IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received 18 August 2022, accepted 6 September 2022, date of publication 12 September 2022, date of current version 20 September 2022.

Digital Object Identifier 10.1109/ACCESS.2022.3206037

RESEARCH ARTICLE

A Few-Shot Inductive Link Prediction Model in Knowledge Graphs

RUITING YANG, ZHONGCHENG WEI, YONGJIAN FAN, AND JIJUN ZHAO, (Member, IEEE)
School of Information and Electrical Engineering, Hebei University of Engineering, Handan 056038, China
Hebei Key Laboratory of Security and Protection Information Sensing and Processing, Handan 056038, China

Corresponding author: Zhongcheng Wei (weizhongcheng@hebeu.edu.cn)

This work was supported in part by the Science and Technology Research Project of Higher Education Institutions of Hebei Province under Grant QN2020193 and Grant ZD2020171, in part by the Graduate Demonstration Course Construction Project of Hebei Province under Grant KCJSX2022090 and Grant KCJSX2022091, and in part by the Handan Science and Technology Research and Development Program under Grant 21422031288.

ABSTRACT Link prediction aims to predict the missing facts in knowledge graphs. Most previous work focuses on the transductive link prediction, which cannot predict unknown entities. However, knowledge graphs are evolving in practical scenarios and new entities are constantly added. A graph neural network based on subgraph structure can effectively make predictions on a knowledge graph composed of unknown entities. Based on this method, we propose a new inductive link prediction model MILP, which uses meta-learning to predict unseen entities on few-shot data. Specifically, MILP divides the training data into four tasks according to the relation types and constructs a subgraph structure of each triplet, and then trains each task sequentially through the meta-learning framework which uses graph neural network to score the triplets. Experiments are carried out on the benchmark inductive link prediction datasets, and the results show that in most cases the proposed model achieves better results than the baseline models, proving the effectiveness of MILP.

INDEX TERMS Few-shot learning, graph neural network, knowledge graph completion, link prediction, meta-learning.

I. INTRODUCTION

Knowledge graphs (KGs) are knowledge bases composed of a large number of facts. Representing facts form reality in from of the triplet (*head entity*, *relationship*, *tail entity*) establishing the relationship between entities is the most typical representation method. In recent years, knowledge graphs already played an important role in supporting for information retrieval [1], intelligent question answering [2], [3] recommendation system [4], and other fields related to artificial intelligence [5], [6]. However, most constructed knowledge graphs are incomplete due to the limitations of existing knowledge and extraction algorithms. Actually, even the widely used large-scale knowledge graphs such as Freebase [7] and Wikidata [8] meet missing data. Therefore, how to improve the incompleteness of knowledge graphs is an urgent problem to be solved. At present, a large number of studies focus on link prediction, which completes knowledge graphs by predicting the missing links.

The most widely used link prediction models based on embedding method. This kind of models project entities and relations into continuous low-dimensional vector spaces and learn appropriate vector representations for them by formulating scoring functions and training mechanisms. For example, TransE [9] projects entities and relations into the real-valued vector space, then it lets the vector representation of the head entity in the target triple equal to the vector sum of the tail entity and their relation representation. TransH [10] improves on the basis of TransE, and the projection based on relationship is proposed. The same entity corresponding to different relations is projected to different relationship spaces in different triplets, so it has different vector representations. TransH obtains better prediction results on complex relations. Similar works include DistMult [11], RotatE [12],

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

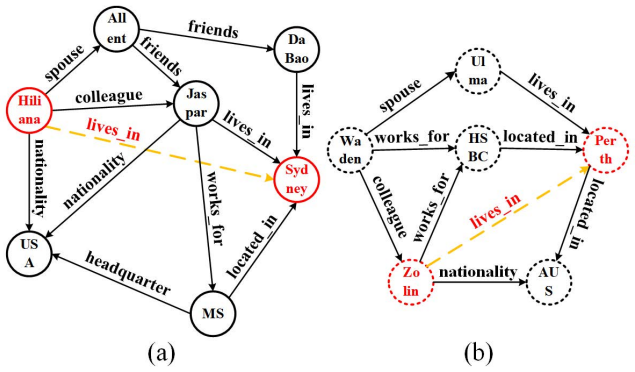

FIGURE 1. Illustration of transductive and inductive link prediction in knowledge graphs.

and DTransE [13], ect. Such models show good performance when predicting seen entities. As shown in Fig. 1, the red entities and relations represent those in target triplets, and the yellow dotted lines represent link prediction on knowledge graphs. The entities in Fig. 1(a) are all seen entities, and the prediction in this scenario is also called transductive link prediction. Relatively, Fig. 1(b) illustrates the inductive link prediction which refers to making prediction on a knowledge graph containing unseen entities, where the dotted nodes represent unseen entities. The embedding-based model can effectively learn the representations of seen entities only, but not the unseen entities, so the unseen entity cannot be predicted after training. However, the data in the real world are constantly changing. For example, new members are often added to social networks, and new products and new users are constantly appearing in recommendation systems. Using this kind of models to make prediction on new entities requires retraining the updated knowledge graph, which is expensive. Therefore, it is of great significance to establish a model suitable for inductive link prediction for real-world applications.

Currently, some research uses external resources to obtain the embedding representations of unseen entities. For example, DEAL [14] uses the attribute information of entities for inductive link prediction. Fu *et al.* [15] design two agents using reinforcement learning. On the one hand, extracting entity-related information from the corpus to enrich knowledge graph, and on the other hand, performing dynamic reasoning. Ali *et al.* [16] use pre-trained language model-Sentence BERT [17] to obtain entity representation from Wikipedia, and the authors also divide inductive link prediction into semi-inductive and fully-inductive scenarios. Although this method can effectively predict inductive links, it spends additional time and resources to train external resources, which are usually not easy to obtain.

Another inductive link prediction method is to learn the logical rules in knowledge graphs. The method first learns the rules appearing in the knowledge graph during training, then it applies the learned rules to prediction. However, the models based on logical rules are inherently poor in expression and are difficult to be generalized. Teru *et al.* [18] regard relational prediction task as a logical induction problem, they use graph neural network (GNN) to predict the relations by modeling the subgraph structures of target triplets and prove the method can also learn logic rules contained in the knowledge graph. In addition, the GNN models using subgraph structures are independent of representations of nodes and has inductive characteristics naturally.

