Gate Feature Interaction Network for Relation Prediction in Knowledge Graph

Jing Wang[1,2], Shuo Zhang[1,2], Runzhi Li[2†]

[1]School of computer and artificial intelligence, Zhengzhou University, Zhengzhou 450001, China
[2]Cooperative Innovation Center of Internet Healthcare, Zhengzhou University, Zhengzhou 450052, China

Keywords: Knowledge graph; Relation prediction; Gate convolution; Expressive feature; Interaction information

Citation: Wang J., Zhang S., Li R.Z.: Gate Feature Interaction Network for Relation Prediction in Knowledge Graph. Data Intelligence 6(3), 749-770 (2024). doi: 10.3724/2096-7004.di.2024.0023
Submitted: July 11, 2022; Revised: February 19, 2023; Accepted: April 27, 2023

ABSTRACT

Recently, many knowledge graph embedding models for knowledge graph completion have been proposed, ranging from the initial translation-based model such as TransE to recent CNN-based models such as ConvE. These models fill in the missing relations between entities by focusing on capturing the representation features to further complete the existing knowledge graph (KG). However, the above KG-based relation prediction research ignores the interaction information among entities in KG. To solve this problem, this work proposes a novel model called Gate Feature Interaction Network (GFINet) with a weighted loss function that takes the benefit of interaction information and deep expressive features together. Specifically, the proposed GFINet consists of a gate convolution block and an interaction attention module, corresponding to catching deep expressive features and interaction information based on these valid features respectively. Our method establishes state-of-the-art experimental results on the standard datasets for knowledge graph completion. In addition, we make ablation experiments to verify the effectiveness of the gate convolution block and the interaction attention module.

1. INTRODUCTION

Many real-world applications are related to Knowledge Graphs (KGs), consisting of question answering [1, 2], dialogue generation [3, 4], machine reading comprehension [40] and recommendation systems [5, 6, 7]. Typically, KGs are represented in the form of (h, r, t), where h and t stand for the head and tail entities,

† Corresponding author: Runzhi Li (E-mail: rzli@ha.edu.cn; ORCID: 0000-0001-7259-9321).

© 2024 Chinese Academy of Sciences. Published under a Creative Commons Attribution 4.0 International (CC BY 4.0) license.

respectively, and r represents the relation from h to t. However, most KGs (e.g., Freebase [8], DBpedia [9]) suffer from incompleteness [10] which would bring challenges for applications, especially for knowledge-based question answering. Knowledge graph completion [11] solves the above problems by filling in the missing connections.

Knowledge graph embedding is a classical method for knowledge graph completion [12, 13, 14]. They embed components of a KG, including entities and relations, into continuous vector spaces to simplify the manipulation while preserving the inherent interaction of the KGs. TransE [15] is a very classical method in the knowledge representation translation model where entities and relationships are mapped into vector spaces, and the representation of entities and relationships becomes a representation between vectors. As extensions of TransE, there are numerous models for the representation of one-to-many, many-to-one, and many-to-many relationships. For example, TransH [16], TransG [17], TransD [18], RESCAL [19], DistMult [20] and ComplEx [21] and so on. While these methods still lack deep expressive features of learning.

With the development of deep learning architectures, CNN-based methods capture the expressive features for improving performance. DKLR [22] employs continuous bag-of-words and deep convolutional neural models to encode the semantics of entity descriptions. SHALLOM [23] is a shallow neural model which could effectively predict missing triples. It is analogous to C-BOW [24] as both approaches predict relation given two known entities. Although the above methods obtain deep expressive features and better performance than TransE and its extensions, some aspects of these methods are still not considered. For example, CNN-based methods do not consider the structural interaction information between two entities.

To solve the problem above, this work proposes gate feature interaction network (GFINet), which employs weighted binary cross-entropy function to alleviate the imbalance problem on the size of links in a knowledge base. GFINet consists of a gate convolution block and an interaction attention module, which are able to respectively extract deep expressive features and interaction information of entities. The gate convolution block employs a gating mechanism to select salient latent information from the feature level. Inspired by the LSTM gate mechanism [25], we design a self-gated activation function to implement the gating mechanism by filtering out useless features. And we design a shortcut connection between the initial convolution feature of entities and the output of the gate convolution block. This encourages entities to carry both global features and local features, so that we can learn the deep expressive features of entities. The interaction attention module excavates the interaction information based on the deep expressive features. Here, we use two "fully connected" layers to determine the attention coefficient of each interaction feature by sharing the same convolution kernel. Finally, GFINet utilizes both interaction information and deep expressive features to generate knowledge graph embedding and utilizes two common fully connected layers as a decoder to calculate the probability of the missing relations. We make experiments on five datasets: WN18, WN18RR, FB15k, FB15k-237, and YOGO3-10. In this work, Hits@N (N=1, 3, 5) and MRR are evaluation criteria.

Our key contributions are summarized as follows:

- We propose a novel model called gate feature interaction network (GFINet), which catches both deep expressive features and interaction information from knowledge graph with weighted loss function. Meanwhile, GFINet is an end-to-end model.
- We propose a gate convolution block and an interaction attention module for knowledge graph embedding. The former utilizes gating mechanism to filter out useless features and the latter uses attention mechanism to achieve interaction information learning for entity representation.
- We conduct extensive experiments on benchmark datasets, and the results demonstrate the effectiveness and efficiency of our method over other methods. In particular, we make ablation experiments to verify the effectiveness of the gate convolution block and the interaction attention module.

This paper is organized as follows: Section 2 describes knowledge graph embedding models and CNN-based methods to solve the KGC problem. Section 3 details the composition of the whole framework and each of the sub-modules proposed in this paper and conducts a theoretical analysis. Section 4 introduces the experimental details and analyzes the results of typical downstream tasks. At length, section 5 concludes this paper.

2. RELATED WORK

Translation model based method. TransE [15] is the most classical translation-based model for KGC, and it predicts relation r between head entity h and tail entity t. Due to the lower parameter complexity, TransE is only suitable for 1-to-1 relationships, but there are problems in handling 1-to-N, N-to-1, and N-to-N relationship. To alleviate this problem, TransH [16] regards a relation vector r as a translation on a hyperplane with w_r as the normal vector. TransR [26] models entities and relations in two different spaces, namely, entity space and multiple relational spaces, and transforms in the corresponding relational space. PTransE [27] predicts r by considering multiple-step relation paths for representation learning.

