IEEE Access
Multidisciplinary ⋮ Rapid Review ⋮ Open Access Journal

Received July 11, 2020, accepted August 24, 2020, date of publication August 26, 2020, date of current version September 11, 2020.

Digital Object Identifier 10.1109/ACCESS.2020.3019592

Knowledge Transfer for Out-of-Knowledge-Base Entities: Improving Graph-Neural-Network-Based Embedding Using Convolutional Layers

ZHONGQIN BI, TIANCHEN ZHANG, PING ZHOU, AND YONGBIN LI
College of Computer Science and Technology, Shanghai University of Electric Power, Shanghai 200090, China

Corresponding author: Tianchen Zhang (zhangtianchen@gmail.com)

This work was supported by the National Nature Science Foundation of China under Grant 61972357 and Grant 61672337.

ABSTRACT Knowledge base completion (KBC) aims to predict missing information in a knowledge base. Most existing embedding-based KBC models assume that all test entities are available at training time. Thus, a question arises-that is, how to answer queries concerning test entities not observed at training time, which is called the out-of-knowledge-base (OOKB) entity problem. In this article, we propose a parameter-efficient embedding model that combines the benefits of a graph neural network (GNN) and a convolutional neural network (CNN) to solve the KBC task with OOKB entities. First, we apply the GNN architecture to learn the information between nodes in the graph. Second, convolution layers are used as a transition matrix in GNN to learn more expressive embeddings with fewer parameters. Finally, we use a transition-based knowledge graph embedding model to solve the KBC task. The model has learnable weights that adapt based on information from neighbors and can exploit auxiliary knowledge for OOKB entities to compute their embedding while remaining parameter efficient. We demonstrate the effectiveness of the proposed model on OOKB datasets, and the code is available at https://github.com/Tianchen627/Knowledge-Transfer-for-Out-of-Knowledge-Base-Entities.

INDEX TERMS Convolutional neural network, graph neural network, knowledge base completion, out-of-knowledge-base entities.

I. INTRODUCTION

Knowledge bases, such as WordNet [1] and Freebase [2], which store complex structured and unstructured information, have important applications in semantic search [3], dialog generation [4], [5] and question answering [6]. These knowledge bases can be viewed as a set of relation triplets, i.e., triplets of the form (h, r, t) with an entity h called the head entity, a relation r, and an entity t called the tail entity [7]. Some examples of these triplets are $(Forrest-Gump, has-director, Robert-Zemeckis)$ and $(Forrest-Gump, is-a, film)$. However, these knowledge bases suffer from incompleteness [8], which means some facts are missing. This problem gives rise to the task of knowledge base completion (KBC), which entails predicting missing facts and whether a given triplet is valid.

Starting with TransE [7], embedding-based KBC models have been successfully applied to large-scale knowledge bases. Since then, some works [9]–[12] have focused on the extension of TransE, whereas others consider semantic matching methods, such as RESCAL [13], [14] and its extensions [15]–[18]. Some works also make use of additional information to improve task performance, e.g., entity types [19], relation paths [20], textual descriptions [21], as well as logical rules [22]. All these models build distributed representations of entities and relations observed in the training data and use various vector operations over the embeddings to predict triplets. Although there are some different solutions, such as the collaborative filtering framework [23], embedding-based models have received the most attention because this method stands out in various KBC tasks. Moreover,increasingly more new technology in deep learning and representation learning can be applied to the KBC embedding-based model.

The state-of-the-art KBC methods are primarily embedding-based models. These popular methods can be broadly classified as graph neural network (GNN)-based models [24]–[26] and convolutional neural network

The associate editor coordinating the review of this manuscript and approving it for publication was Ananya Sen Gupta.

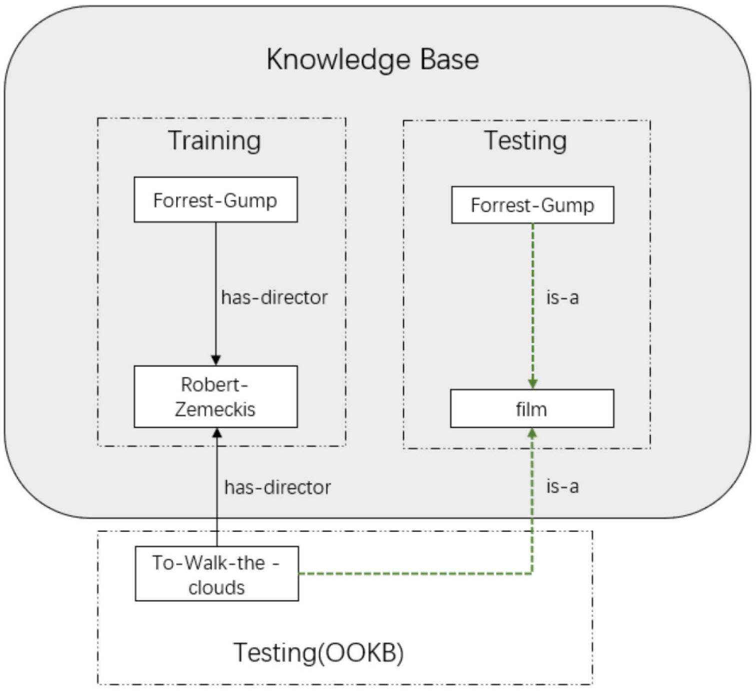


FIGURE 1. **The knowledge base completion (KBC) models can be divided into 2 stages: training stage (1) and testing stage (2). Case (3) is the KBC task with out-of-knowledge-base (OOKB) entities. (1): During training, triplets that represent facts in the knowledge base are fed to the model, for example, ($Forrest - Gump, has - director, Robert - Zemeckis$), which contains the head entity "Forrest-Gump", the relation "has-director" and the tail entity "Robert-Zemeckis". In this stage, the model performs knowledge base embedding by building distributed representations of entities and relations observed in the training data. (2): During testing, the model applies various vector operations over the embeddings to predict missing relation triplets. Suppose the fact "Forrest-Gump is a film", which can be depicted as the triplet ($Forrest - Gump, is - a, film$), is missing from the current knowledge base. The model can find this missing fact through vector operations. For example, if we want to answer the question "What is Forrest Gump?". We can obtain the answer "film" by following the green dashed arrow in the figure. (3): OOKB entity problem. The difference between the traditional KBC task and the OOKB KBC task is that the OOKB test triplet contains an entity that is not observed in the training step, which means the model does not have access to its vector embedding and vector operations cannot work. In the traditional KBC task, all the entities are observed in the knowledge base (shown as the shaded box). In this case, we use the new triplet ($To - Walk - the - clouds, has - director, Robert - Zemeckis$), which is called auxiliary knowledge in the OOKB situation, to obtain the embedding of the new entity. Thus, we can answer the question "What is to walk the clouds?" by predicting the triplet ($To - Walk - the - clouds, is - a, film$).**

(CNN)-based models [27], [28]. The former are known for the ability to incorporate the connectivity structure in the graph since the knowledge base can be seen as a graph, and the latter are known for their highly parameter efficiency.

The out-of-knowledge-base (OOKB) entity problem in the KBC task was first introduced in [29]. Before that, there were many studies focusing on OOKB in different knowledge base applications and tasks [30]–[33]. The OOKB entity problem arises when new entities (OOKB entities) occur in the triplets that are given to the system after training. Although we can retrain the model with the new triplets containing the OOKB entities, a method to avoid costly retraining is desirable. Fig.(1) illustrates the KBC task with OOKB entities schematically, and a more detailed definition can be found in Section III.