The data in real-word knowledge graphs follow a long-tailed distribution, i.e., most relations and entities have only a few triplets. Taking Wikidata for example, there are around 10% relations have no more than 10 triplets [19], and about 82.6% entities have only one triplet [20]. Especially for the setting of inductive link prediction, the newly added entity has a smaller number of triplets. However, most of the current work assumes that the entities and relations in knowledge graphs have sufficient triplets, such as the GNN models based on the subgraph structure [18], [21], most entities and relations in the data occur more frequently than those in the knowledge graphs from real world, which make the inductive link prediction task easier and their function on few-shot data are limited.

In this paper, we present a **M**eta-learning network based on GNN for **I**nductive **L**ink **P**erdiction (MILP), which can still work effectively on few-shot data. The relations in knowledge graphs can be divided into four types: one-to-one, one-to-many, many-to-one, and many-to-many, and we group triplets with the same relation type to the same task. Therefore, four corresponding tasks are obtained: $\mathcal{T}_1$, $\mathcal{T}_2$, $\mathcal{T}_3$, and $\mathcal{T}_4$. Then the GNN is used as the scorer, and meta-learning is used to train each task in turn to improve the prediction ability of the model, so that the model can obtain more expressive relation representations and initialization parameters. Finally, few-shot samples are extracted from three public datasets: FB15k-237, WN18RR, and NELL-995, and the predictive ability of the model is verified on these inductive link prediction test sets.

The contributions of our work are summarized as follows:

1) We tackle a realistic problem of few-shot data and focus on the link prediction in knowledge graphs, aiming to perform link prediction among unseen entities, where each entity has only few triplets.
2) A meta-learning inductive link prediction model MILP learning subgraph features in knowledge graphs is proposed, which can be generalized to unseen knowledge graphs naturally, and it still validly on few-shot data.
3) Our inductive link prediction experiments on the benchmark datasets show that MILP achieves better AUC, AUC-PR, and Hits@10 in most cases.

The remainder is structured as follows. In Section II, we introduce the classical models related to our work and compare them with ours. The detailed description of our model is shown in Section III. Section IV presents the experimental results. Section V is our conclusion about the paper.

II. RELATED WORK

A. LOGIC RULES BASED

Inductive logic programming (ILP) is a machine learning method for mining relational structure. Starting from specific examples, it summarizes the general rules of these examples and finally learns the first-order logic symbol rules that are easy to understand by human beings. Therefore, the learning process is also a process of rule generalization. But it assumes a large potential space and is difficult to apply to large-scale knowledge bases. The link prediction models based on logical rules mainly learn the rules implied in the knowledge bases. Zeng *et al.* [22] propose QuickFOIL to solve the problem of large hypothetical space by formulating top-down greedy search strategy and pruning candidate sets. QuickFOIL reduces the search space, but only learns rules with high confidence. AMIE [23] proposes partial completeness hypothesis to construct counterexamples, it formulates new confidence measurement standards, and tests rule mining on knowledge bases. RuleN [24] also redefines the calculation method of confidence to learn the rules that help reasoning. AMIE calculates confidence on the whole knowledge graph, but RuleN approximates it by random sampling. RuleN can mine longer path rules and has good effect in inductive link prediction.

TensorLog [25] represents the reasoning of the first-order logic in the form of matrix multiplication and establishes the connection between logical reasoning and differential operation. Yang *et al.* [26] propose an end-to-end differential logical reasoning model Neural-LP based on TensorLog, Neural-LP makes it possible to learn logical rules using gradient-based optimization, and has the ability to reason independently of entity representations. DRUM [27] theoretically proves that the logic rules generated by Neural-LP have limitations, and a bidirectional recurrent neural network is proposed to enhance the information capture ability in the process of rule learning, which shares the information when reasoning.

However, the obvious disadvantage of the method based on logical rules is that it cannot learn enough rules when the knowledge graphs are sparse.

B. GRAPH NEURAL NETWORK BASED

Graph neural networks extend neural networks for processing data represented in graphs [28], [29]. Gilmer *et al.* [30] unify the neural network models on graphs as Message Passing Neural Network (MPNN) and point out the difference between these models lies in defining the message function, aggregation function, and readout function. It provides a new way for the comparison and extension of graph neural network models. RGCN [31] designs a relation-specific aggregation function, which applies different parameter matrices to neighbor nodes connected by different relations, first perform feature aggregation respectively, and then perform information fusion and update nodes. GraphSAGE [32] uses the attribute information of entities to generate the embedding representation of unknown entities, thus achieving the prediction of unknown entities. GraIL [18] proposes a naturally inductive graph neural network message passing paradigm and improves RGCN [31]. The model aggregates the neighbor information of the head entity and the tail entity of a triplet, updating based on their common neighbor information. It consists of three parts: i) extracting subgraphs of target triples; ii) labeling nodes in subgraphs; and iii) scoring triplets through GNN. CoMPILE [21] strengthens the role of reations in message passing and introduces the node-edge message transmission mechanism to update the nodes and edges. TACT [33] models the relational topological structures in the knowledge graph, and then uses the relational correlation module and graph structure module to train. The graph neural network models GraIL and CoMPILE based on subgraph structure have strong inductive learning ability, so the similar method is used to construct and update subgraph information in this paper.

C. META-LEARNING BASED

Meta-learning also called learning to learn, i.e., using different tasks to learn universal experience. Meta-learning for few-shot learning aims to obtain favorable generalization ability through a small amount of training samples and quickly apply to new tasks. There are two main research directions: i) Method based on optimization, which usually used to learn initialization parameters with strong generalization. MAML [34] is a model-agnostic meta-learning model, which can be used for classification, regression, and other gradient-based models without introducing additional neural network parameters. MAML calculates the gradient once within the tasks whereas Reptile [35] updates the parameters using a simpler method saved the computational cost. ii) Method based on metric learning, it tends to extract the features contained in the task samples to the greatest extent and uses feature comparison to determine the types of test samples. For example, Matching Nets [36] uses cosine similarity to measure features in embedding space and achieves classification by calculating the matching degree of the test samples.