CNN model based method. DKLR [22] explore two encoders, including continuous bag-of-words and deep convolutional neural models to encode the semantics of entity descriptions. Then, this model learns knowledge representations with both triples and descriptions, which is a benefit for KGC and entity classification. However, the prediction of r between two different entities is not easy for DKLR. KGML [28] uses a multi-label deep neural network to predict relations given two known entities. KGML embeds entities only; given subject and object are embedded and concatenated to indicate the probability distribution of predicates. Results show predictions by KGML are more accurate than those by TransE, TransR, and DKLR, and its learning speed is faster than PTransE. SHALLOM [23] is analogous to CBOW to predict missing relations given two entities, and the runtime of SHALLOM on benchmark datasets is within a few minutes.

GNN model based method. GNN-based model attempts to use GNN for graph context modeling. These models first aggregate graph context into entity or relation embedding via GNN, then utilize the context-aware embeddings to the predicted scoring function for scoring, RGCN [39] is an extension of graph convolutional network, which can effectively model multi-relational data for link prediction. LightCAKE [38]

considers both entity context and relation context, and introduces no new parameters, making it very lightweight and capable of being used on large-scale KGs. For a vast knowledge base, inherent latent features and interaction feature between entities significantly impact relation prediction. The above methods do not consider both latent features and interaction features, thus in this work, we propose a novel neural network framework with a gate convolution block and interaction attention layer to predict missing relations for knowledge graph completion.

3. OUR APPROACH

To catch deep expressive features and interaction information of entities, we propose GFINet, which reasonably extracts deep expressive features and interaction information of entities from different learning modules. Specifically, GFINet not only catches deep expressive features of given two entities through a gate convolution block, but also catches interaction information of deep expressive features with an interaction attention module. Moreover, considering different categories in the training data set, we employ weighted loss function to train the proposed model. In this section, we will elaborate the proposed GFINet and its two learning module consisting of gate convolution block and interaction attention module. Finally, we describe the specific implementation steps of the weighted loss function.

3.1 Gate Feature Interaction Network

GFINet is an end-to-end model with a weighted loss function for knowledge graph completion with knowledge graph embedding. In this work, we resolve the problem of KGC by predicting the missing relations between two entities. In other word, given the KG, there is being a triple (e_i, r, e_j) where e_i and e_j are known and r is a set of relations that are unknown and need to be predicted. We formulate the relation prediction problem as a multi-label classification problem in this work. Specifically, GFINet consists of a gate convolution block and an interaction attention module to get the representation of entities, and utilizes two fully connected layers to predict the probability for an unknown relation r.

Figure 1 is an overview of GFINet. Given a knowledge graph, the input entities denoted as e_i and e_j will be transformed into embedding vectors of the knowledge graph. The outputs are the probability for one or multi-relation between the input entity pair. First, we employ the embedding and convolution layers to preprocess the entity pairs for the entity embedding vector. Then we design a gate convolution block for extracting the primary latent information and an interaction attention module for catching the interaction information of each interaction feature. Next, two fully connected layers compute probabilities on the missing relations. Moreover, considering the different categories of training data set, we use the weighted categorical cross-entropy loss function to solve the imbalance problem, in which the values of the weight coefficients are related to the size per category. In the following, we will introduce the gate convolution block and how to catch interaction information through the interaction attention module, and how to alleviate the imbalance problem with the weighted loss function.

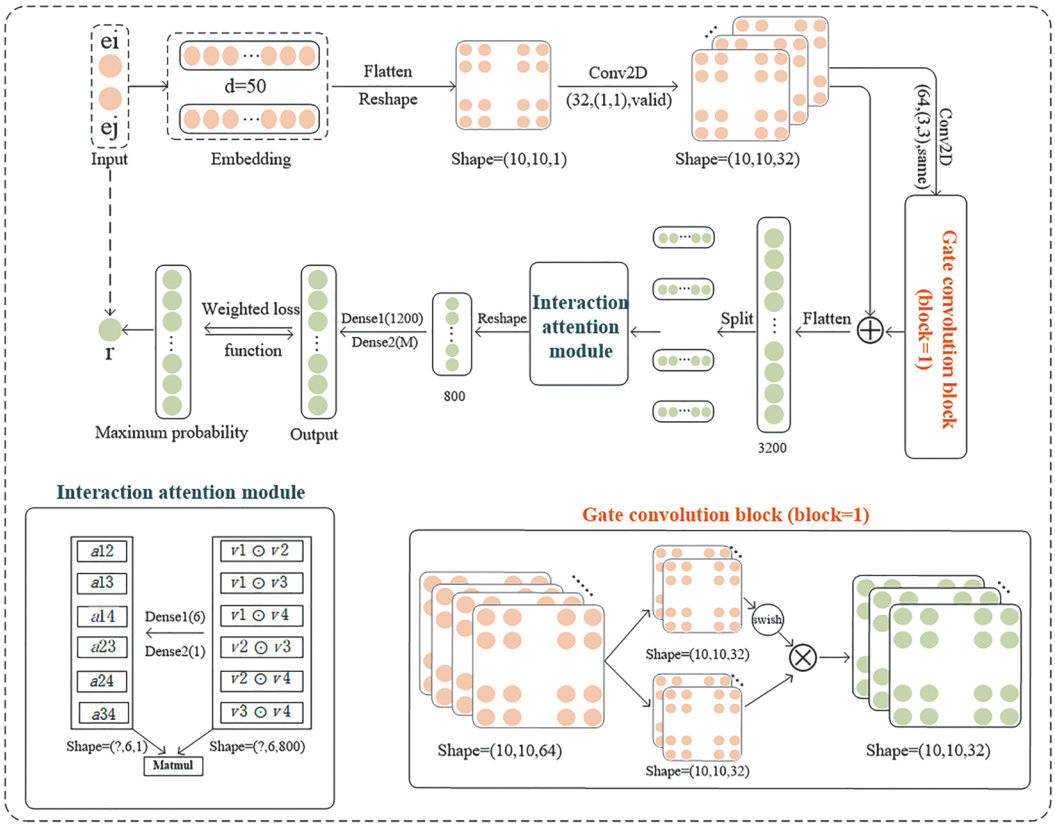


Figure 1. The framework diagram of our proposed model. Input is two known entities (e_i, e_j), the model is to predict the relation r between e_i and e_j. It mainly consists of a gate convolution block and interaction attention module enclosed with dashed frames and two fully connected layers.