As these entities were unknown to the system at training time, the system does not have their embeddings and, hence, does not have a means to predict relations for these entities. Therefore, traditional KBC methods cannot be applied directly to the OOKB KBC task. A GNN method was used to solve the OOKB entity problem in [29]. The method embodies knowledge transfer,which is a process included in knowledge management domain. However, GNN models suffer from potentially prohibitive memory requirements [24] and generally do not outperform CNN-based models [28].

The OOKB problem is of practical importance because OOKB entities can occur whenever new entities, such as events and products, are produced, which happens everyday, and in domain-specific knowledge base where the range of entities is limited. Moreover, we often want to infer more facts (triplets) from the knowledge (triplets) we already have.

In this article, we propose a parameter-efficient embedding model that combines the benefits of GNN and CNN by replacing the transition weight matrix in GNN, which represents the relations, with a multilayer convolutional network. The model has learnable weights that adapt to the amount of information from neighbors and can exploit auxiliary knowledge for OOKB entities to compute their embeddings while remaining parameter efficient.

Our contributions are summarized as follows:

1. We propose an end-to-end model for the KBC task with OOKB entities. The model combines the advantages of the GNN structure and the CNN structure.

2. We develop a new method to transfer knowledge for OOKB entities. In contrast to using a vector or weight matrix to represent relation embeddings in the KBC model, we use a convolution kernel to learn expressive features from the auxiliary knowledge of OOKB entities.

3. We verify the effectiveness of our model in OOKB datasets. The model has good accuracy and parameter efficiency. Since the OOKB problem often occurs in realistic cases, our work is a successful attempt to combine the KBC task with practical scenarios.

The remainder of this article is organized as follows. We discuss related work in Section II and introduce the background of KBC and OOKB in Section III. We describe our approach in detail in Section IV. Then, we elaborate our experimental study in Section V and compared the results with the original GNN-for-KBC. The conclusion is presented in Section VI.

II. RELATED WORK

The knowledge graph embedding model has been an active research topic for KBC since TransE [7] addressed the task by projecting both entities and relations into the same embedding vector space with a translational constraint of $h + r \approx t$. Later works introduced new representations of relational translations and thus improved the performance and increased the model complexity. Recent research incorporates GNN [24]–[26] and CNN structures [27], [28] into the model.

GNN-based models are famous for taking the graph structure into consideration. Since previous works consider each triplet independently without taking into account the relationships between triplets, GNN-based models aggregate local information in the graph neighborhood for each node. Graph convolutional networks (GCNs) have been an effective tool to create node embeddings and were first applied to the KBC task in [24]. Relational GCN (R-GCN) [25] is an extension that performs well for highly multirelational data. Furthermore, in [26], graph attention was applied to the GNN. However, these models have been criticized for their huge memory requirements and failure to outperform CNN-based models [27].

GCN approaches define convolutions directly on a graph and sum node features over all spatial neighbors using an adjacency matrix, whereas our method replaces the transition weight matrix in the GNN with a multilayer convolutional network. GCN suffers from prohibitive memory requirements, while our work maintains the parameter efficiency of CNN-based models.

CNN-based models use convolutional layers to learn embeddings because the performance in previous work was limited by feature interactions. ConvE [27] was the first model to use 2D convolutions over embeddings of different embedding dimensions to extract more feature interactions. InteractE [28] later extended ConvE through more feature interactions via feature permutation, checkered reshaping and circular convolution. These models have good parameter efficiency: ConvE achieves better scores than R-GCNs on FB15k-237 with 17× fewer parameters. The reason why CNN-based models have high parameter efficiency is that the convolutional layer has more feature interactions that can make the model learn more expressive features in low embedding size with fewer parameters. More details about CNN-based models and parameter efficiency can be found in Section IV-C. However, CNN-based models do not use the relationships between triplets: every triplet is treated independently.

The traditional KBC supposes that all the test entities are observed in the training data, so every entity's embedding is available. The situation changes when a new entity occurs in the test triplets, and the traditional KBC models cannot be applied in this scenario because they do not have the embeddings of these entities. This situation is called the OOKB entity problem. We focus on handling the KBC task with the OOKB entity problem in this article.

The most closely related work to ours is GNN-for-OOKB [29], which proposed a GNN model for the KBC task with OOKB entities. We improve the model by replacing the transition weight matrix in the GNN with a multilayer convolutional network to take advantage of the high computational efficiency of the convolutional network and adapt to information from neighbors with the GNN structure.

Another similar work [34] proposed an end-to-end graph structure-aware convolutional network (SACN) that combines the benefits of GCN and ConvE. However, their work is not designed for the OOKB entity problem, and the SACN cannot handle OOKB entities. Moreover, they focus on link prediction, while we focus on triplet classification, which are both typical KBC tasks [35].

There are also some recent works focusing on the OOKB problem in the KBC task recently. Some researchers focus on different type of tasks like link prediction [36], [37] in few-shot learning and entity detection [38] while we focus on triplet classification. Some researchers used the jointly embedding method [39] and multimodal data enhanced representation [40] to achieve OOKB entity embedding while in our work these external data are not considered. Reference [41] used attention-based aggregation to solve the new OOKB relation problem. Their idea and method are exciting and we want to extend our work to the OOKB relation problem in the future. Many studies assumed a specific scenario while our work considers only the standard scenario of the OOKB entity problem in which all the knowledge we have is from the current knowledge base.

III. BACKGROUND

A. KNOWLEDGE GRAPH

A knowledge base $\mathcal{G}$ contains many facts that can be represented as a triplet (h, r, t) with head entity h, relation r, and tail entity t. A knowledge base can also be called a knowledge graph because each triplet in $\mathcal{G}$ can be regarded as a labeled edge in a graph. The entities in a triplet correspond to nodes, and the relations are the edges. Therefore, some algorithms that are applied to graphs can also be applied to the knowledge base as a graph-structured knowledge graph.

Now, we define $\mathcal{E}$ as a set of entities and $\mathcal{R}$ as a set of relations. The fact, or triplet (h, r, t), has $h, t \in \mathcal{E}$ and $r \in \mathcal{R}$. Let $\mathcal{G}_{\text{gold}} \subset \mathcal{E} \times \mathcal{R} \times \mathcal{E}$ be the set of gold facts, that is, ground truth facts in knowledge base $\mathcal{G}$. If a triplet is in $\mathcal{G}_{\text{gold}}$, we say it is a positive triplet; otherwise, it is a negative triplet. Suppose we have an incomplete knowledge base $\mathcal{G} \subset \mathcal{G}_{\text{gold}}$; the goal of KBC is to identify $\mathcal{G}_{\text{gold}}$ by finding all the missing facts in knowledge base $\mathcal{G}$.

B. OOKB ENTITY PROBLEM

In the OOKB situation,we have OOKB entities that are not observed in the training step.

With knowledge base $\mathcal{G}$ observed at training time, new triplets $\mathcal{G}_{\text{aux}}$ are provided at test time, with $\mathcal{E}(\mathcal{G}_{\text{aux}}) \not\subset \mathcal{E}(\mathcal{G})$ and $\mathcal{R}(\mathcal{G}_{\text{aux}}) \subseteq \mathcal{R}(\mathcal{G})$. The aux contains new entities, but no new relations are involved. We call these new entities OOKB entities, as $\mathcal{E}_{OOKB} = \mathcal{E}(\mathcal{G}_{\text{aux}}) \backslash \mathcal{E}(\mathcal{G})$. Then, we have $\mathcal{E} = \mathcal{E}(\mathcal{G}) \cup \mathcal{E}_{OOKB} \neq \mathcal{E}(\mathcal{G})$. In this situation, the KBC task is to correctly identify missing relation triplets that involve the OOKB entities $\mathcal{E}_{OOKB}$.,The difference from the traditional KBC task is that the embeddings for these entities are missing: they must be computed with the help of $\mathcal{G}_{\text{aux}}$, every triplet of which contains exactly one OOKB entity in $\mathcal{E}_{OOKB}$ and one entity in $\mathcal{E}(\mathcal{G})$.

