages and disadvantages. From the perspective of segmentation, the and method with undivided paths can capture the potential connection information between entities and relations, while the or method modeling entity paths and relation paths separately can pay more attention to mining sequence information between entities or relations. From the result in Table VIII, we can see that the or model, which computes the entity paths and relation paths separately, achieves much better performance in the WN18RR and Kinship datasets. A powerful explanation is that entities and relations belong to two different types, so modeling entities and relations separately are more conducive to learning their characteristics and reducing the mutual influence. And the length of paths for the and method is always longer than one for the or method, which makes the RNN model cannot capture sequence information in long-distance neighbors.

From the perspective of parameters, the type method designs different weights for each relation type which can fully exploit the potential connection between relation types and entity pairs. The results in Table VIII show that the type model have better performance with 0.1 point in Hits@3 and MRR metrics than the or method in FB15K-237 dataset, and have poor performance in other datasets. The analysis in Section IV-A shows that the number of relations in the FB15k-237 (237) dataset is relatively larger than the WN18RR (11) and Kinship (25) datasets, which should be the main reason for the type method to get better performance in FB15K-237 dataset. However, the number of parameters of the type method increases with the growth of relation types, so the experiment of the type method in the FB15K-237 dataset takes more time and adds massive pressure to the training environment. Therefore, the or model is regarded as our path modeling method in all main experiments.

2) RNN Types: We study how the type of RNNs affects model performance. The RNN models we compare here are GRU, LSTM, BiGRU and BiLSTM models. And the NoRNN model is our GGAE model without RNN operate in the path modeling method. Table IX shows the results of different RNN types on the Kinship dataset. Performance improvements on RNN models over the NoRNN model prove our path modeling method can obtain long-distance information. Comparing the

TABLE X
ABLATION STUDY ON EMBEDDING AGGREGATION MODULE

Method	FB15K-237			WN18RR			Kinship		
	$Hits@3\uparrow$	$MRR\uparrow$	$MR\downarrow$	$Hits@3\uparrow$	$MRR\uparrow$	$MR\downarrow$	$Hits@3\uparrow$	$MRR\uparrow$	$MR\downarrow$
GGAE (Our Best Model)	**50.0**	**47.1**	**183**	**45.3**	**40.9**	2417	**95.1**	**93.2**	1.88
w/o relation graph attention	49.4	46.6	194	45.1	40.8	2582	94.1	91.6	**1.76**
w/o path modeling	49.1	46.5	195	45.1	**40.9**	**2411**	93.7	91.0	1.85
w/o original graph	48.3	45.6	193	41.8	37.2	2965	94.3	92.0	1.79

LSTM (GRU) with the BiLSTM (BiGRU), we observe that the model with bidirectional RNN models can achieve better results than the ones with directional RNN models. This is because both the outgoing and incoming paths of an entity carry valuable information, which is consistent with the ones in the main experiments. At the same time, we find that the model with the LSTM model as RNN model is slightly better than that with the GRU model. So we use the BiLSTM model as our RNN model in all main experiments.

C. Ablation Study on Embedding Aggregation Module

The Embedding Aggregation module has two important components: the entity graph attention mechanism to obtain the entity embeddings, and the relation graph attention method to get the relation embeddings. The entity embeddings are computed with the features from two parts: 1) direct neighbors with original knowledge graph, and 2) multihop neighbors with new knowledge graph $\mathcal{G}'$ constructed by the multihop paths whose embeddings are updated by Path Modeling methods. So, we design the following ablation experiments. The w/o relation graph attention model subtracts the relation graph attention mechanism from our best model, that is, the relation embeddings are not updated in this experiment. The w/o path modeling model subtracts the features from multihop neighbors in paths, where the path construction module and path modeling module are deleted together. The w/o original graph model subtracts the information from direct neighbors, but the features from multihop neighbors are preserved. From Table X, the models without some components of our best model have lower performance in all metrics on FB15K-237 dataset, and in Hits@3 and MRR metrics on WN18RR and Kinship datasets. The MR metric has lower values in some ablation models, but the performance gap is very small. The results on almost all metrics show that the components of our model are very important and indispensable. Compared with Tables IV and V, the w/o path modeling model performs better than the KBGAT model in all datasets, which proves that both incoming and outgoing information is necessary for a better understanding of each entity.