3.2 Gate Convolution Block

This section introduces how to catch deep expressive features with the gating mechanism for a knowledge graph through a gate convolution block. First, we will make preprocessing operations for input entities as follows:

Given two different entities e_i and e_j, we design the embedding layer and Flatten, Reshape, and Conv2D operations to represent the structural features. Embedding layer is implemented through the input function in keras, and the input vector is generated given e_i and e_j. The function of Flatten and Reshape operations is used to convolution the input vector. The procedures are shown as following (1) to (3).

$$v_{input} = Input\left(e_i, e_j\right) \tag{1}$$

$$v_{reshape} = \text{Reshape}\left(Flatten\left(v_{input}\right)\right) \tag{2}$$

$$v_{1\times1(32)} = \text{SELU}\left(v_{reshape} * w_{1\times1(32)}\right) \tag{3}$$

Firstly, we use 1×1 convolutions to extend the dimensions of the feature channel, where SELU is a non-linear activation function with saturation region, $*$ denotes the convolutional operation and $w_{1\times1(32)}$ is the parameter of convolution filters with 1×1 kernel size, $v_{1\times1(32)}$ is the extended features of the first 1×1 convolutional layer.

Then, we send the initial convolutional features $v_{1\times1(32)}$ into the gate convolution block. The object of Gate Convolution Block is to filter out useless features. And we also design a shortcut connection between the initial convolution feature and the output of the gate convolution block, where the initial convolution feature is local feature, and the features that are handled with gate convolution block are global features, that is, the output of gate convolution block is local features and the designation of shortcut connection is to capture the global features. Thus, the output is v_{final_gate} consisting of local and global features. Specifically, based on the initial convolution feature, we use a 3×3 convolutional kernel to capture high-order interaction features to a certain degree. Then "slice" operation divides this high-order interaction features into a linear output and a gate output. The linear output saves original features. Inspired by the sigmoid function for gating in LSTM and Highway Network, we use the "swish" activation function as gate output to filter out useless features. They are listed in (4) to (6)

$$v_{3\times3(64)} = \text{Selu}\left(v_{1\times1(32)} * w_{3\times3(64)}\right) \tag{4}$$

$$linear_output = l_out = v_{3\times3(64)}[:,\ :,\ :,\ :32] \tag{5}$$

$$gate_output = \text{g_out} = v_{3\times3(64)}[:,\ :,\ :,\ 32:] \tag{6}$$

Next, we compute the gate convolution features v_{gate_conv} as (7) and (8).

$$v_{gate_conv} = l_out \otimes swish\left(g_out\right) \tag{7}$$

$$\text{swish}(x) = x \cdot \sigma(x) = x \cdot \left(1 + \exp(-x)\right)^{-1} \tag{8}$$

where $\otimes$ is the element-wise product between linear output and gate output with the "swish" activation function. And $\sigma(x)$ is the sigmoid function. The gating mechanism would extract latent information from the feature level. Meanwhile, to retain global features, we design a shortcut connection between the initial convolution feature $v_{1\times1(32)}$ and the output of the gate convolution block. Equation (9) describes the "add" operation.

$$v_{final_gate} = add\left(v_{1\times1(32)},\ v_{gate_conv}\right) \tag{9}$$

3.3 Interaction Attention Module

This section introduces how to catch interaction information based on deep expressive features by the interaction attention module. The interaction attention module consists of the interaction module and

attention mechanism. First, we will introduce how the interaction module works and then for attention mechanism.

The interaction module captures the interaction features of entities. The input v_{input_inter} is from the output of the gate convolution block represented by v_{final_gate}. Specifically, we make Flatten and split operation for v_{final_gate}. We split v_{final_gate} into four independent vectors represented as $V=(v_1, v_2, v_3, v_4)$, and we make a stack and transpose operation to obtain v_{input_inter}. Stack operation is implemented by the tf.stack() which is a matrix splicing function in tensorflow. Transpose operation is to transpose the matrix, this operation is implemented by the tf.transpose() that is also a transpose function in tensorflow. They are listed in (10) to (12).

$$v_{flatten_gate} = Flatten\left(v_{final_gate}\right) \tag{10}$$

$$v_1, v_2, v_3, v_4 = split\left(v_{flatten_gate}, 4, axis=1\right) \tag{11}$$

$$v_{input_inter} = transpose\left(tf.stack\left(\left[V\right]\right), \left[1, 0, 2\right]\right) \tag{12}$$

Then, the four vectors are transformed into six groups of interacting vectors according to the idea of permutation and combination. This interaction vector is the product of the elements of two different feature vectors to encode their interaction. We can formulate them as $V_{dot}=(v_1\odot v_2, v_1\odot v_3, v_1\odot v_4, v_2\odot v_3, v_2\odot v_4, v_3\odot v_4)$. Finally, we make stack and transpose to capture the output of the interaction module, that is v_{inter}, shown in (13) and (14). Equation (14) represents we make transpose operation for v_{inter} that is a tensor of three dimensions. Each number in [1, 0, 2] corresponds to the corresponding dimension, and "[1, 0, 2]" means exchanging the third dimension and the first dimension of the v_{inter}.

$$v_{inter} = stack\left(\left[V_{dot}\right]\right) \tag{13}$$

$$v_{inter} = transpose\left(v_{inter}, \left[1, 0, 2\right]\right) \tag{14}$$

Considering the different contributions of each set of interactive features, we apply attention mechanism into output of the interaction module, where we utilize two fully connected layers to determine the attention coefficient of each set of interactive features. As shown in Figure 2, instead of Initialization weight and "softmax" operation, this attention method directly learns information from each set of feature vectors through convolution operation to allocate different attention coefficients more reasonably.

$$v_{dense1} = relu\left(dot\left(v_{inter}, kernel1\right) + bias1\right) \tag{15}$$

$$v_{dense2} = dot\left(v_{dense1}, kernel2\right) + bias2 \tag{16}$$

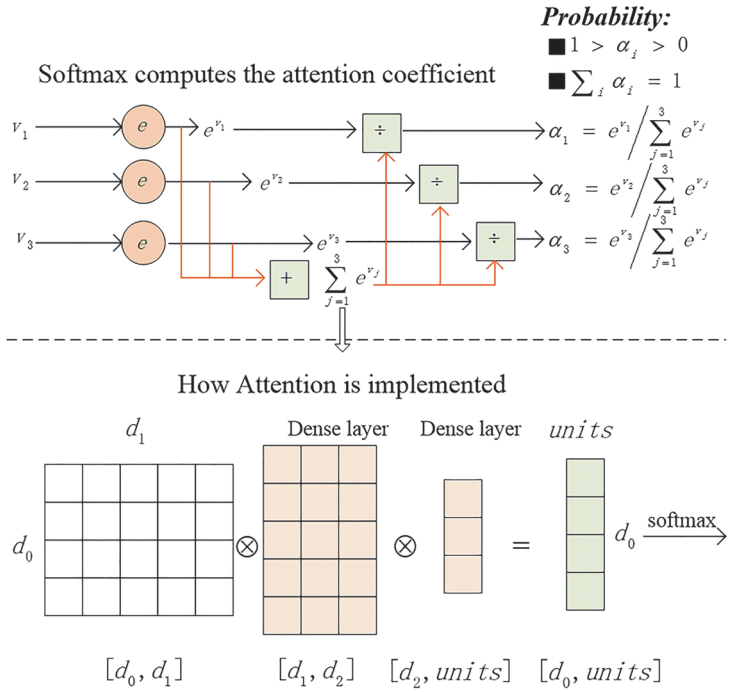


Figure 2. Two different ways to implement the attention mechanism.