C. KBC: TRIPLET CLASSIFICATION

Triplet classification, a typical KBC task first introduced in [7] and regarded as a standard benchmark for KBC methods, aims to verify whether an unseen triplet (h, r, t) is true, e.g., $(Forrest-Gump, has-director, Robert-Zemeckis)$ should be classified as a true fact while $(Forrest - Gump, has - director, James-Cameron)$ should be classified as false [35].

Viewed as a machine learning problem, triplet classification is a classifier induction task in which $\mathcal{E}$ and $\mathcal{R}$ are given, and knowledge base $\mathcal{G}$ forms the training set (with only positive examples), with $\mathcal{G}_{\text{test}}$ being the test set. The set $\mathcal{G}_{\text{test}}$ can be divided into the set of positive test examples $\mathcal{H} \cap (\mathcal{G}_{\text{gold}}) = (\mathcal{G}_{\text{gold}}) \backslash \mathcal{G}$ and the set of negative test examples $\mathcal{G}_{\text{test}} \backslash \mathcal{G}_{\text{gold}}$.

For the KBC task, the existing knowledge base $\mathcal{G}$ is assumed to be incomplete, which means that some triplets that must be present in $\mathcal{G}$ are missing; i.e., $\mathcal{G} \neq \mathcal{G}_{\text{gold}}$.

Here, we define $\mathcal{H} = (\mathcal{E} \times \mathcal{R} \times \mathcal{E}) \backslash \mathcal{G}$ as the set of triplets not presented in $\mathcal{G}$. The set does not appear during training or in $\mathcal{G}$. We assume that $\mathcal{G}$ is incomplete, so two cases are possible for each triplet $x \in \mathcal{H}$; that is, $x \in \mathcal{G}_{\text{gold}}$: positive triplet and $x \notin \mathcal{G}_{\text{gold}}$: negative triplet. For the former case, the triplet x is not contained in knowledge base $\mathcal{G}$ because of incompleteness. Thus, we encounter the problem of determining which of the above two possible cases holds for each triplet not present in $\mathcal{G}$. This is the approach used for triplet classification in the KBC task.

IV. OUR APPROACH

A. OVERVIEW

In this article, we use an embedding-based model to solve the KBC task. We propose a parameter-efficient embedding model that combines the benefits of GNN and CNN by replacing the transition weight matrix in GNN, which represents the relation, with a multilayer convolutional network. The overall structure is based on a GNN.

A GNN structure consists of two models, the propagation model and the output model [42]. In some papers [26], [34], these two components are called the encoder and decoder. The propagation model determines how to propagate information between nodes in a graph while learning information from the graph structure and the neighbor nodes. The output model defines an objective function according to given tasks using vector-represented nodes and edges. In this article, we focus on the propagation component, modifying the structure to improve its parameter efficiency. For the output model, we retain the settings in [29]. A simple illustration of our model is given in Fig.2, and the detailed procedure is described in Algorithm 1.

B. GNN FOR KBC

GNN is a popular embedding-based model for KBC tasks. In contrast to some existing GNN structures that encode the entire graph into a vector, GNNs for KBC models encode nodes and edges into vectors. Let $\mathcal{G}$ be a knowledge graph and $e \in \mathcal{E}(\mathcal{G})$ be an entity. The head neighborhood $\mathcal{N}_{head}$ and tail neighborhood $\mathcal{N}_{tail}$ of the entity e are:

$$\mathcal{N}_{head}(e) = \{(h, r, e) | (h, r, e) \in \mathcal{G}\} \quad (1)$$

$$\mathcal{N}_{tail}(e) = \{(e, r, t) | (e, r, t) \in \mathcal{G}\} \quad (2)$$

$\mathrm{v_e} \in \mathbb{R}^d$ is a d-dimensional representation vector of e. Our goal is to obtain $\mathrm{v_e}$ through the propagation model by means of the following equation:

$$\mathrm{v_e} = P(S_{head}(e) \cup S_{tail}(e)) \quad (3)$$

Here, $S_{head}(e)$ and $S_{tail}(e)$ are sets of vectors. $S_{head}(e)$ contains the representation vectors of neighborhood $\mathcal{N}_{head}(e)$, and $S_{tail}(e)$ contains the representation vectors of neighborhood $\mathcal{N}_{tail}(e)$. P is a pooling function that maps a set of

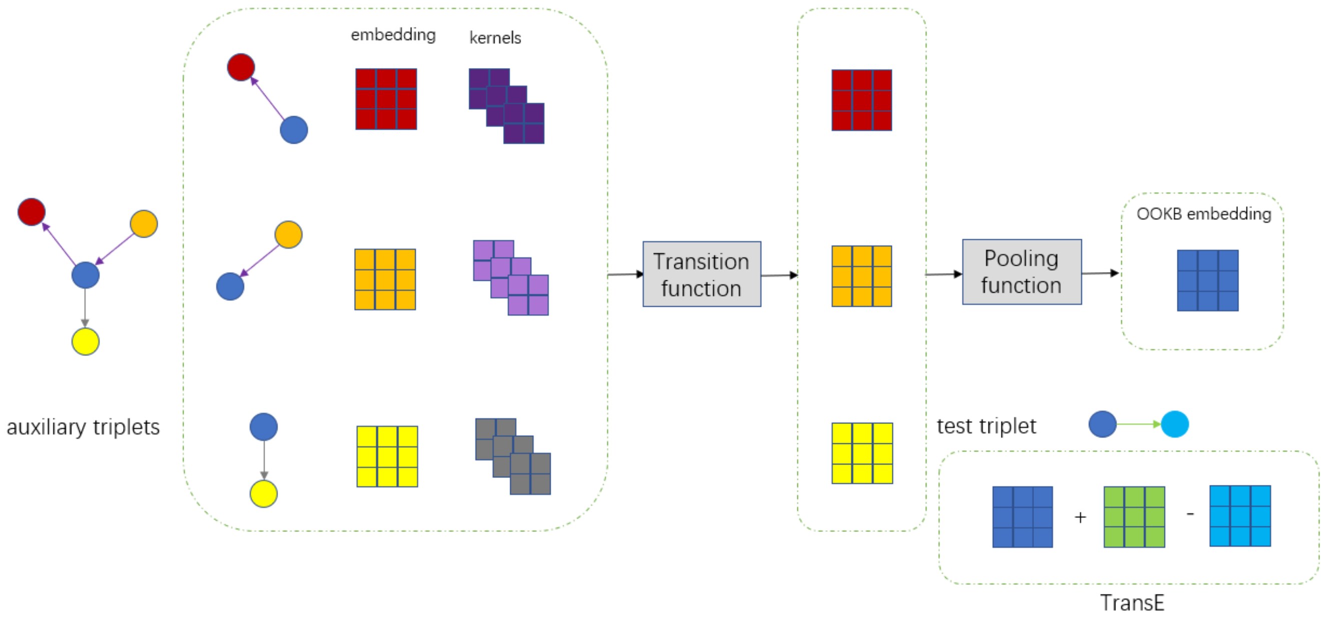


FIGURE 2. An illustration of OOKB knowledge base completion. Here, the different colors of the nodes and edges represent different entities and relations, and the blocks are their embeddings. Notably, a relation in a different direction has a different embedding, such as the purple relation in the figure, which has a different embedding when it goes in or goes out blue entity. These embedding are used as convolution kernels in the model. The figure is only a sketch; more details can be found in Algorithm 1.

vectors into a vector, i.e., $P : 2^{\mathbb{R}^d} \rightarrow \mathbb{R}^d$. The objective is to extract shared aspects from a set of vectors. For $S = \left\{x_i \in \mathbb{R}^d\right\}_{i=1}^N$, some common pooling function are as follows:

$$P(S) = \sum_{n=1}^{N} x_i \tag{4}$$

$$P(S) = \frac{1}{N}\sum_{n=1}^{N} x_i \tag{5}$$

$$P(S) = max(\{x_i\}_{i=1}^N) \tag{6}$$

Equation (4)(5)(6) are sum pooling, average pooling, and max pooling, respectively. In [42], sum pooling was used in the propagation model, while in [29], average pooling was used and proved to be the best choice for KBC. In this article, we use average pooling based on previous experience.