VI. CONCLUSION

We propose GGAE, a GGAE network for relation prediction in knowledge graphs, which enables the embeddings of entities and relations to better capture global information from both direct neighbors and multihop neighbors. Concurrently, two path construction mechanisms and three path modeling methods have been introduced to fully exploit the sequential information. We also present a new relation graph attention to learn relation embeddings, which could be of great help to future related research. The experimental results verify the superiority of our model over state-of-the-art models.

REFERENCES

[1] M. Schuhmacher and S. P. Ponzetto, "Knowledge-based graph document modeling," in *Proc. 7th ACM Int. Conf. Web Search Data Mining*, Feb. 2014, pp. 543–552.

[2] C. Xiong and J. Callan, "EsdRank: Connecting query and documents through external semi-structured data," in *Proc. 24th ACM Int. Conf. Inf. Knowl. Manage.*, Oct. 2015, pp. 951–960.

[3] Y. Hao *et al.*, "An end-to-end model for question answering over knowledge base with cross-attention combining global knowledge," in *Proc. 55th Annu. Meeting Assoc. Comput. Linguistics (Long Papers)*, vol. 1, 2017, pp. 221–231.

[4] D. Diefenbach, K. Singh, and P. Maret, "WDAqua-core1: A question answering service for RDF knowledge bases," in *Proc. Companion Web Conf. Web Conf. (WWW)*, 2018, pp. 1087–1091.

[5] H. He, A. Balakrishnan, M. Eric, and P. Liang, "Learning symmetric collaborative dialogue agents with dynamic knowledge graph embeddings," in *Proc. 55th Annu. Meeting Assoc. Comput. Linguistics*, 2017, pp. 1766–1776.

[6] D. Ghosal, N. Majumder, S. Poria, N. Chhaya, and A. Gelbukh, "DialogueGCN: A graph convolutional neural network for emotion recognition in conversation," in *Proc. Conf. Empirical Methods Natural Lang. Process. 9th Int. Joint Conf. Natural Lang. Process. (EMNLP-IJCNLP)*, Jan. 2019, pp. 154–164.

[7] C. Xu *et al.*, "Graph contextualized self-attention network for session-based recommendation," in *Proc. 28th Int. Joint Conf. Artif. Intell.*, Aug. 2019, pp. 3940–3946.

[8] Y. Xu, Y. Zhu, Y. Shen, and J. Yu, "Learning shared vertex representation in heterogeneous graphs with convolutional networks for recommendation," in *Proc. 28th Int. Joint Conf. Artif. Intell.*, Aug. 2019, pp. 4620–4626.

[9] H. Zeiner, W. Weiss, R. Unterberger, D. Maurer, and R. Jöbstl, "Time-aware knowledge graphs for decision making in the building industry," in *Proc. Int. Conf. Decis. Support Syst. Technol.*, 2019, pp. 57–69.

[10] Q. Cao, B. Li, X. Liang, K. Wang, and L. Lin, "Knowledge-routed visual question reasoning: Challenges for deep representation embedding," *IEEE Trans. Neural Netw. Learn. Syst.*, early access, Jan. 1, 2021, doi: 10.1109/TNNLS.2020.3045034.

[11] R. West, E. Gabrilovich, K. Murphy, S. Sun, R. Gupta, and D. Lin, "Knowledge base completion via search-based question answering," in *Proc. 23rd Int. Conf. World Wide Web (WWW)*, 2014, pp. 515–526.

[12] M. 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.*, 2018, pp. 593–607.

[13] A. Bordes, N. Usunier, A. Garcia-Duran, J. Weston, and O. Yakhnenko, "Translating embeddings for modeling multi-relational data," in *Proc. Adv. Neural Inf. Process. Syst.*, 2013, pp. 1–9.