The application of meta-learning in link prediction is mostly used to implement few-shot relation learning. Xiong *et al.* [19] propose a one-shot relation learning framework GMatching to address the problem of insufficient training samples for newly added relations. GMatching treats each relation as a meta-learning task and uses a similarity function to measure the similarity between known and unknown triplets under the same relation. It requires that each relation in a query set need to appear at least once in the support set to learn appropriate similarity metric through training, at the same time, it uses entities' one-hop neighbor information to better represent the entities. Wang *et al.* [37] note that rare entities in triples often appear with uncommon relations and formulate it as a few-shot completion problem. Then they collect the text description information of entities and relations from Wikidata and DBpedia to enhance the

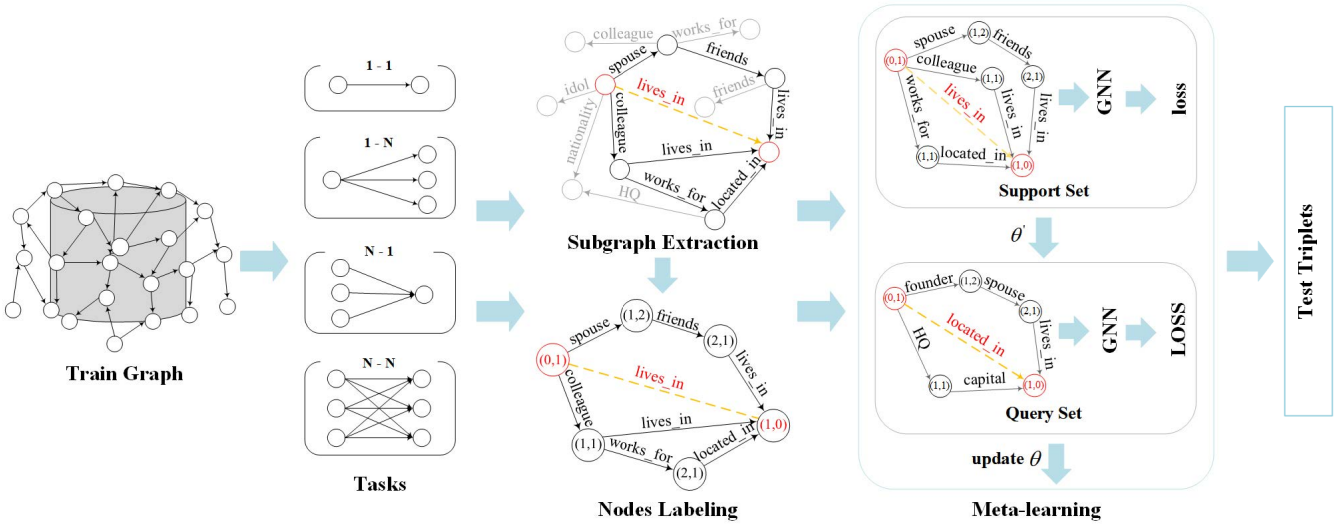

FIGURE 2. Visual illustration of MILP. The yellow dotted lines represent the predictions on the target triplets.

representations of triplets. Meta-Graph [38] learns multiple graphs using meta-learning framework, and then predicts links on few-shot data. Lv *et al.* [39] use high frequency relations to obtain the optimized model parameters and apply them to the low frequency relations. GEN [40] improves RGCN to learn the embedding representations of unknown entities, adding neighbor relations in the process of neighbor information aggregation and using the average aggregation function to generate the representation of neighbor information. GEN regards each entity as a task and uses support sets as the input of the neural network to obtain the representation of the entities, and then passes the learned representation to the entities in the query sets. GEN and Gmatching are part of the comparison models in this paper.

III. METHODOLOGY

A. OVERVIEW

In a triplet (h, r, t), h represents the head entity, t represents the tail entity and r represents the relationship between them. Giving the missing triplet $(?, r, t)$ or $(h, r, ?)$, our model is used to predict the missing entity. The knowledge graph used for training is expressed as $G_{Train} = \{(h, r, t)|h, t \in V_{Train}, r \in R_{Train}\}$, and the knowledge graph used for testing is expressed as $G_{Test} = \{(h, r, t)|h, t \in V_{Test}, r \in R_{Test}\}$, where R_{Train} and R_{Test} represent the relation set in the training knowledge graph and test knowledge graph, $R_{Test} \subset R_{Train}$. V_{Train} and V_{Test} represent the entity set in the training knowledge graph and test knowledge graph respectively. In transductive link prediction, $V_{Test} \subset V_{Train}$, whereas in the inductive link prediction of our work, $V_{Train} \cap V_{Test} = \emptyset$, i.e., a fully-inductive scenario.

Raghu *et al.* [41] prove that MAML is effective due to the reuse of features, i.e., the model parameters contain high quality features, so it can effectively make predictions on new data. Therefore, our model uses meta-learning to learn representations of relations and better initialization parameters to improve the accuracy of prediction results, at the same time, it achieves inductive ability through GNN based subgraph structure. Fig. 2 is the frame diagram of MILP, the process consists of four steps: i) The relations in the train graph are divided into four types by calculating the average appearing counts of entities contained in the relations [42]. In terms of the division rule, we record the average degree of the head entities and tail entities under any relation r as r_h and r_t, respectively. If both r_h and r_t are less than 1.5, the relation r is called a one-to-one relation, if only $r_h < 1.5$, the relation r is a one-to-many relation, if only $r_t < 1.5$, the relation r is a many-to-one relation, if neither r_h nor r_t is less than 1.5, the relation r is a many-to-many relation. Then regard triplets with the same type of relation as the same task, resulting in $\mathcal{T}_1$, $\mathcal{T}_2$, $\mathcal{T}_3$, and $\mathcal{T}_4$ tasks. ii) Extracting subgraphs from the neighbor information for target triplets in each task, and labeling the nodes in the subgraphs. iii) The training of the meta-learning network, which will be discussed in Section III-C. The triplets in each task $\mathcal{T}_i$ are randomly divided into a support set $\boldsymbol{D}_i$ and a query set $\boldsymbol{Q}_i$ in a ratio of approximately 4:1, which are disjoint. iv) After a fixed training epochs, the model with best verification results on the query set is saved for testing. About 15% of the triplets are randomly selected from the test graphs as test triplets, and other triplets are saved as background information for the test triplets, denoted as G'_{Test}.