The sets $S_{head}(e)$ and $S_{tail}(e)$ can be represented as follows:

$$S_{head}(e) = \{T_{head}(\mathrm{v_h}; h, r, e) | (h, r, e) \in \mathcal{N}_{head}(e)\} \tag{7}$$

$$S_{tail}(e) = \{T_{tail}(\mathrm{v_t}; e, r, t) | (e, r, t) \in \mathcal{N}_{tail}(e)\} \tag{8}$$

$T_{head}, T_{tail} : \mathbb{R}^d \times \mathcal{E}(\mathcal{G}) \times \mathcal{R}(\mathcal{G}) \times \mathcal{E}(\mathcal{G}) \rightarrow \mathbb{R}^d$ are called transition functions, and they are used to transform the vector of the head neighborhood entity and tail neighborhood entity of e to the vector $\mathrm{v_e}$, depending on the edge between them. Various transition functions have been proposed, and many neural network techniques can be used, such as batch-normalization, residual connection, and long short-term memory. Here, we introduce the transition function that is most closely related to our work, which was proposed in GNN for OOKB [29]:

$$T_{head}(\mathrm{v_h}; h, r, e) = \mathrm{ReLu}(\mathrm{BN}(A_r^{head}\mathrm{v_h})) \tag{9}$$

$$T_{tail}(\mathrm{v_t}; e, r, t) = \mathrm{ReLu}(\mathrm{BN}(A_r^{tail}\mathrm{v_t})) \tag{10}$$

where BN indicates batch normalization. The above equation makes the transition function dependent on the relation between the current node (entity) and the neighbor. Note that the parameter matrices are defined individually for the relation r. The reverse relationship is considered as in different situations $\mathrm{v_e}$ is either a head entity or tail entity. Therefore, every relation in the knowledge base has two parameter matrices, one for transforming the head entity and the other for transforming the tail entity.

Overall, the transition function can be written as:

$$\mathrm{v_e} = T(x) = \begin{cases} \mathrm{ReLu}(\mathrm{BN}(A_r^{head}\mathrm{v_h})), & \text{if } x = (h, r, e), \\ \mathrm{ReLu}(\mathrm{BN}(A_r^{tail}\mathrm{v_t})), & \text{if } x = (e, r, t), \end{cases} \tag{11}$$

Here, $\mathrm{v_e}$ is the candidate embedding vector of entity e, which is obtained by a single triplet, while $S_{head}(e)$ and $S_{tail}(e)$ always contain multiple of triplets. By means of (3), we can obtain the final embedding $\mathrm{v_e}$ of e.

C. PARAMETER EFFICIENCY AND CONVOLUTIONAL LAYER

The efficiency of an algorithm can be measured based on the usage of different resources. In deep learning, a deeper architecture and higher embedding dimension always lead to better performance. However, the cost of such a model is enormous, with higher capacity of calculation and memory usage, and the number of parameters in the model will be astronomical. Therefore, many works focus on the parameter efficiency of

Algorithm 1 Test Stage of Our Model

```
Input: Test triplets $\mathcal{G}_{\text{test}}$, auxiliary triplets $\mathcal{G}_{\text{aux}}$, transition
    function $T$, pooling function $P$, threshold $\gamma$, entity
    embedding dictionary $\mathcal{E}$, relation embedding dictionary
    $\mathcal{R}$
Output: result dictionary $Result$
 1: $\mathcal{N}_{head}(e) = \{(h, r, e)|(h, r, e) \in \mathcal{G}_{\text{aux}}\}$ //head neighbor-
    hood entity set
 2: $\mathcal{N}_{tail}(e) = \{(e, r, t)|(e, r, t) \in \mathcal{G}_{\text{aux}}\}$ //tail neighborhood
    entity set
 3: for each $(h, r, t) \in \mathcal{G}_{\text{test}}$ do
 4:     if $h \notin \mathcal{E}$ then //head entity is the OOKB entity
 5:         $S_{head}(h) = \{T(h_1, r, h)|(h_1, r, h) \in \mathcal{N}_{head}(h)\}$
 6:         $S_{tail}(h) = \{T(h, r, t_1)|(h, r, t_1) \in \mathcal{N}_{tail}(h)\}$
 7:         $\text{v}_\text{h} = P(S_{head}(e) \cup S_{tail}(e))$
 8:         $\text{v}_\text{r}, \text{v}_\text{t} = \mathcal{R}(r), \mathcal{E}(t)$
 9:     else if $t \notin \mathcal{E}(\mathcal{G})$ then //tail entity is the OOKB entity
10:         $S_{head}(t) = \{T(h_1, r, t)|(h_1, r, t) \in \mathcal{N}_{head}(t)\}$
11:         $S_{tail}(t) = \{T(t, r, t_1)|(t, r, t_1) \in \mathcal{N}_{tail}(t)\}$
12:         $\text{v}_\text{t} = P(S_{head}(e) \cup S_{tail}(e))$
13:         $\text{v}_\text{r}, \text{v}_\text{h} = \mathcal{E}(r), \mathcal{E}(h)$
14:     else//no OOKB entity
15:         $\text{v}_\text{h}, \text{v}_\text{r}, \text{v}_\text{t} = \mathcal{E}(h), \mathcal{R}(r), \mathcal{E}(h)$
16:     end if
17:     score=$\|\text{v}_\text{h} + \text{v}_\text{r} - \text{v}_\text{t}\|$
18:     if score $< \gamma$ then
19:         $Result((h, r, t)) = True$
20:     else
21:         $Result((h, r, t)) = False$
22:     end if
23: end for
```

the model and design the model to have acceptable performance with fewer parameters. Various model compression techniques, such as distillation, pruning and quantization, can be used to reduce the number of network parameters by removing redundant parameters with minimal loss in performance.

In the KBC task, ConvE [27] is known for its high parameter efficiency and robust performance on different datasets, achieving better scores than DistMult and R-GCNs on FB15k-237 with 8× and 17× fewer parameters. When the embedding size is 200, the ConvE model contains nearly 5M parameters. The embedding layer, convolutional layer, and projection layer account for 2.96M, 320, and 1.96M parameters, respectively. The convolutional layer needs few parameters for the filter, and the embedding layer accounts for a large proportion of the total number of parameters. The number 2.96M is obtained from $(14541 + 237) * 200 = 2,955,600$, where 14541 is the entity number in FB15k-237 and 237 stands for the relation number. A larger knowledge base would result in more parameters, especially in the embedding layer, so controlling the embedding layer size is very important. Moreover, a model that achieves robust performance with a small embedding size, such as ConvE, is highly desirable. In the GNN framework, the design of the transition function is key to improving the parameter efficiency because when the embedding size is fixed, the transition function determines the performance of the model and the number of of parameters.