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

[15] B. Yang, W.-T. 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.*, 2014, pp. 1–12.

[16] T. Trouillon, J. Welbl, S. Riedel, E. Gaussier, and G. Bouchard, "Complex embeddings for simple link prediction," in *Proc. Int. Conf. Mach. Learn.*, 2016, pp. 2071–2080.

[17] T. Dettmers, P. Minervini, P. Stenetorp, and S. Riedel, "Convolutional 2D knowledge graph embeddings," in *Proc. 32nd AAAI Conf. Artif. Intell.*, 2018, pp. 1–14.

[18] 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, Hum. Lang. Technol., (Short Papers)*, vol. 2, 2018, pp. 327–333.

[19] 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.

[20] X. Wang *et al.*, "Heterogeneous graph attention network," in *Proc. World Wide Web Conf.*, May 2019, pp. 2022–2032.

[21] 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. 1–10.

[22] A. García-Durán, A. Bordes, and N. Usunier, "Composing relationships with translations," in *Proc. Conf. Empirical Methods Natural Lang. Process.*, 2015, pp. 286–290.

[23] K. Guu, J. Miller, and P. Liang, "Traversing knowledge graphs in vector space," in *Proc. Conf. Empirical Methods Natural Lang. Process.*, 2015, pp. 1–10.

[24] K. Toutanova, V. Lin, W.-T. Yih, H. Poon, and C. Quirk, "Compositional learning of embeddings for relation paths in knowledge base and text," in *Proc. 54th Annu. Meeting Assoc. Comput. Linguistics (Long Papers)*, vol. 1, 2016, pp. 1434–1444.

[25] X. Lin, Y. Liang, F. Giunchiglia, X. Feng, and R. Guan, "Relation path embedding in knowledge graphs," *Neural Comput. Appl.*, vol. 31, no. 9, pp. 5629–5639, Sep. 2019.

[26] L. Guo, Z. Sun, and W. Hu, "Learning to exploit long-term relational dependencies in knowledge graphs," in *Proc. 36th Int. Conf. Mach. Learn., (ICML)*, Long Beach, CA, USA, vol. 97, Jun. 2019, pp. 2505–2514.

[27] Y. Zhu, H. Liu, Z. Wu, Y. Song, and T. Zhang, "Representation learning with ordered relation paths for knowledge graph completion," in *Proc. Conf. Empirical Methods Natural Lang. Process. 9th Int. Joint Conf. Natural Lang. Process. (EMNLP-IJCNLP)*, 2019, pp. 2662–2671.

[28] H. Huang, Y. Song, Y. Wu, J. Shi, X. Xie, and H. Jin, "Multitask representation learning with multiview graph convolutional networks," *IEEE Trans. Neural Netw. Learn. Syst.*, early access, Dec. 1, 2020, doi: 10.1109/TNNLS.2020.3036825.

[29] X. V. Lin, R. Socher, and C. Xiong, "Multi-hop knowledge graph reasoning with reward shaping," in *Proc. Conf. Empirical Methods Natural Lang. Process.*, 2018, pp. 1–12.

[30] Q. Wang, Z. Mao, B. Wang, and L. Guo, "Knowledge graph embedding: A survey of approaches and applications," *IEEE Trans. Knowl. Data Eng.*, vol. 29, no. 12, pp. 2724–2743, Dec. 2017.

[31] C. Moon, P. Jones, and N. F. Samatova, "Learning entity type embeddings for knowledge graph completion," in *Proc. ACM Conf. Inf. Knowl. Manage.*, Nov. 2017, pp. 2215–2218.

[32] G. Ji, S. He, L. Xu, K. Liu, and J. Zhao, "Knowledge graph embedding via dynamic mapping matrix," in *Proc. 53rd Annu. Meeting Assoc. Comput. Linguistics 7th Int. Joint Conf. Natural Lang. Process.*, 2015, pp. 687–696.