B. CONSTRUCTION OF SUBGRAPH STRUCTURES

For a target triplet (h, r, t), its directed closed subgraph is obtained firstly by finding the common neighbors of h and t, and then the nodes in the subgraph are initialized based on their relative positions.

1) SUBGRAPH EXTRACTION

The neural message passing scheme assumes that the neighbors of (h, r, t) contain the information about it [18].

Therefore, the directed closed subgraphs of target triplets are extracted firstly by following CoMPILE [21], which consist of the nodes on the path from h to t and the edges between them. $\mathcal{N}^k_{out}(h)$ and $\mathcal{N}^k_{in}(t)$ are the set of k-hop outgoing neighbors of h and the set of k-hop incoming neighbors of t, respectively. Calculating $\mathcal{N}^k_{out}(h) \cap \mathcal{N}^k_{in}(t)$, then removing the isolated nodes and nodes only located on the path length more than k+1, we obtain the k-hop directed common neighbors of the target triplet, as shown in Fig.2, the gray nodes are discarded. The k-hop directed closed subgraph $\mathcal{G}_{(h,r,t)}$ of the target triplet is generated by adding relations between the associated nodes.

2) NODES LABELING

When initializing the nodes in the directed closed subgraph, we represent the nodes according to the relative position of the nodes in the subgraph on the basis of the representation rule in GraIL [18], which is unrelated to the characteristics of the nodes themselves. For each neighbor node v_i in the directed closed subgraph, the shortest distance $d(v_i, h)$ from v_i to h and the shortest distance $d(v_i, t)$ from v_i to t are calculated. Then represent the initial value of v_i as [one-hot$(d(v_i, h)) \oplus$ one-hot$(d(v_i, t))$], where $\oplus$ is the connection operation. h and t are specially marked as (0, 1) and (1, 0) to differentiate from the neighbors. As shown in Fig.2, the nodes of the subgraph are labeled with $(d(v_i, h), d(v_i, t))$.

C. THE NETWORK FRAMEWORK OF MILP

After obtaining the subgraph representations of target triplets, the subgraph data are input into the meta-learning network framework, which updates model parameters based on the total loss on query set in each task.

1) SCORING NETWORK

The meta-learning framework can use different graph neural network models to score, the CoMPILE [21] message passing method is adopted which introduces edge attention mechanism to aggregate the information of nodes and relations for updating. Given the closed subgraph $\mathcal{G}_{(h,r,t)}$ of triplet (h, r, t), the target relation r is taken as a learning parameter, and the information of the triplet is expressed as:

$$\boldsymbol{T} = \boldsymbol{h} + \boldsymbol{r} - \boldsymbol{t} \tag{1}$$

where $\boldsymbol{h}$, $\boldsymbol{r}$ and $\boldsymbol{t}$ represent embeddings of h, r and t, respectively. In the k-th layer, the edge attention of edge i is:

$$a^k_i = f_1((\boldsymbol{T}^k_i \oplus \boldsymbol{T}^k_{\mathcal{G}})\boldsymbol{W}^k_a) \tag{2}$$

where $\boldsymbol{T}^k_i$ represents the triplet of current edge i, $\boldsymbol{T}^k_{\mathcal{G}}$ represents the target triplet in the subgraph, $\boldsymbol{W}^k_a$ is a learnable weight matrix, and f_1 is a two-layer nonlinear transformation. In the k-th layer, the embedding of edge i is:

$$\boldsymbol{E}^k_i = \boldsymbol{h}^k_i \oplus \boldsymbol{r}^k_i \oplus \boldsymbol{t}^k_i \tag{3}$$

The edge i with attention is expressed as:

$$\boldsymbol{E}^{k_a}_i = a^k_i \boldsymbol{E}^k_i \tag{4}$$

In layer k+1, the nodes are updated as:

$$\boldsymbol{V}^{k+1}_{agg} = \boldsymbol{A}^{te}\boldsymbol{E}^{k_a} \tag{5}$$

$$\boldsymbol{V}^{k+1} = f_1((\boldsymbol{V}^{k+1}_{agg} + \boldsymbol{V}^k)\boldsymbol{W}^{k+1}_n) \tag{6}$$

where $\boldsymbol{A}^{te}$ is the adjacency matrix representing the connection of tail entities and edges, $\boldsymbol{E}^{k_a}$ and $\boldsymbol{W}^{k+1}_n$ are the parametric matrices used to learn the embeddings of nodes, $\boldsymbol{V}^{k+1}_{agg}$ represents the aggregated neighbor information in layer k+1, and $\boldsymbol{V}^{k+1}$ represents the updated nodes. The final score function is defined as:

$$\boldsymbol{S} = f_2(\boldsymbol{T}^l_{\mathcal{G}}) \tag{7}$$

where f_2 represents the nonlinear transformation to improve the expressive ability of the model, l represents the number of layers, and $\boldsymbol{T}^l_{\mathcal{G}}$ represents the final representation of the target triplet.