The scoring function of ConvE is defined as follows:

$$\psi_r(\text{e}_\text{s}, \text{e}_\text{o}) = f(\text{vec}(f([\overline{\text{e}_\text{s}}; \overline{\text{r}_\text{r}}] * w))\text{W})\text{e}_\text{o} \tag{12}$$

e_s represents the subject entity and e_o is the object entity. A higher score represents a higher likelihood the triplet is true. We believe ConvE can be used as the transition function in the propagation model, so we reorganize the equation as (13).

$$\text{v}_\text{e} = T(x) = \begin{cases} \text{ReLu}(\text{BN}(\text{vec}([\overline{\text{v}_\text{h}}; \overline{\text{v}_\text{r}}] * w)\text{W})), & \text{if } x = (h, r, e), \\ \text{ReLu}(\text{BN}(\text{vec}([\overline{\text{v}_\text{t}}; \overline{\text{v}_\text{r}}] * w)\text{W})), & \text{if } x = (e, r, t), \end{cases} \tag{13}$$

where vec() is the flatten operation, w is the convolution operation and W is the parameter matrix for projection. $[\overline{\text{e}}; \overline{\text{r}}]$ denotes 2D reshaping of vectors e and r.

The main difference between (11) and (13) is the treatment of the relation embedding. In GNN (11), the embedding of relations is achieve via weight matrix A used for projection, and every relation has two corresponding matrices for the sake of the alternative relation direction, which means the reverse relationship has a separate embedding. For ConvE (13), every relation is embedded as a unique vector v_r. Convolutional filter w and projection matrix W are applied for the transition, and all these components share parameters.

Here, we represent the relation as a convolution kernel, which can learn expressive features from the auxiliary knowledge of OOKB entities. In contrast to ConvE, we discard the vector representation in our propagation model. The alternative is to directly convolve v_h or v_t with the specific convolution kernel w_r^{head} or w_r^{tail} representing the corresponding relation. The convolution kernels no longer share parameters, and every relation has two corresponding convolution kernels for whether the direction is head to tail or tail to head. The convolution operation can either be 1D convolution, which is always used in NLP work, or 2D convolution, commonly used in CV work (ConvE is the first KBC model to use 2D convolutional layers). In addition, if 2D convolutions is applied, the embedding vector of a given entity v_h or v_t should be reshaped to a matrix for the 2D convolutional layer. In a later experiment, we found the performance of 2D convolution to be slightly better. For our new transition function, see (14).

$$\text{v}_\text{e} = T(x) = \begin{cases} \text{ReLu}(\text{BN}(\text{vec}[\text{v}_\text{h}] * w_r^{head})\text{W}_\text{r}^\text{head})), & \text{if } x = (h, r, e), \\ \text{ReLu}(\text{BN}(\text{vec}[\text{v}_\text{r}] * w_r^{tail})\text{W}_\text{r}^\text{tail})), & \text{if } x = (e, r, t), \end{cases} \tag{14}$$

The formula retains a different projection weight matrix for each relation. The number of parameters appears to increase,

TABLE 1. Given a triplet x, whether $x = (h, r, e)$ or $x = (e, r, t)$, transition function $T(x)$ is used to obtain the embedding $\mathrm{v_e}$ of entity e. We modify the score function from ConvE(12) into a transition function suitable for the propagation model.

Model	Transition function	triplet
GNN-for-OOKB	$\mathrm{ReLu}(\mathrm{BN}(A_r^{head}\mathrm{v_h})$	(h, r, e)
	$\mathrm{ReLu}(\mathrm{BN}(A_r^{tail}\mathrm{v_t})$	(e, r, t)
ConvE (modified)	$\mathrm{ReLu}(\mathrm{BN}(\mathrm{vec}([\overline{\mathrm{v_h}}; \overline{\mathrm{v_r}}] * w)\mathrm{W}))$	(h, r, e)
	$\mathrm{ReLu}(\mathrm{BN}(\mathrm{vec}([\overline{\mathrm{v_t}}; \overline{\mathrm{v_r}}] * w)\mathrm{W}))$	(e, r, t)
proposed model	$\mathrm{ReLu}(\mathrm{BN}(\mathrm{vec}[\mathrm{v_h}] * w_r^{head})\mathrm{W_r^{head}}))$	(h, r, e)
	$\mathrm{ReLu}(\mathrm{BN}(\mathrm{vec}[\mathrm{v_r}] * w_r^{tail})\mathrm{W_r^{tail}}))$	(e, r, t)

but we can achieve better performance with lower embedding size, which means the number decrease overall. The final performance is shown in Table 6, and Table 1 summarizes all the transition functions mentioned above.

D. OBJECTIVE FUNCTION

The objective function is designed to guide the training for the task. We use TransE [7] as the output model because TransE is the most representative translational distance model and the translational property of TransE is highly regarded [34]. What is different is that we use an absolute-margin objective function instead of the pairwise-margin objective function in original TransE.

The objective function is as follows:

$$\mathcal{L} = \sum_{i=1}^{N} f(h_i, r_i, t_i) + [\gamma - f(h_i', r_i', t_i')]_+ \tag{15}$$

Here, $[x]_+$ is the hinge function max(0,x). (h_i, r_i, t_i) denotes a positive triplet in the training set, and (h_i', r_i', t_i') denotes a negative triplet generated by negative sampling. $\gamma \in \mathbb{R}$ is a threshold, also called the margin. The objective function requires the score $f(h_i, r_i, t_i)$ to be greater than the score (h_i', r_i', t_i') by at least γ. The score function f evaluates the implausibility of a triplet (h, r, t): smaller scores indicate that the triplet is more likely to be positive. In TransE [7], the score function is defined by $f(h, r, t) = \|\mathrm{v_h} + \mathrm{v_r} - \mathrm{v_t}\|$. The objective function is used to optimize the scores for the positive triplets towards zero, whereas the scores of the negative triplets are going to be at least γ.

Some recent research used CNN-based models as the output model for GNN [26], [34]. However, these works studied link prediction, another KBC task. We attempted to adapt the CNN structure to triplet classification, but the results were not satisfactory. Therefore, we chose TransE, a parameter-efficient model whose parameters consist of only embedding parameters, as the output model.

V. EXPERIMENT

A. DATASETS

The OOKB datasets were generated through WordNet11 [1], [29]. The data files can be downloaded from https://github.com/takuo-h/GNN-for-OOKB.

TABLE 2. Specifications of the WordNet11. All the training triplets are positive. Half of the validation and test sets are negative triplets, and these are included in the numbers of validation triplets and test triplets.

	WordNet11
Relations	11
Entities	38,696
Training triplets	112,581
Validation triplets	5,218
Test triplets	21,088

The specifications of these datasets are shown in Table 2 and Table 3. The datasets include nine independent datasets obtained by different filtering and splitting methods that can be denoted by Head,Tail,Both-1,000,3,000,5,000, respectively, where the first part represents the position of the OOKB entities and the second part represents the number of triplets used for generating the OOKB entities. More details about the datasets can be found in the original paper [29]. All entities and relations come from WordNet11, and every dataset contains training, validation, and test sets. In addition, the validation and test sets include positive and negative triplets, whereas the training set does not contain negative triplets.

Here, we use Both-1000 and Both-5000 for our experiment, as the OOKB entities can be either head or tail entities, which is a more general situation in practice. Both-1000 contains 93,364 training triplets and 1,238 OOKB entities, while Both-5000 contains 57,601 training triplets and 4,963 OOKB entities. Thus, the difficulty of Both-1000 is relatively low, and Both-5000 is more difficult: we want to evaluate our model in both easy and difficult scenarios. Details of the datasets are shown in Table 4.