This work further employs a "softmax" operation for output vector $v_{dense\,2}$. The final output of the interaction attention module is a weighted vector v_{inter_att}, as in (18). This weighted vector compresses feature interactions in the embedded space by distinguishing the importance of each dimension of interactive features, which reduces the computation of parameters to a certain extent and increases the inherent hidden features between the head and tail entities.

$$\alpha_{score} = transpose\left(softmax\left(v_{dense\,2}\right),\ [0,\ 2,\ 1]\right) \tag{17}$$

$$v_{inter_att} = matmul\left(\alpha_{score},\ v_{inter}\right) \tag{18}$$

After the gate convolution block and the interaction attention module, GFINet catches both deep expressive features and interaction information for knowledge graph embedding and uses two fully connected layers to obtain the probability of missing relation. These two fully connected layers solve the nonlinear problems, so as to achieve multiple classifications. The first fully connected layer uses the LeakyReLU activation function and uses L2 regularization. In this work, hyper-parameter α in LeakyReLU is 0.1. L2 regularization coefficient is selected as 0.1. The softmax activation function maps the original input value to a value in the range of 0 to 1 in the second fully connected layer.

3.4 Weighted Loss Function

This section introduces that we use weighted loss function to train the GFINet. First, we will introduce the reason for using weighted loss function and then the computation procedure of this loss function and weight coefficient of each category.

Considering the different categories of training data set, we use the weighted categorical cross-entropy loss function to solve the imbalance problem. Given different categories' weights $w_i\ (i = 1,\ 2,\ \cdots,\ M)$, M is the number of categories in the training data set, we need to get the related weight coefficients. In this work, we follow the hypothesis that small samples result in more contribution to classification results and vice versa. We have a hypothesis that y_t is the real category label, y_p is the predicted category label and $class_w$ denotes the allocated weight of each category. First, we take a dot product between the category weight and one-hot for y_t to determine the weights. Python's broadcast mechanism is used here. It will extend $class_w$ into the same dimension as y_t, we will obtain the weights of the M samples. The specific implementation process is (19) to (21).

$$\text{loss} = \text{sum}\left(\text{w}\left(y_t * \log y_p\right) + (1-\text{w})\left((1-y_t) * \left(1-\log y_p\right)\right)\right) \tag{19}$$

$$\log y_p = \log\left(\text{sigmoid}\left(y_p\right)\right) \tag{20}$$

$$class_w = \text{w} = \left(\left[w_1,\ \ldots w_i,\ \cdots w_M\right]\right),\ w_i = \frac{n_x}{n_i} \tag{21}$$

where w_i represents the weight coefficient of each category, $\log y_p$ is value after log operation on the predicted relationship probability.

We also describe the computational logic of weight coefficient: there are M classes of training dataset, the number of samples for each class is n_i, i is from 1 to M. We utilize median balance to calculate the w_i. In detail, we calculate the median of these M sample numbers and denoted it as n_x, all of n_i divided by n_x is a new set of coefficients, the inverse of this set of coefficients gives the weight coefficients of the corresponding categories. Finally, we select the highest probability value as the classification result of the input entities, which is the predicted relationship.

4. EXPERIMENTS

4.1 Datasets

In this work, we deploy experiments on five widely used public benchmark datasets: WN18RR [12], WN18 [15], FB15k [8], FB15k-237 [8], and YAGO3-10 [29]. FB15k and WN18 are subsets of Freebase and WordNet, respectively. FB15k-237 is a subset of FB15k with the inverse relationship removed, and WN18RR is the subset of WN18 with the inverse relationship removed. The main schema on these two

datasets is symmetric/antisymmetric and combinatorial. YAGO3-10 is a subset of YAGO, a huge semantic knowledge base. Table 1 gives the summary statistics of the five datasets.

Table 1. Statistics of the five datasets.

Datasets	entity	relation	train_data	valid_data	test_data
FB15k	14951	1345	483142	50000	59071
WN18	40943	18	141442	5000	5000
FB15k-237	14541	237	272115	17535	20466
WN18RR	40943	11	86835	3034	3134
YAGO3-10	123182	37	1079040	5000	5000

4.2 Evaluation Metrics

We convert each triple (e_i, r, e_j) in the test set into a query and compute ranking-based evaluation metrics. The method takes e_i, e_j as input and output a list of candidate relations $R = \left[r^1, \cdots r^L\right]$ ranked in decreasing order of confidence score. We compute $rank_r$, the rank of r among R, after removing the other correct answers from R and use it to compute two types of metrics: Hits@N (N=1, 3, 5) which is the percentage of examples where $rank_r \leq N$ and mean reciprocal rank (MRR). For each query q_i, we rank the first relevant result at position k_i (rank is k_i), then denote the score of Reciprocal Rank (RR) as $\frac{1}{k_i}$, and average the RR of all queries, namely MRR, as in (22).

$$\mathrm{MRR} = \frac{1}{N}\sum_i \frac{1}{k_i} \tag{22}$$

4.3 Baselines

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

- Non-neural: Methods that use translation distance based or semantic matching based score functions are given for FB15k, FB15k-237, WN18, and WN18RR. For instance, TransE [15], TransR [26], RESCAL [19], ComplEx [21].
- Neural: Methods that leverage convolutional neural network are given to predict relation. CNN-based methods includes DKLR (CNN) [22], KGML [28], RDFDNN [33], RGCN [39], SHALLOM [23], LightCAKE [38] and Qiao2021 [37].

To validate the effectiveness of each component in GFINet, we designed two different variants for the ablation study. We also designed another two variants for the ablation study to further validate whether we need to stack the gate convolution block. The details are as follows:

- -GA removes gate convolution block on the basis of GFINet (our model) for relation prediction.
- -IA removes interaction attention operation by comparison with GFINet.
- GA_2 stacks two gate convolution blocks on the basis of our model to predict relation.
- GA_3 uses three gate convolution blocks to learn the latent effective features.