2) META-LEARNING REGIME

The purpose of meta-learning is to learn better parameters for the model, so as to correctly predict missing entities. We train each task $\mathcal{T}_i$ in turn. Firstly, the positive and negative sampling triplets are input into the graph neural network, and the corresponding scores are output. Then we use the hinge loss function proposed by TransE [9] to calculate the loss. As shown in (8), $\mathcal{L}$ represents the training loss in a batch. Among them, ε represents the batch size, p_i and n_i represent the positive and negative samples, respectively, and γ is a hyperparameter representing the margin.

$$\mathcal{L} = \sum_{i=1}^{|\varepsilon|} max(0, \boldsymbol{S}(n_i) - \boldsymbol{S}(p_i) + \gamma) \tag{8}$$

Then, the model parameters are updated by gradient descent to make the model score the positive sampling triplets higher than the negative sampling triplets. In task $\mathcal{T}_i$, the support set $\boldsymbol{D}_i$ is used firstly to calculate the updated parameters through one or more gradient descent, and the parameters updated by a batch are expressed as:

$$\boldsymbol{\theta}' = \boldsymbol{\theta} - \alpha \bigtriangledown_{\boldsymbol{\theta}} \mathcal{L}_{\boldsymbol{d}_i}(f_{\boldsymbol{\theta}}) \tag{9}$$

where α represents the learning rate, and $\boldsymbol{d}_i$ is a batch of samples randomly selected from $\boldsymbol{D}_i$. Then we use $\boldsymbol{\theta}'$ to calculate the loss of the query set $\boldsymbol{Q}_i$ and accumulate the loss of all batches within a task, finally we update $\boldsymbol{\theta}$ according to the total loss.

$$\mathcal{L}_{sum} = \mathcal{L}_{\boldsymbol{q}^{t=1}_i}(f_{\boldsymbol{\theta}}) + \sum_{t=1}^{k} \mathcal{L}_{\boldsymbol{q}^t_i}(f_{\boldsymbol{\theta}'}) \tag{10}$$

$$\boldsymbol{\theta} = \boldsymbol{\theta} - \beta \bigtriangledown_{\boldsymbol{\theta}} \mathcal{L}_{sum} \tag{11}$$

where β represents the meta-learning rate, k represents the total number of batches, and q^t_i represents a batch of samples randomly selected from $\boldsymbol{Q}_i$ at the t-th iteration. Different from MAML, the parameters of neural network are updated

TABLE 1. Statistics of inductive benchmark datasets.

		FB15k-237			NELL-995			WN18RR		
		relations	nodes	links	relations	nodes	links	relations	nodes	links
v1	train graph	85	1312	3221	10	1585	3834	9	4475	9604
	test graph	85	902	1614	9	540	1590	9	2435	3427
v2	train graph	93	1923	3430	39	1135	3382	6	6092	10586
	test graph	93	1233	1934	36	834	1549	6	2159	2971
v3	train graph	105	4191	9685	57	3141	6585	11	15871	35735
	test graph	45	1427	2155	57	2246	6035	11	5779	7203
v4	train graph	62	2304	3649	25	1255	4314	6	4307	8313
	test graph	54	1460	1946	25	1424	2027	6	8553	13914

after a task. The operation in Equations (8)-(11) for each task is performed, and the final parameters are obtained after fixed epochs of training. The specific process is shown in Algorithm 1.

Algorithm 1 MILP Training

Input: GNN parameters $\boldsymbol{\theta}$, learning rate α, meata-learning rate β, meta-training task set $\mathbb{T}_{meta-training}$

Output: GNN parameters $\boldsymbol{\theta}$

while not done **do**
  Get the tasks $\mathcal{T}_{meta-training}$ from $\mathbb{T}_{meta-training}$
  for $\mathcal{T}_i$ in $\mathcal{T}_{meta-training}$ **do**
    Shuffle the batchs in $\mathcal{T}_i$
    for iteration = 1, 2, ...k **do**
      Sample datapoints $\boldsymbol{d}_i$ from $\boldsymbol{D}_i$
      Calculate the batch loss $\mathcal{L}_{\boldsymbol{d}_i}(f_{\boldsymbol{\theta}})$ via (8)
      $\boldsymbol{\theta}^{'} \leftarrow \boldsymbol{\theta} - \alpha \nabla_{\boldsymbol{\theta}} \mathcal{L}_{\boldsymbol{d}_i}(f_{\boldsymbol{\theta}})$
      Sample datapoints $\boldsymbol{q}_i$ from $\boldsymbol{Q}_i$
      if iteration == 1 **then**
        Calculate the batch loss $\mathcal{L}_{\boldsymbol{q}_i}(f_{\boldsymbol{\theta}})$ via (8)
      end if
      Calculate the batch loss $\mathcal{L}_{\boldsymbol{q}_i}(f_{\boldsymbol{\theta}}^{'})$ via (8)
    end for
    Calculate the total loss of $\mathcal{L}_{sum}$ via (10)
    $\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} - \beta \nabla_{\boldsymbol{\theta}} \mathcal{L}_{sum}$
  end for
end while

IV. EXPERIMENTS

The experimental datasets in this paper are derived from three benchmark datasets: FB15k-237, WN18RR, and NELL-995. The purpose of our experiments is to verify the effectiveness of the proposed model for inductive link prediction on few-shot data.

A. EXPERIMENTAL SETUP

1) DATASETS

FB15k-237 [43], WN18RR [44], and NELL-995 [45] are three public datasets for knowledge graph reasoning, initially used for transductive link prediction. Teru *et al.* [18] extract four datasets suitable for inductive link prediction from each of them, the entities in training set and test set of each dataset are disjoint. Referring to the division of Teru *et al.*, we make modifications on the basis of their datasets and further extract few-shot inductive datasets. The extracted datasets have the following characteristics: i) small amount of data, the proportion of entities that contain only a few triplets is increased, and the triplets for most entities do not exceed 30; ii) the relations of each dataset includes four types: one-to-one, one-to-many, many-to-one, and many-to-many, the relations have the same relation types in the training set and test set; iii) meeting the requirements of inductive link prediction, and we include entities not present in background information in test triplets. The specific information is shown in Table 1. All the experiments in this paper are carried out on these inductive datasets.

2) BASELINES

The baselines include: RuleN [24] based on statistical rules, differentiable models Neural-LP [26] and DRUM [27] based on logic rules, GraIL [18] and CoMPILE [21] based on graph neural network, and Gmatching [19] and GEN [40] based meta-learning. To test the inductive ability of the models, none of the models used embedding methods to obtain initial representations of entities and relations. It should be point out that Gmatching aims to predict one-shot relations. Specifically, $R_{Train} \cap R_{Test} = \emptyset$, $V_{Test} \subset V_{Train}$. And in the created datasets NELL-One and Wiki-One [19], Gmatching uses triplets removed training data and test data as background information. This scenario is different from ours, and we achieve very poor results with the same setting on our datasets. In our experiments, G_{Train} and G'_{Test} are used as the background information for Gmatching during training and test, respectively, which can provide more accurate information for target triplets, thus the prediction results of Gmatching are significantly improved. During the experiments, the settings of baseline models are consistent with the best settings pointed out by the authors in their papers and codes, including hyperparameters, etc.