B. IMPLEMENTATION AND HYPERPARAMETERS

Recently, many state-of-the-art empirical results have been challenged for whether they were achieved due to a better model/algorithm or simply by means of a more extensive hyperparameter search [43]. Since we want to compare our work with GNN-for-OOKB [29], we use the same hyperparameters including training epoch,batch size,and learning rate provided in the previous work for the sake of fairness.

All networks were trained by stochastic gradient descent with the Adam optimizer [44]. The step size of Adam was $\alpha_1/(\alpha_2 \cdot k + 1.0)$, where k indicates the current epoch in training, $\alpha_1 = 0.01$, and $\alpha_2 = 0.0001$. The batch size was 5000, and the threshold $\gamma = 300$. The embedding size was 200 in GNN-for-OOKB, and the number of training epochs was 500 in every experiment. We trained our model from the beginning and we did not use pretraining.

To generate corrupted triplets, we used the "Bernoulli" trick, a technique also used in [9]. When corrupting triplets, the Bernoulli trick set different probabilities for replacing the head or tail entity to reduce the chance of generating false negative labels.

For the hyperparameters in the transition function, we set the kernel size to 3 × 3 and the number of channels to 10.

TABLE 3. The OOKB datasets. The numbers of triplets in the validation and test sets include negative triplets. All relations and entities come from WordNet11.

	Head			Tail			Both		
	1000	3000	5000	1000	3000	5000	1000	3000	5000
OOKB entities	348	1,034	1,744	942	2,627	4,011	1,238	3,319	4,963
Training triplets	108,197	99,963	92,3091	96,968	78,763	67,774	93,364	71,097	57,601
Validation triplet	4,613	4,184	3,845	3,999	3,122	2,601	3,799	2,759	2,166
Test triplet	994	2,969	4,919	986	2,880	4,603	960	2,708	4,196

TABLE 4. "Both" means both head and tail entities can be OOKB entities. "1000" and "5000" denote the number of OOKB entities.

	Both-1000	Both-5000
Relations	11	11
Entities	38,696	38,696
OOKB entities	1,238	4,963
Training triplets	93,364	57,601
Validation triplet	3,799	2,166
Test triplet	960	4,196
Auxiliary entities	9,899	23,792
Auxiliary triplets	18,638	48,425

TABLE 5. Triplet number of a given relation in training set of both-1000.

relation	number
type_of	25,171
synset_domain_topic	3,100
has_instance	29,772
member_holonym	7,950
part_of	5,690
has_part	5,243
member_meronym	8,011
similar_to	1,612
subordinate_instance_of	2,971
domain_region	3,309
domain_topic	535

These settings are from the original ConvE paper. The dimension of the embedding space was 36 in the proposed model; thus, in the feed-forward pass, the input vector is reshaped as $\mathbb{R}^{6\times6}$ and then the convolution operation is applied to the $\mathbb{R}^{10\times3\times3}$ filter.

C. RESULTS AND ANALYSIS

The performance of the model is shown in Table 6. From Section II, we know that most popular KBC models cannot handle OOKB entities because they are not designed for such entities. Here, we choose GNN-for-OOKB [29] as the baseline and run both models with the settings in Section V-B. The proposed model performs better with fewer parameters, which means the model has higher parameter efficiency.

As discussed in Section IV-C, the embedding results in more parameters, and ConvE [27] is highly parameter efficient because it performs well with a low embedding size. The proposed model is similar, and its high parameter efficiency is a result of the robust performance at low embedding size. Table 7 shows the results from our ablation study where we evaluate the performance of both models on the Both-1000 dataset with different embedding sizes. The proposed model always outperforms the previous model for a given parameter scale. The proposed model can learn more expressive features in low embedding size with fewer parameters.

TABLE 6. Results. The accuracy of triplet classification in different datasets.

model	Param.count	Both-1000	Both-5000
GNN-for-OOKB	8.6M	73.33%	63.79%
proposed model	1.7M	74.89%	64.56%

TABLE 7. Performance with different embedding size in Both-1000 dataset.

model	Param.count	Emb.size	accuracy
GNN-for-OOKB	0.98M	25	69.27%
GNN-for-OOKB	1.4M	36	71.67%
GNN-for-OOKB	1.9M	49	72.27%
GNN-for-OOKB	2.6M	64	72.68%
proposed model	1.1M	25	71.77%
proposed model	1.7M	36	74.89%
proposed model	2.4M	49	74.69%
proposed model	3.4M	64	75.52%

We investigate the training process in the Fig.3. The image has 12 subcharts: the former 11 contain the triplet scores of 11 different relations in the both-1000 dataset, showing how the scores change with learning, and the last is the overall performance, that is, the triplet classification accuracy on the test set. The x label means epoch. In the first 11 charts, we sampled the triplet score during training, and every red and blue line indicates an individual negative or positive triplet's score, the black line is the threshold, which is 300 in this article, and the green line is the accuracy using this threshold. The ''simalar-to'' relation only has two triplets in the test set, so there is only a single red line and single green line in the chart. The information about triplet number in the training set can be found in Fig.4 and Table 5.

Our model classified triplets with a score less than 300 as positive, so in Fig.3, the score of positive triplets (blue line) decreases as training proceeds, while the score of negative triplets (red line) is increasing. The accuracy (green line) increases as more red lines go above the threshold and blue lines descend under the threshold, which means the model can distinguish the triplets accurately. The red dots in the charts

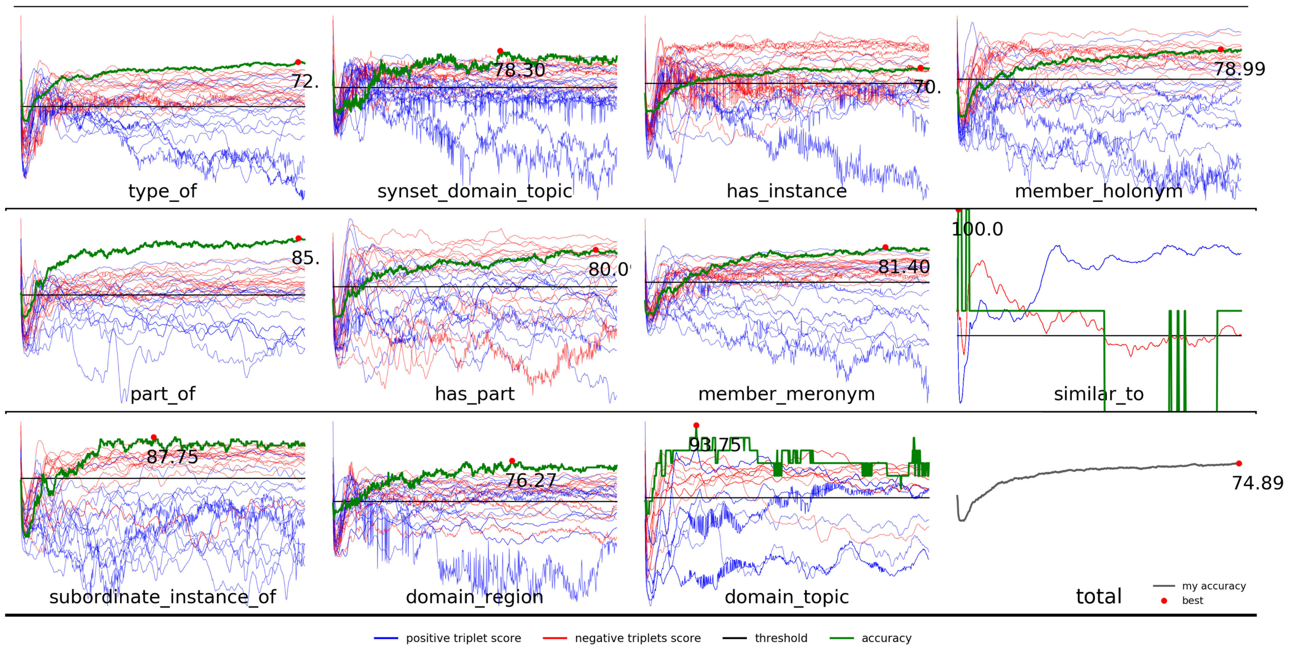


FIGURE 3. Changes in the scores of triplets in the both-1000 dataset during training.