4.4 Experimental Settings

We implement our model using Keras platform and python. The codes run on an Intel Corel I7 and a GeFore GTX 1080 Ti Graphics Cards in Ubuntu 18.04 environment. We design the contrast experiments among our model and baseline models on different datasets which are shown in table 2, table 3 and table 4. Relation prediction analysis on each relation is given in table 5 and table 6 in this manuscript to verity the effectiveness of the proposed model. To verify the effect of the proposed gate convolution block and the interaction attention module, we deploy ablation analysis illustrated in table 7. Moreover, we also design experiments of parameter optimization, the results correspond to figure 3, figure 4, table 9 and table 10. To demonstrate the robustness of our model without overfitting, the training and validation accuracy/loss curves on different datasets are visualized in figure 5. We use grid search to optimize the hyper-parameters of our proposed model. The final parameters of GFINet are determined according to the Hits@1 on the validation set of each dataset. The hyper-parameter ranges for the grid search were set as follows: embedding size d=[8, 18, 32, 50], batch_size=[128, 256, 512, 1000], and L2-regularization=[0.001, 0.01, 0.1, 0.2, 0.5]. The hyper-parameters we find work well are as follows: for FB15k-237, FB15k, WN18RR, and WN18, the embedding size is 50, the batch size is 128 and L2-regularization is 0.1, and the number of the first fully connected layer is 1200. For YAGO3-10, the embedding size is 8, the number of the first fully connected layer is 200 and the rest remains the same.

4.5 Results and Analysis

Performance comparison with state-of-the-art methods. Our results for the relation prediction task are shown in table 2, table 3, and table 4. The bold results in each column are the-state-of-art and on five large benchmark datasets, we can see our model outperforms other methods on WN18, FN15k-237, WN18RR, and YAGO3-10. We also have a competitive result on FB15k. Especially, GFINet achieves a great improvement compared with translation-based models and outperforms CNN-based models.

Specifically, Table 2 gives the Hits@1 relation prediction on FB15k and WN18, our model achieves 98.4 accuracies on Hits1 for the WN18 dataset. For FB15k, our model outperforms many state-of-the-art approaches and achieves 73.8 accuracies. Table 3 shows the result of Hits@N (N=1, 3, 5) relation prediction on WN18RR and FB15k-237. Our model achieves the best accuracy on Hits@1, Hits@3, and Hits@5 respectively, for FB15k-237. For WN18RR, this model also owns 91.4 and 98.2 accuracy on

Hits@1 and Hits@3 respectively. We notice that the results on FB15k is not very outstanding. It has 1345 relations with 483142 training factual triplets. Although there exist enough training data, many of relations do not own enough training sample, and we will overcome this drawback in future research.

Table 2. Relation prediction results on FB15k and WN18. Here we only list the accuracy on Hits@1.

Method	FB15k	WN18
TransE [15]	.651	.736
TransR [26]	.702	.713
ProjE-listwise [10]	**.758**	-
PTransE (ADD, len-2 path) [27]	.695	-
DKLR (CNN) [22]	.698	-
TKRL (RHE) [32]	.711	-
RDFDNN [33]	.691	.770
KGML [28]	.725	.975
SSP [34]	.709	-
SHALLOM [23]	.734	.970
RelationPF [35]	.726	-
PERL [36]	.704	-
Our model	0.738	**.984**

Table 3. The mean of Hits@N (N is 1, 3, and 5) relation prediction on WN18RR and FB15k-237.

Datasets	WN18RR			FB15k-237		
Metric	Hits@1	Hits@3	Hits@5	Hits@1	Hits@3	Hits@5
RESCAL	.331	.529	.734	.115	.327	.456
TransE	.507	.761	.864	.774	.899	.918
ComplEx	.515	.652	.758	.153	.300	.378
CP	.332	.518	.659	.467	.609	.675
DistMult	.497	.677	.799	.092	.176	.428
KGML	.868	.954	.975	.921	.960	.976
RDFDNN	.819	.967	.985	.913	.934	.953

Table 3. *Continued.*

Datasets	WN18RR			FB15k-237		
Metric	Hits@1	Hits@3	Hits@5	Hits@1	Hits@3	Hits@5
$RDF2Vec_{Skip\text{-}Gram}$	.534	.815	.940	.518	.600	.677
$RDF2Vec_{CBOW}$	.451	.785	.932	.522	.608	.687
URC	.095	.265	.446	-	.013	.020
RGCN	-	.854	**-**	-	.973	-
SHALLOM	.874	.982	**.995**	.948	.993	.997
Qiao2021	-	**.989**	**-**	-	.989	-
LightCAKE	-	.988	**-**	-	.988	-
Our model	**.914**	.982	.994	**.957**	**.994**	**.997**

Table 4. The mean of Hits@N (N is 1, 3, and 5) relation prediction on YAGO3-10.

Datasets	YAGO3-10		
Metric	Hits@1	Hits@3	Hits@5
RDF2VecSkip-Gram	.487	.796	.875
$RDF2Vec_{CBOW}$	.491	.803	.873
SHALLOM	.630	.983	.996
Our model	.656	.992	.996

Table 4 lists the Hits@N relation prediction on YAGO3-10, a subset of YAGO, which is a huge semantic knowledge base. Our model reaches 65.6 accuracies on Hits@1, 2.6 point higher than SHALLOM. Looking at all results of these three tables, this model obtains better or comparable performance on Hits@1 on huge semantic knowledge base, such as FB15k and YAGO3-10. However, the results still leave a lot of room for improvement, this is because out model still does not excavate more powerful effective features from their million-level training datasets. We will investigate knowledge graph completion of this huge knowledge base with millions of training triples in future work.

Effects of Different Hyper-parameters. Hyper-parameters and optimizer are very important for the performance of our proposed model. As is shown in Figure 3 and Figure 4, when we give different selections on the unit of the first fully connected layer, and the factor of L2 regularization for FB15K-237, GFINet will obtain different mean test score. We can see that when the factor of L2 regularization is 0.1 and the units of first fully connection layer is 1300, the mean test score on FB15K-237 reach the peak.

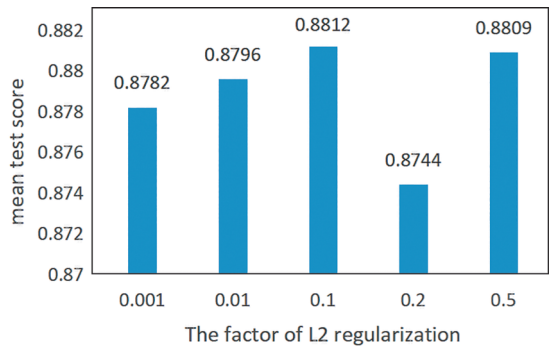


Figure 3. Result analysis on the factor of L2 regularization.