TABLE 2. Inductive link prediction results of AUC.

	FB15k-237				NELL-995				WN18RR			
	v1	v2	v3	v4	v1	v2	v3	v4	v1	v2	v3	v4
RuleN	64.70	53.58	45.11	44.40	69.19	44.60	50.41	38.98	29.25	51.41	47.42	40.44
Neural-LP	44.37	31.88	29.82	26.55	34.00	34.24	61.32	54.96	23.19	47.77	35.10	39.90
DRUM	42.35	32.47	41.89	26.39	38.04	34.65	63.93	55.82	23.55	47.85	44.13	39.85
GraIL	88.18	80.58	62.99	63.76	70.89	62.26	89.81	71.20	51.11	58.13	72.37	54.41
CoMPILE	89.13	91.77	84.27	69.87	83.52	75.30	**96.88**	65.00	53.77	52.36	81.76	59.58
Gmatching	75.29	61.40	52.95	53.29	73.17	59.77	69.88	63.38	53.65	64.79	54.11	45.36
GEN	78.69	57.79	73.26	60.41	77.08	63.69	82.64	**74.08**	49.95	51.29	73.66	58.91
MILP	**91.76**	**93.82**	**88.85**	**71.31**	**91.99**	**92.59**	93.48	56.45	**57.98**	**66.17**	**82.96**	**65.87**

TABLE 3. Inductive link prediction results of AUC-PR.

	FB15k-237				NELL-995				WN18RR			
	v1	v2	v3	v4	v1	v2	v3	v4	v1	v2	v3	v4
RuleN	64.65	60.19	56.85	50.70	77.18	53.00	50.41	38.98	49.23	55.02	61.78	49.03
Neural-LP	69.96	62.92	56.77	51.90	57.65	54.55	61.32	54.96	49.61	55.90	56.83	52.00
DRUM	67.59	63.16	55.74	51.56	62.14	55.06	63.93	55.82	50.22	56.03	65.63	51.99
GraIL	90.96	85.68	69.64	67.42	82.42	67.05	89.81	**71.20**	51.58	57.70	75.56	54.82
CoMPILE	92.02	92.73	88.57	77.18	76.76	77.29	**96.88**	65.00	57.30	57.29	83.49	61.44
Gmatching	72.20	60.30	53.24	53.36	70.83	59.25	72.48	60.00	57.33	62.36	55.97	50.51
GEN	78.53	56.34	72.88	59.44	75.95	62.62	82.87	70.90	49.63	51.67	70.72	54.93
MILP	**93.12**	**94.74**	**91.94**	**78.01**	**91.76**	**91.09**	93.20	62.60	**58.33**	**73.51**	**84.02**	**71.28**

3) EVALUATION METRICS

AUC, AUC-PR, Mean Reciprocal Rank (MRR), and Hits@k (k = 1, 3, 10) are used as the evaluation metrics. AUC and AUC-PR are used during training, MRR and Hits@k are also used during testing. AUC and AUC-PR, which respectively represent the probability of positive samples have higher scores than negative samples and precision-recall curve, generate a negative sample for each positive sample when they are calculated. MRR is the mean reciprocal rank of positive samples. Hits@k is the proportion of positive samples ranked in top-k among all test triplets. When calculating MRR and Hits@k, the evaluation of each target triplet includes two groups of samples. The first group is the prediction of the head entity, including one positive sample and 49 negative samples obtained by randomly substituting the head entity of the positive sample with other entities. The second group predicts the tail entity, including one positive sample and 49 negatives samples obtained by randomly substituting the tail entity of the positive sample with other entities.

4) IMPLEMENTATION

The model is implemented in PyTorch, equipment used is Tesla P100 with 12GB RAM. In the experiments, the first and second layers of f_1, and f_2 are the ReLU, Tanh and the Sigmoid activation function, respectively. 3-hop neighbors of the directed closed subgraph are extracted, and the training epochs are set to 50. The margin in loss function $\gamma = 10$, and the number of GNN layers $l = 3$. In WN18RR-v2 and WN18RR-v4 we use fine-turning mechanism to improve AUC and AUC-PR. More implementation details of MILP and the baselines are shown in the supplementary materials.

B. RESULTS AND DISCUSSION

On the datasets created in this paper, the proposed model is compared with baselines. Running 10 times on each test set, we get relatively stable results.

1) MAIN RESULTS

The best results for all models of AUC, AUC-PR, and Hits@10 are shown in Table 2, Table 3, and Table 4. Numbers in bold denote the best results on each dataset while the underlined ones are the suboptimal results. It can be seen that in most cases GNN based and meta-learning based models achieve better results than logical rule based models. MILP achieves the best results for the most part, in particular, on NELL-995-v2 outperforms the suboptimal models on AUC and AUC-PR by 17.29% and 13.80% respectively, and 22.99% higher than the suboptimal model about Hits@10 on WN18RR-v4. CoMPILE is the second-best model with respect to AUC and AUC-PR, and GraIL and Gmatching are the second-best models of Hits@10. MILP has the best performance on all three metrics on the FB15k-237 and WN18RR datasets. On NELL-995-v1 and NELL-995-v2, Gmatching has better Hits@10 than MILP, it probably

TABLE 4. Inductive link prediction results of Hits@10.