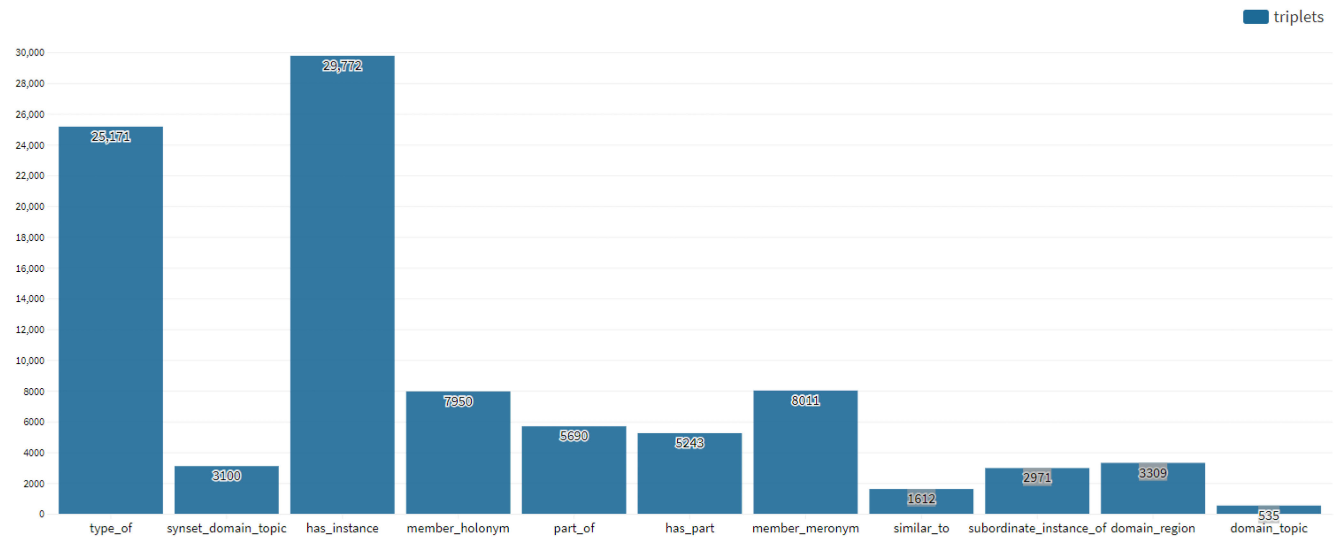


FIGURE 4. Triplet number of a given relation in training set of both-1000.

indicate the highest accuracy achieved during training. The GNN structure successfully obtains the embeddings of the OOKB entities, and the convolution layer guarantees robust training. Furthermore, in the overall performance subchart, the black line continues rising and the red dot always appears in the latter part of training.

Fig.3 shows that most triplets can be optimized towards the correct score, but some triplets are difficult to optimized, that is, the red lines and blue lines are too close to be divided by the black line. This situation is substantial, especially for the "type-of" and "has-instance" relations. Fig.4 shows that these two relations have the largest number of triplets, which is interesting because more data always means better performance in representation learning. Combining Fig.3 and Fig.4, we can also see that the accuracy of these two relations rises steadily, which may indicate that relations with a large number of triplets need more training time. This phenomenon may also be related to the knowledge base dataset. There are some discussions [27], [45] of the available KBC datasets, in which the researchers believe the existing dataset and the evaluation protocol are deficient. In the future, we will attempt to generalize the OOKB problem by applying a newly proposed dataset and evaluation method.

On the basis of the evidence and analysis above, we conclude that the proposed model has advantages in OOKB entity problem. We believe the model's steady convergence in training and high parameter efficiency make it worthy to extend to more KBC scenarios.

VI. CONCLUSION

In this article, we focus on the KBC task in the OOKB entity problem. The OOKB entity problem means that entities are unobserved at training time. We propose a parameter-efficient embedding model that benefits from both a GNN and a CNN to handle the OOKB KBC task. The effectiveness of our model in terms of improved accuracy and parameter efficiency is verified in OOKB datasets. Our model achieved better accuracy with approximately one-fifth of the parameter count in the previous work.

Recently increasing studies have focused on the KBC task in specific scenario. These studies have made KBC task more practical because in real-world cases there are always complicated situations. The OOKB entity problem is one of the common scenarios in real-world cases. Our work is a successful attempt to combine the KBC task with practical scenarios. In the future we would like to extend our model to be scalable to larger datasets and make the model adaptable to different scenarios.

REFERENCES

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

[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*, 2008, pp. 1247–1250.

[3] J. Berant and P. Liang, "Semantic parsing via paraphrasing," in *Proc. 52nd Annu. Meeting Assoc. Comput. Linguistics*, vol. 1, 2014, pp. 1415–1425.

[4] 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*, vol. 1, 2017, pp. 1766–1776.

[5] S. Keizer, M. Guhe, H. Cuayahuitl, I. Efstathiou, K.-P. Engelbrecht, M. Dobre, A. Lascarides, and O. Lemon, "Evaluating persuasion strategies and deep reinforcement learning methods for negotiation dialogue agents," in *Proc. 15th Conf. Eur. Chapter Assoc. Comput. Linguistics*, vol. 2, 2017, pp. 480–484.

[6] 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.

[7] 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. 2787–2795.

[8] M. Nickel, K. Murphy, V. Tresp, and E. Gabrilovich, "A review of relational machine learning for knowledge graphs," *Proc. IEEE*, vol. 104, no. 1, pp. 11–33, Jan. 2016.

[9] 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.

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

[11] 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.*, vol. 1, 2015, pp. 687–696.

[12] G. Ji, K. Liu, S. He, and J. Zhao, "Knowledge graph completion with adaptive sparse transfer matrix," in *Proc. 30th AAAI Conf. Artif. Intell.*, 2016, pp. 985–991.

[13] M. Nickel, V. Tresp, and H.-P. Kriegel, "A three-way model for collective learning on multi-relational data," in *Proc. ICML*, vol. 11, 2011, pp. 809–816.

[14] A. García-Durán, A. Bordes, and N. Usunier, "Effective blending of two and three-way interactions for modeling multi-relational data," in *Proc. Joint Eur. Conf. Mach. Learn. Knowl. Discovery Databases*. Berlin, Germany: Springer, 2014, pp. 434–449, doi: 10.1007/978-3-662-44848-9_28.

[15] M. Nickel, L. Rosasco, and T. Poggio, "Holographic embeddings of knowledge graphs," in *Proc. AAAI*, 2016, vol. 2, no. 1, pp. 2–3.

[16] B. Yang, W.-t. Yih, X. He, J. Gao, and L. Deng, "Embedding entities and relations for learning and inference in knowledge bases," 2014, *arXiv:1412.6575*. [Online]. Available: http://arxiv.org/abs/1412.6575

[17] T. Trouillon, J. Welbl, S. Riedel, E. Gaussier, and G. Bouchard, "Complex embeddings for simple link prediction," in *Proc. Int. Conf. Mach. Learn. (ICML)*, 2016, pp. 1–10.