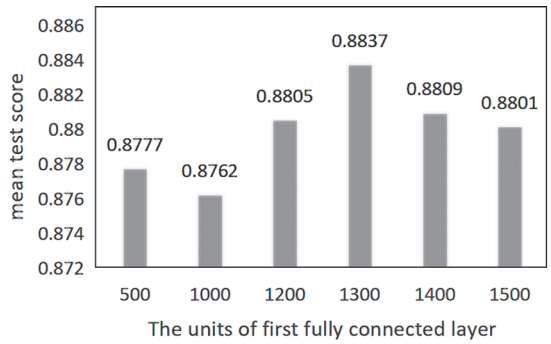


Figure 4. Result analysis on the units of first fully connection layer.

Performance Result on Each Relation of WN18 and WN18RR. Considering the number of relationship categories in the given data set, we illustrate relation prediction analysis on each relation for WN18 with 18 relations and WN18RR with 11 links, shown in Table 5 and Table 6. Although two datasets have the same connection, the result on the same relation is different. Different sizes of the training datasets may cause this. For example, for link "hypernym" in WN18 and WN18RR, our model captures higher accuracy with 99.6 on Hits@1 for WN18. We find that "hypernym" appears 184 times more in WN18 than WN18RR. In addition, when we analyze Table 5 and Table 6, we can obtain specific accuracy on Hits@N (N is 1, 3, and 5) and MRR on each relation and index the number of occurrences per relation. And then we notice that it is not true that the more often, the better the accuracy.

Validation on Model Robustness. To demonstrate the robustness of GFINet without overfitting, we evaluate our model on both training dataset and validation dataset. Figure 5 exhibits training and validation accuracy and training and validation loss on FB15k-237, FB15k, WN18RR, and WN18. The training loss and validation loss show the same trend from the loss curves, so do the accuracy curves. This phenomenon demonstrates there does not exist over-fitting in our model. Moreover, our model can achieve convergence quickly, especially for FB15k, due to the considerable training dataset.

Table 5. The mean of Hits@N relation prediction and MRR on WN18.

Relation	Hits@1	Hits@3	Hits@5	MRR	occurrences
_member_of_domain_usage	1.000	1.000	1.000	1.000	24
_verb_group:	0.949	1.000	1.000	0.974	39
_hyponym	0.993	0.999	1.000	0.996	1153
_member_of_domain_region	1.000	1.000	1.000	1.000	26
_member_meronym	0.980	0.992	1.000	0.988	253
_part_of	0.970	0.988	0.988	0.979	165
_hypernym	0.996	0.999	1.000	0.998	1251
_has_part	0.983	0.988	0.988	0.987	172
_derivationally_related_form	0.980	0.999	1.000	0.990	1074
_member_holonym	0.978	0.996	0.996	0.986	278
_also_see	0.732	0.857	0.946	0.807	56
_instance_hypernym	0.984	1.000	1.000	0.990	122
_synset_domain_region_of	1.000	1.000	1.000	1.000	37
_synset_domain_topic_of	0.982	1.000	1.000	0.990	114
_member_of_domain_topic	0.955	1.000	1.000	0.974	111
_instance_hyponym	0.991	0.991	1.000	0.993	108
_synset_domain_usage_of	1.000	1.000	1.000	1.000	14
_similar_to	1.000	1.000	1.000	1.000	3

Table 6. The mean of Hits@N relation prediction and MRR on WN18RR.

Relation	Hits@1	Hits@3	Hits@5	MRR	occurrences
_member_of_domain_usage	0.909	0.955	0.955	0.930	22
_verb_group:	0.974	1.000	1.000	0.987	39
_member_of_domain_region	0.808	0.923	0.962	0.879	26
_member_meronym	0.894	0.967	1.000	0.934	246
_hypernym	0.843	0.972	0.996	0.908	1067
_has_part	0.704	0.929	0.994	0.819	169

Table 6. *Continued.*

Relation	Hits@1	Hits@3	Hits@5	MRR	occurrences
_derivationally_related_form	0.978	0.998	1.000	0.988	1074
_also_see	0.804	0.946	0.964	0.881	56
_instance_hypernym	0.920	0.964	0.991	0.948	112
_synset_domain_topic_of	0.873	0.918	0.973	0.911	110
_similar_to	1.000	1.000	1.000	1.000	3

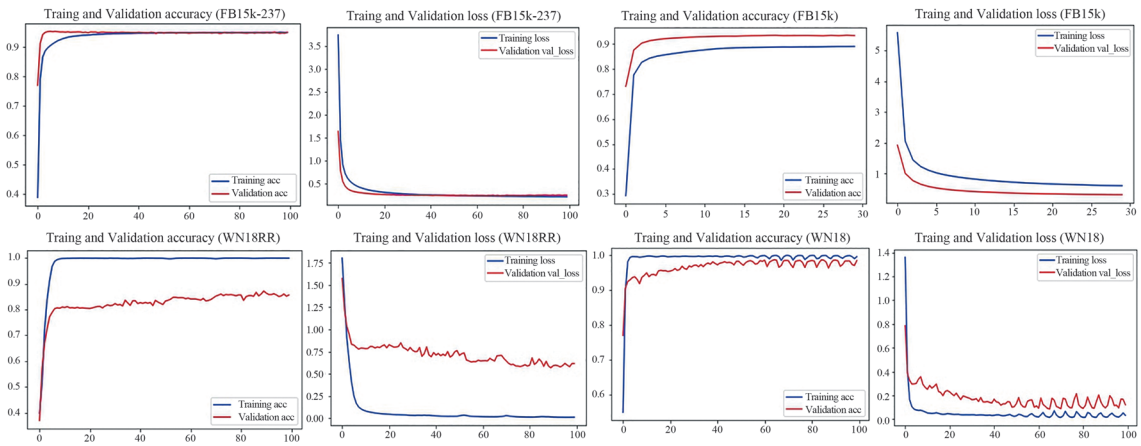


Figure 5. Visualization of training and validation accuracy(loss) curves on different datasets.

4.6 Ablation Study

In this section, we discuss the effects of three key factors of our proposed approach. i.e., the gate convolution block (GA), the interaction attention module (IA) and the embedding size of embedding vectors. All of the ablation studies are tested on FB15k-237 and WN18RR.

Effect of GA and IA. GA block and IA module are vital for deep expressive features and interaction information of entities. To verify the effect of GA block and IA module, we remove GA block and IA module from our model. As shown in Table 7, we show the results of Hits@1 of our model and our model without GA block and IA module on FB15k-237, FB15k, WN18RR, and WN18. In this ablation study, we find in GFINet, when removing the GA block and the IA module, there exist a performance drop on Hits@1 score on four datasets. Specifically, the gate convolution block yields a significant result, especially for FB15k and WN18, which implies gate convolution block could excavate inherent latent information on a vast knowledge base. Interaction attention brings shows a better lift on WN18RR and WN18 datasets. The reason is that both of them are subsets extracted from WordNet carrying out more semantic information, and interaction between entities becomes vital for relation prediction.