	FB15k-237				NELL-995				WN18RR			
	v1	v2	v3	v4	v1	v2	v3	v4	v1	v2	v3	v4
RuleN	64.53	42.03	22.50	15.59	79.09	22.43	45.48	26.43	0.88	10.05	29.53	4.66
Neural-LP	48.25	31.58	18.15	5.97	46.26	17.71	44.34	16.40	1.80	12.06	39.13	4.45
DRUM	48.99	32.29	15.05	6.06	47.82	18.46	46.21	16.33	1.80	12.06	47.83	4.39
GraIL	77.03	58.91	27.50	19.64	67.82	25.63	89.23	39.11	2.59	12.56	36.99	5.08
CoMPILE	75.68	57.81	26.52	13.97	55.45	26.05	90.85	40.71	3.03	10.05	34.58	5.94
Gmatching	70.27	50.30	33.84	32.79	**83.82**	**39.83**	73.04	52.68	30.26	39.20	43.56	13.61
GEN	44.59	15.94	35.98	12.65	38.55	24.21	51.62	39.29	1.58	3.52	36.46	7.14
MILP	**81.08**	**60.31**	**37.77**	**33.91**	81.45	36.34	**92.12**	**58.39**	**33.86**	**42.11**	**52.77**	**36.60**

because the one-hop neighbors of the target triplets provide more valuable information as Gmatching aggregates all one-hop neighbor information. On NELL-995-v3 and NELL-995-v4, the lower AUC and AUC-PR of MILP may be due to the datasets contain noisy data, which reduce the ability of MILP to distinguish positive samples from negative samples.

We observe that test results on different datasets show great differences, mainly due to the different number of triplets in the background information owned by the test entities in different datasets, and the test entities refer to the entities in the test triplets. In particular, the prediction results of link prediction models on zero-shot and one-shot entities are lower, so increasing the proportion of that entities will lead to lower overall test results of the datasets. Zero-shot entities are those that only appear in the test triplets, and one-shot entities refer to entities that only appear once in the background information. Fig.3 shows that the proportion of test entities with different number of triplets on all benchmark datasets. For example, compare and analyze the content of Fig. 3 and the results of Hits@10 that have the most obvious change. From FB15k-237-v1 to FB15k-237-v4, the proportion of entities with 0 or 1 triplet is increasing, but the results of Hits@10 are lower and lower. Entities with 0 or 1 triplet in NELL-995-v1 and NELL-995-v3 are higher than those in NELL-995-v2 and NELL-995-v4, but the results of Hits@10 shows the opposite characteristics. Similarly, on WN18RR-v1, WN18RR-v2, and WN18RR-v4 the proportion of zero-shot and one-shot entities increases significantly compared to WN18RR-v3, the Hits@10 results decrease significantly. This also shows that there is still room for improvement in the performance of the models in predicting zero-shot and one-shot entities, especially GEN and models based on logical rules and graph neural networks, whose performance drop significantly.

2) ZERO-SHOT AND ONE-SHOT ENTITY PREDICTION

Taking WM18RR-v3 and WN18RR-v4 as examples, the prediction results of the models based on GNN and meta-learning on zero-shot entities and one-shot entities are displayed in a finer granularity. As shown in Fig. 4, the statistical metrics are Hits@1, Hits@3, and Hits@10, and

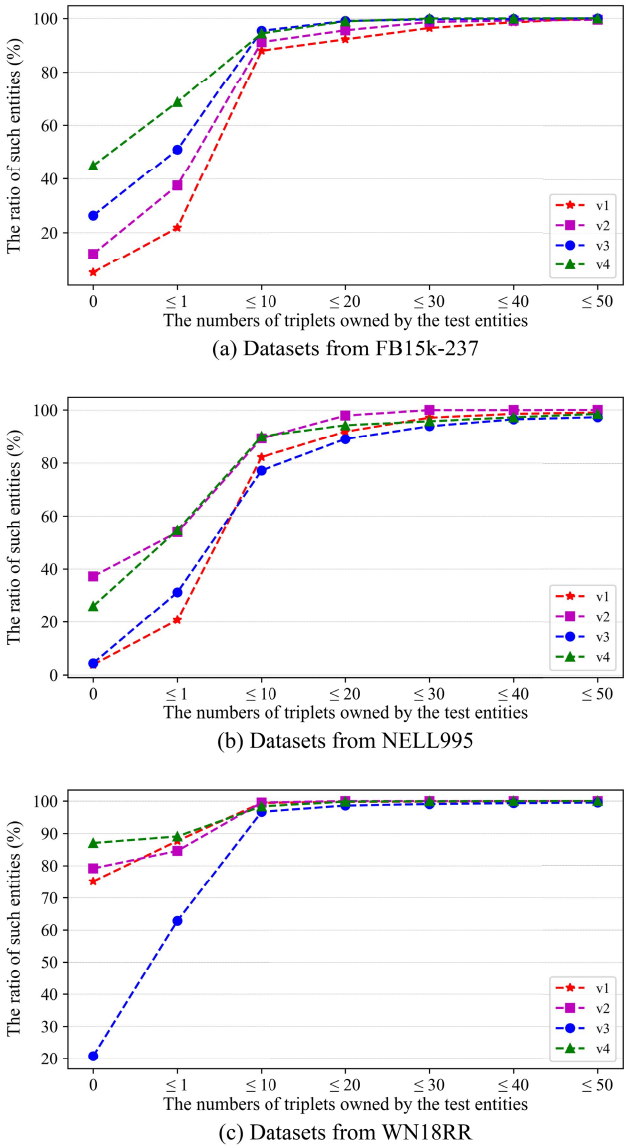

FIGURE 3. Distribution of test entities with different triplets on all benchmark datasets.

the missing models in the histogram indicate that the corresponding results are zero. From the diagram, we can see that