[18] H. Liu, Y. Wu, and Y. Yang, "Analogical inference for multi-relational embeddings," in *Proc. 34th Int. Conf. Mach. Learn.*, vol. 70. New York, NY, USA: JMLR.org, 2017, pp. 2168–2178.

[19] R. Xie, Z. Liu, and M. Sun, "Representation learning of knowledge graphs with hierarchical types," in *Proc. IJCAI*, 2016, pp. 2965–2971.

[20] 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.

[21] R. Xie, Z. Liu, J. Jia, H. Luan, and M. Sun, "Representation learning of knowledge graphs with entity descriptions," in *Proc. 13th AAAI Conf. Artif. Intell.*, 2016, pp. 1–7.

[22] S. Guo, Q. Wang, L. Wang, B. Wang, and L. Guo, "Jointly embedding knowledge graphs and logical rules," in *Proc. Conf. Empirical Methods Natural Lang. Process.*, 2016, pp. 192–202.

[23] J. Liang, Y. Xiao, H. Wang, Y. Zhang, and W. Wang, "Probase+: Inferring missing links in conceptual taxonomies," *IEEE Trans. Knowl. Data Eng.*, vol. 29, no. 6, pp. 1281–1295, Jun. 2017.

[24] T. N. Kipf and M. Welling, "Semi-supervised classification with graph convolutional networks," 2016, *arXiv:1609.02907*. [Online]. Available: http://arxiv.org/abs/1609.02907

[25] 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.* Springer, 2018, pp. 593–607.

[26] 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.

[27] T. Dettmers, P. Minervini, P. Stenetorp, and S. Riedel, "Convolutional 2D knowledge graph embeddings," in *Proc. 32nd AAAI Conf. Artif. Intell.*, 2018, pp. 1–9.

[28] 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. AAAI*, 2020, pp. 3009–3016.

[29] T. Hamaguchi, H. Oiwa, M. Shimbo, and Y. Matsumoto, "Knowledge transfer for out-of-knowledge-base entities : A graph neural network approach," in *Proc. 26th Int. Joint Conf. Artif. Intell.*, Aug. 2017, pp. 1802–1808.

[30] N. Nakashole and G. Weikum, "Real-time population of knowledge bases: Opportunities and challenges," in *Proc. Joint Workshop Autom. Knowl. Base Construct. Web-Scale Knowl. Extraction (AKBC-WEKEX)*, 2012, pp. 41–45.

[31] N. Nakashole, T. Tylenda, and G. Weikum, "Fine-grained semantic typing of emerging entities," in *Proc. 51st Annu. Meeting Assoc. Comput. Linguistics*, vol. 1, 2013, pp. 1488–1497.

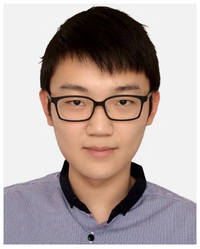
[32] Y. Jin, E. Kıcıman, K. Wang, and R. Loynd, "Entity linking at the tail: Sparse signals, unknown entities, and phrase models," in *Proc. 7th ACM Int. Conf. Web Search Data Mining (WSDM)*, 2014, pp. 453–462.

[33] P. Manchanda, "Entity linking and knowledge discovery in microblogs," in *Proc. ISWC-DC ISWC Doctoral Consortium*, 2015, p. 25.

[34] 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. AAAI Conf. Artif. Intell.*, vol. 33, 2019, pp. 3060–3067.

[35] 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.
[36] J. Baek, D. Bok Lee, and S. Ju Hwang, ''Learning to extrapolate knowledge: Transductive few-shot out-of-graph link prediction,'' 2020, *arXiv:2006.06648*. [Online]. Available: http://arxiv.org/abs/2006.06648
[37] F. Kong, R. Zhang, H. Guo, S. Mensah, Z. Hu, and Y. Mao, ''A neural bag-of-words modelling framework for link prediction in knowledge bases with sparse connectivity,'' in *Proc. World Wide Web Conf. (WWW)*, 2019, pp. 2929–2935.
[38] D. Fernàndez-Cañellas, J. Espadaler, D. Rodriguez, B. Garolera, G. Canet, A. Colom, J. M. Rimmek, X. Giro-i Nieto, E. Bou, and J. C. Riveiro, ''VLX-stories: Building an online event knowledge base with emerging entity detection,'' in *Proc. Int. Semantic Web Conf.* Cham, Switzerland: Springer, 2019, pp. 382–399, doi: 10.1007/978-3-030-30796-7_24.
[39] J. Ding, S. Ma, W. Jia, and M. Guo, ''Jointly modeling structural and textual representation for knowledge graph completion in zero-shot scenario,'' in *Proc. Asia–Pacific Web (APWeb) Web-Age Inf. Manage. (WAIM) Joint Int. Conf. Web Big Data*. Cham, Switzerland: Springer, 2018, pp. 369–384, doi: 10.1007/978-3-319-96890-2_31.
[40] Z. Wang, L. Li, Q. Li, and D. Zeng, ''Multimodal data enhanced representation learning for knowledge graphs,'' in *Proc. Int. Joint Conf. Neural Netw. (IJCNN)*, Jul. 2019, pp. 1–8.
[41] M. Zhao, W. Jia, and Y. Huang, ''Attention-based aggregation graph networks for knowledge graph information transfer,'' in *Proc. Pacific–Asia Conf. Knowl. Discovery Data Mining*. Cham, Switzerland: Springer, 2020, pp. 542–554, doi: 10.1007/978-3-030-47436-2_41.
[42] Y. Li, D. Tarlow, M. Brockschmidt, and R. Zemel, ''Gated graph sequence neural networks,'' 2015, *arXiv:1511.05493*. [Online]. Available: http://arxiv.org/abs/1511.05493
[43] R. Kadlec, O. Bajgar, and J. Kleindienst, ''Knowledge base completion: Baselines strike back,'' in *Proc. 2nd Workshop Represent. Learn. NLP*, 2017, pp. 69–74.
[44] D. P. Kingma and J. Ba, ''Adam: A method for stochastic optimization,'' 2014, *arXiv:1412.6980*. [Online]. Available: http://arxiv.org/abs/1412.6980
[45] Z. Sun, S. Vashishth, S. Sanyal, P. Talukdar, and Y. Yang, ''A re-evaluation of knowledge graph completion methods,'' 2019, *arXiv:1911.03903*. [Online]. Available: http://arxiv.org/abs/1911.03903

ZHONGQIN BI received the Ph.D. degree in system analysis and integration from East China Normal University, China, in 2006. He is currently an Assistant Professor with the College of Computer Science and Technology, Shanghai University of Electric Power, Shanghai, China. He has published more than 30 articles in refereed journals and conference proceedings. His main research interests include cloud computing, data processing, and quality control in smart grids.



TIANCHEN ZHANG is currently pursuing the master's degree with the College of Computer Science and Technology, Shanghai University of Electric Power, Shanghai, China. His research interests include knowledge bases, natural language processing, and representation learning.

PING ZHOU received the M.A.Sc. degree in computer application technology from the University of Shanghai for Science and Technology, China, in 2005. He is currently an Assistant Professor with the College of Computer Science and Technology, Shanghai University of Electric Power. He has published more than ten articles in refereed journals and conference proceedings. His main research interests include artificial intelligence, big data technology, and intelligent voice interaction.

YONGBIN LI received the M.A.Sc. degree in computer application technology from Shanghai Jiaotong University, China, in 2008. He is currently an Assistant Professor with the College of Computer Science and Technology, Shanghai University of Electric Power. He has published more than ten articles in refereed journals and conference proceedings. His main research interests include data process and software engineering. He has won the Third Prize of Anhui Provincial Science and Technology Progress.