Table 7. Relation prediction of our model and our model without GA or IA module.

Datasets	Our model	-GA	-IA
FB15k-237	95.7	90.5	94.0
FB15k	73.8	39.0	69.2
WN18RR	91.4	83.8	77.3
WN18	98.4	58.3	88.0

Effect of Different GA Blocks. GA Block is important to catch deep expressive features of given entities. However, stacking this gate convolution block is not an excellent choice, considering the computational efficiency and training time. Thus, we only use a gate convolution block in our model and we experiment to validate this rationality. GA_2 and GA_3 in Table 8 indicate that our model stacks two gate convolution blocks and three gate convolution blocks, respectively. The results show that more blocks could bring extra training time and reduce the accuracy.

Table 8. Relation prediction and train time of our model, GA_2, and GA_3.

Datasets	Our model	GA_2	GA_3
FB15k-237	95.7 (86s)	94.6 (243s)	94.5 (285s)
WN18RR	91.4 (62s)	84.8 (79s)	86.5 (96s)

Effects of Different Embedding and Split Size. Embedding and split size is very important for representation learning in this work. A small size of embeddings will cause limited representation for knowledge graph, and a large size of embeddings will get a sparse representation to waste storage space and sometimes even have worse result. We test our model with four different embedding sizes which are 50, 32, 18, and 8. As shown in Table 9 and Table 10, we show the results of Hits@1 on WN18RR and FB15k-237. In this ablation study, we find in GFINet, when the split size is 4, the larger embedding size will get a better Hits@1 score. For the split size, the number of partitions is 2 for YAGO3-10, and other datasets are 4. We list hits@1 on different combinations for WN18RR and FB15k-237 shown in Table 9

Table 9. The mean of Hits@1 relation prediction on WN18RR.

Split	Embedding_size=50	Embedding_size=32	Embedding_size=18	Embedding_size=8
Split=2	89.9	91.2	90.2	89.9
Split=4	**91.4**	89.5	89.9	88.1
Split=8	89.0	90.1	89.6	89.5

and Table 10. We can see that when the embedding size is 50, and the split is 2, the model reaches 91.4 accuracies on Hits@1 for WN18RR and also obtains the best accuracy with 95.7 for FB15k-237.

Table 10. The mean of Hits@1 relation prediction on FB15k-237.

Split	Embedding_size=50	Embedding_size=32	Embedding_size=18	Embedding_size=8
Split=2	94.1	93.8	94.3	94.0
Split=4	**95.7**	94.0	93.9	93.7
Split=8	94.2	93.8	93.7	93.6

5. CONCLUSION

In this work, we propose a model called gate feature interaction network (GFINet) to handle the missing relation of knowledge graph, which consists of two encoders and one decoder. Encoders are a gate convolution block and a interaction attention block, where the first one's input is given entities from knowledge graph to learn deep expressive features with gating mechanism, and the second one's input is the combined feature fusing the output of the gate convolution block with the initial convolution feature. Specifically, the gate convolution block utilizes gating mechanism to filter out useless features. The interaction attention module computes the interaction feature vector with a reasonable attention coefficient, which could better extract interaction information between the entity pairs. In addition, considering the different categories of training samples, GFINet employs weighted binary cross-entropy loss function to alleviate imbalance problem. Comprehensive experiments have been conducted on five benchmark datasets and results show that the proposed GFINet establishes the new state-of-the-art or comparable performance on these standard datasets.

In our work, we mainly focus on extracting inherent features and structural information of given entities. However, entity similarity is vital for relation prediction, we will use the entity similarity technique [41] to improve the performance of relation prediction in future work. Moreover, we applied our model into the bio-medical knowledge graph to excavate latent relation (such as adverse drug-drug interaction) for bio-medical knowledge graph completion, which avoids morbidity and even death of patients. Thus, we will consider context-based method [42] or ChatGPT [43] to build a more comprehensive bio-medical knowledge graph, which can obtain more training data and improve the accuracy of model. In conclusion, the combination of entity similarity and knowledge graph completion is vital to obtain high quality data, which is worth exploring by more researchers.

ACKNOWLEDGEMENTS

This work was supported in part by the Science and Technology Innovation 2030- "New Generation of Artificial Intelligence" Major Project under Grant No.2021ZD0111000 and in part by the Henan Province Science and Technology Research Project (232102311232).

AUTHOR CONTRIBUTIONS

Jing Wang: Conceptualization, Methodology, Validation, Investigation, Resources, Visualization, Writing-original draft.

Shuo Zhang: Software, Data curation, Validation, Formal analysis.

Runzhi Li: Supervision, Project administration, Writing-review & editing.

REFERENCES

[1] Yin, J., Zhao, W. X., Li, X. M.: Type-aware question answering over knowledge base with attention-based tree-structured neural networks. Journal of Computer Science and Technology, 32, 805–813 (2017)
[2] Hu, X., Duan, J., Dang, D.: Natural language question answering over knowledge graph: the marriage of SPARQL query and keyword search. Knowledge and Information Systems, 63(4), 819–844 (2021)
[3] Wu, Y., Zhao, S.: Community answer generation based on knowledge graph. Information Sciences, 545, 132–152 (2021)
[4] He, H., et al.: Learning symmetric collaborative dialogue agents with dynamic knowledge graph embeddings. arxiv preprint arxiv:1704.07130 (2017)
[5] Ammar, N., Shaban-Nejad, A.: Explainable artificial intelligence recommendation system by leveraging the semantics of adverse childhood experiences: proof-of-concept prototype development. JMIR medical informatics, 8(11), e18752 (2020)
[6] Tai, C. Y., et al.: User-centric path reasoning towards explainable recommendation. In: Proceedings of the 44th International ACM SIGIR Conference on Research and Development in Information Retrieval, pp. 879–889 (2021)
[7] Ling, C. Y., et al.: Graph embedding based API graph search and recommendation. Journal of Computer Science and Technology, 34, 993–1006 (2019)
[8] Bollacker, K., et al.: Freebase: a collaboratively created graph database for structuring human knowledge. In: Proceedings of the 2008 ACM SIGMOD international conference on Management of data, pp. 1247–1250 (2008)
[9] Lehmann, J., et al.: Dbpedia–a large-scale, multilingual knowledge base extracted from wikipedia. Semantic web, 6(2), 167–195 (2015)
[10] Nickel, M., et al.: A review of relational machine learning for knowledge graphs. Proceedings of the IEEE, 104(1), 11–33 (2015)
[11] Shi, B., Weninger, T.: Proje: Embedding projection for knowledge graph completion. In: Proceedings of the AAAI Conference on Artificial Intelligence, Vol. 31, No. 1 (2017)
[12] Zhu, J. Z., et al.: Modeling the correlations of relations for knowledge graph embedding. Journal of Computer Science and Technology, 33, 323–334 (2018)
[13] Demir, C., Ngomo, A. C. N.: A physical embedding model for knowledge graphs. In Semantic Technology: 9th Joint International Conference, pp. 192–209 (2020)
[14] Wang, Q., et al.: Knowledge graph embedding: A survey of approaches and applications. IEEE transactions on knowledge and data engineering, 29(12), 2724–2743 (2017)
[15] Bordes, A., et al.: Translating embeddings for modeling multi-relational data. Advances in neural information processing systems, 26 (2013)