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

[34] W. Xiong, T. Hoang, and W. Y. Wang, "DeepPath: A reinforcement learning method for knowledge graph reasoning," in *Proc. Conf. Empirical Methods Natural Lang. Process.*, 2017, pp. 564–573.

[35] R. Das *et al.*, "Go for a walk and arrive at the answer: Reasoning over paths in knowledge bases using reinforcement learning," in *Proc. 6th Int. Conf. Learn. Represent.*, 2018, pp. 1–18.

[36] W. Chen, W. Xiong, X. Yan, and W. Y. Wang, "Variational knowledge graph reasoning," in *Proc. Conf. North Amer. Chapter Assoc. Comput. Linguistics: Human Lang. Technol., (Long Papers)*, vol. 1, 2018, pp. 1823–1832.

[37] Y.-Y. Liu, J.-J. Slotine, and A.-L. Barabási, "Controllability of complex networks," *Nature*, vol. 473, no. 7346, p. 167, 2011.

[38] K. Murota, *Matrices and Matroids for Systems Analysis*, vol. 20. Springer, 2009.

[39] Y.-Z. Chen, L.-Z. Wang, W.-X. Wang, and Y.-C. Lai, "Energy scaling and reduction in controlling complex networks," *Roy. Soc. Open Sci.*, vol. 3, no. 4, Apr. 2016, Art. no. 160064.

[40] J. E. Hopcroft and R. M. Karp, "An n5/2 algorithm for maximum matchings in bipartite graphs," *SIAM J. Comput.*, vol. 2, no. 4, pp. 225–231, 1973.

[41] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural Comput.*, vol. 9, no. 8, pp. 1735–1780, 1997.

[42] K. Cho *et al.*, "Learning phrase representations using RNN Encoder–Decoder for statistical machine translation," in *Proc. Conf. Empirical Methods Natural Lang. Process. (EMNLP)*, 2014, pp. 1–15.

Qian Li received the M.S. degree from the School of Computer Science and Engineering, Northeastern University, Shenyang, China, in June 2018.

Her research interests include knowledge graph and natural language processing.

Daling Wang received the Ph.D. degree in computer software and theory from Northeastern University, Shenyang, China, in March 2003.

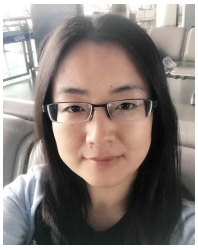
She is currently a Professor with the School of Computer Science and Engineering, Northeastern University. She has authored or coauthored many articles in the top and major tiered journals and conferences, including International Joint Conferences on Artificial Intelligence (IJCAI), Annual Meeting of the Association for Computational Linguistics (ACL), AAAI Conference on Artificial Intelligence (AAAI), International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR), The Web Conference (WWW), and Conference on Empirical Methods in Natural Language Processing (EMNLP). Her research interests include social media processing, sentiment analysis, data mining, and information retrieval.

Shi Feng received the Ph.D. degree in computer software and theory from Northeastern University, Shenyang, China, in January 2011.

He is currently an Associate Professor with the School of Computer Science and Engineering, Northeastern University. He has authored or coauthored more than 20 articles in top-tier journals and conferences, including International Joint Conferences on Artificial Intelligence (IJCAI), Annual Meeting of the Association for Computational Linguistics (ACL), AAAI Conference on Artificial Intelligence (AAAI), International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR), The Web Conference (WWW) and Conference on Empirical Methods in Natural Language Processing (EMNLP). His research interests include sentiment analysis and dialogue systems.

Cheng Niu received the Ph.D. degree from State University of New York at Buffalo, Buffalo, NY, USA, in January 1999.

He is currently an Expert Researcher with Tencent Company, Beijing, China. His research interests include natural language processing and machine translation.



Yifei Zhang received the Ph.D. degree in computer software and theory from Northeastern University, Shenyang, China, in July 2009.

She is currently an Assistant Professor with the School of Computer Science and Engineering, Northeastern University. Her research interests include image processing and machine learning.