[16] Wang, Z., et al.: Knowledge graph embedding by translating on hyperplanes. In: Proceedings of the AAAI conference on artificial intelligence, Vol. 28, No. 1 (2014)
[17] Xiao, H., et al.: Transg: A generative mixture model for knowledge graph embedding. arxiv preprint arxiv:1509.05488 (2015)
[18] Ji, G., et al.: Knowledge graph embedding via dynamic mapping matrix. In: Proceedings of the 53rd annual meeting of the association for computational linguistics and the 7th international joint conference on natural language processing (volume 1: Long papers), pp. 687–696 (2015)
[19] Nickel, M., et al.: A three-way model for collective learning on multi-relational data. In: Proceedings of the 28th International Conference on International Conference on Machine Learning, Vol. 11, No. 10.5555, pp. 3104482–3104584 (2011)
[20] Yang, B., et al.: Embedding entities and relations for learning and inference in knowledge bases. arxiv preprint arxiv:1412.6575 (2014)
[21] Trouillon, T., et al.: G. Complex embeddings for simple link prediction. In the Proceedings of the 33rd International Conference on Machine Learning, pp. 2071–2080 (2016)
[22] Xie, R., et al.: Representation learning of knowledge graphs with entity descriptions. In: Proceedings of the AAAI conference on artificial intelligence, Vol. 30, No. 1 (2016)
[23] Demir, C., et al.: A shallow neural model for relation prediction. In 2021 IEEE 15th International Conference on Semantic Computing (ICSC), pp. 179–182 (2021)
[24] Kim, Y.: Convolutional neural networks for sentence classification. In: Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP), pp. 1746–1751 (2014)
[25] Klambauer, G., et al.: Self-normalizing neural networks. Advances in neural information processing systems, 30 (2017)
[26] Lin, Y., et al.: Learning entity and relation embeddings for knowledge graph completion. In: Proceedings of the AAAI conference on artificial intelligence, Vol. 29, No. 1 (2015)
[27] Lin, Y., et al.: Modeling relation paths for representation learning of knowledge bases. arxiv preprint arxiv:1506.00379 (2015)
[28] Onuki, Y., et al.: Relation prediction in knowledge graph by multi-label deep neural network. Applied Network Science, 4, 1–17 (2019)
[29] Rebele, T., et al.: YAGO: A multilingual knowledge base from wikipedia, wordnet, and geonames. In The Semantic Web–ISWC 2016: 15th International Semantic Web Conference, pp. 177–185 (2016)
[30] Sumathi, B.: Grid search tuning of hyperparameters in random forest classifier for customer feedback sentiment prediction. International Journal of Advanced Computer Science and Applications, 11(9), (2020)
[31] Lee, D., Myung, K.: ADAM: Method for Stochastic Optimization. IEEE International Conference on Consumer Electronics, pp. 434–435 (2017)
[32] Xie, R., et al.: Representation learning of knowledge graphs with hierarchical types. International Joint Conference on Artificial Intelligence, pp. 2965–2971 (2016)
[33] Onuki, Y., et al.: Predicting relations of embedded RDF entities by Deep Neural Network. In International Semantic Web Conference (2017)
[34] Xiao, H., et al.: SSP: semantic space projection for knowledge graph embedding with text descriptions. In: Proceedings of the AAAI conference on artificial intelligence, Vol. 31, No. 1 (2017)
[35] Tagawa, Y., et al.: Relation prediction for unseen-entities using entity-word graphs. In: Proceedings of the Thirteenth Workshop on Graph-Based Methods for Natural Language Processing, pp. 11–16 (2019)
[36] Wang, J., et al.: Relation Path Modeling with Entity Types for Knowledge Graph Completion. In 2021 International Conference on Computer Engineering and Application, pp. 209–215 (2021)

[37] Qiao, Z., et al.: Context-enhanced entity and relation embedding for knowledge graph completion (student abstract). In: Proceedings of the AAAI Conference on Artificial Intelligence, pp. 15871–15872 (2021)
[38] Ning, Z., et al.: Lightcake: A lightweight framework for context-aware knowledge graph embedding. In Pacific-Asia Conference on Knowledge Discovery and Data Mining, pp. 181–193 (2021)
[39] Schlichtkrull, M., et al.: Modeling relational data with graph convolutional networks. In The semantic web: 15th international conference, pp. 593–607 (2018)
[40] Gao, F., et al.: Knowledge graph based mutual attention for machine reading comprehension over anti-terrorism corpus. Data Intelligence, 5(3), 685–706 (2023)
[41] Kilanioti, I., A. Papadopoulos, G.: A knowledge graph-based deep learning framework for efficient content similarity search of Sustainable Development Goals data. Data Intelligence, 5(3), 663–684 (2023)
[42] Ligabue, P. D. M., et al.: Applying a Context-based Method to Build a Knowledge Graph for the Blue Amazon. Data Intelligence, 6(1), 64–103 (2024)
[43] Azaria, A., Azoulay, R., Reches, S.: ChatGPT is a remarkable tool—for experts. Data Intelligence, 6(1), 240–296 (2024)

AUTHOR BIOGRAPHY



Runzhi Li is an associate professor, a doctoral supervisor. She participated in a number of national and provincial scientific research projects, presided over 1 research project of the Provincial Department of Science and Technology, published more than 40 academic papers, and 15 papers were included in EI/SCIE/ISTP. The main research directions: artificial intelligence, medical big data analysis, disease risk prediction, knowledge graph completion, and network behavior analysis.
E-mail: rzli@ha.edu.cn; ORCID: 0000-0001-7259-9321



Jing Wang is currently a doctoral student at the School of Computer and Artificial Intelligence, Zhengzhou University, China. Her research interests mainly include Knowledge graphs and graph neural networks.
E-mail: wangjingzzu@gs.zzu.edu.cn; ORCID: 0000-0003-4955-5819



Shuo Zhang is currently a lecturer at the School of Artificial Intelligence, Henan University, China. The author's research fields are medical big data analysis, disease risk prediction, computer vision, etc.
E-mail: szhang@ha.edu.cn; ORCID: 0000-0002-7427-1933