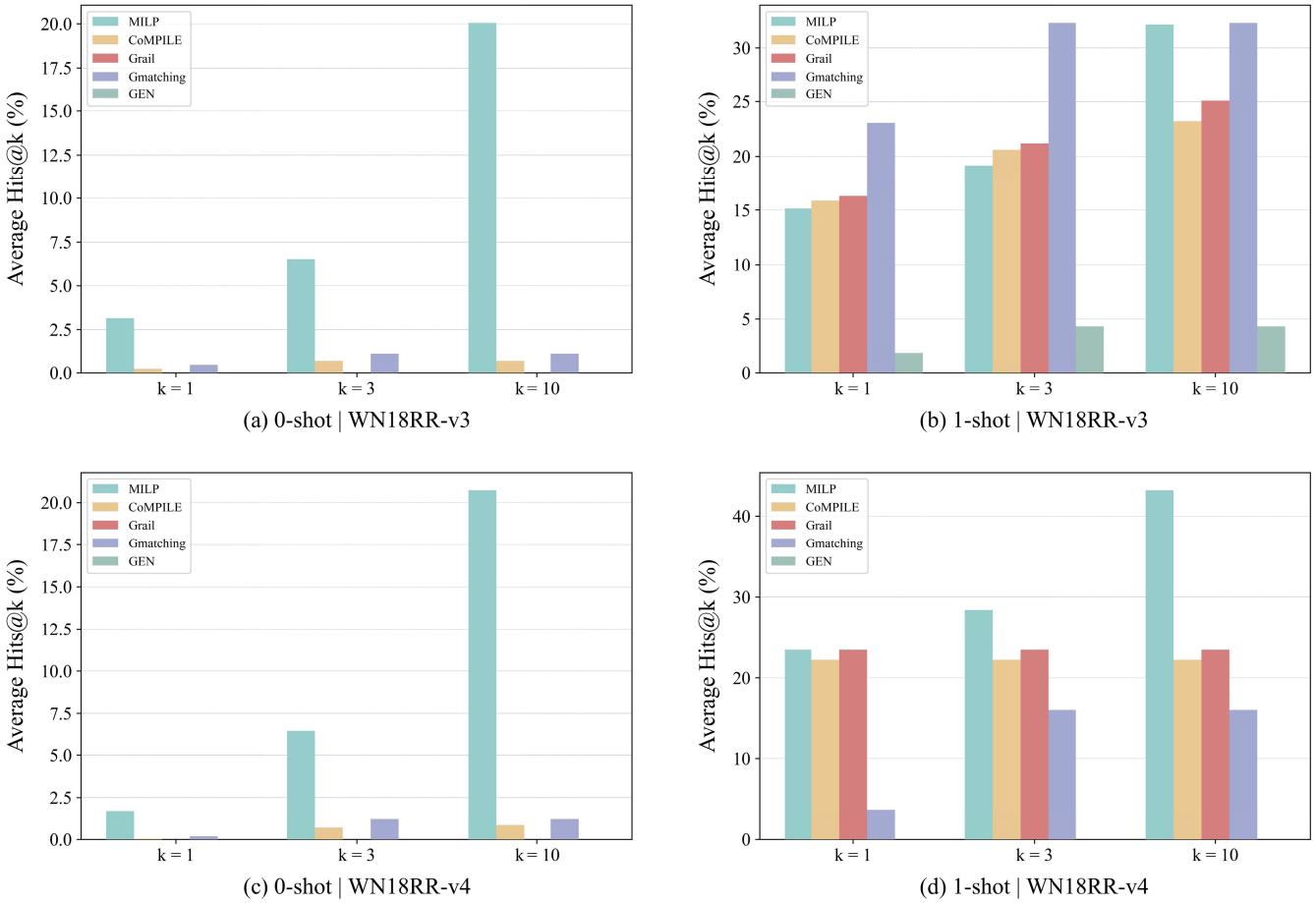

FIGURE 4. Zero-shot and one-shot entity prediction results on WN18RR-v3 and WN18RR-v4.

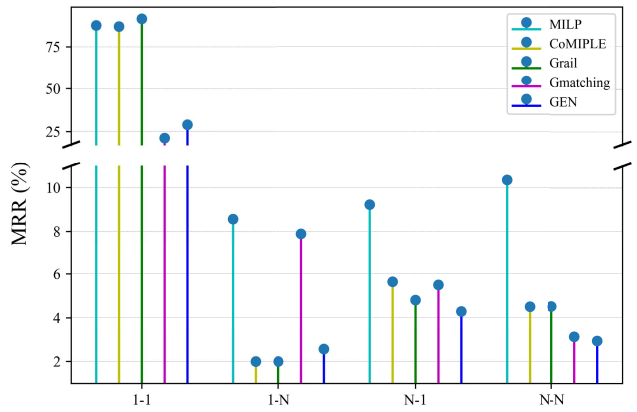

FIGURE 5. MRR results of different types of relations on WN18RR-v4.

MILP yields better results in all other cases except that it performs poorer on Hits@1 and Hits@3 in the one-shot entity prediction of WN18RR-v3, especially for zero-shot MILP has obvious advantages. The results of Gmatching are unstable because it relies too much on background information. The model GEN, which predicts by obtaining the embedding representations of entities, has not achieved good performance. GraIL and CoMPILE are also significantly more effective in handing one-shot than zero-shot. In addition, it is found that the results of GraIL and GEN on zero-shot are poor on all experimental datasets, and MILP has obvious advantages on Hits@10.

3) EVALUATION ON DIFFERENT RELATION TYPES

Fig. 5 illustrates the MRR results of MILP, CoMPILE, GraIL, Gmatching, and GEN on WN18RR-v4 with different relation types. It can be seen that in addition to the result of one-to-one relations on WN18RR-v4 is slightly lower than that of GraIL, the best results are obtained by MILP on one-to-many, many-to-one, and many-to-many relations, explaining that it has more advantages in dealing with complex relations.

V. CONCLUSION

We propose the model MILP using meta-learning and graph neural network for inductive link prediction in knowledge graph. Through meta-learning, four different relation types of tasks are trained to learn better initialization parameters and improve the prediction results of the model. In this article, we extract new inductive link prediction datasets from public knowledge graphs to set up scenarios for few-shot entity

prediction, and MILP is compared with the baseline models on these inductive datasets. The experimental results show that MILP has better generalization ability than the baseline models. In most datasets, MILP achieves the best prediction results about AUC, AUC-PR, and Hits@10. Finally, a finer-grained analysis indicates the effectiveness of MILP in predicting entities with only 0 or 1 triplet and different types of relations.

In future work, we will continue to study inductive link prediction, the directions we're interested include: i) using external resources to enhance the representation of entities and relations; ii) enhancing the information capture capability of GNN by data enhancement [46] and iii) exploring the method to improve prediction accuracy of zero-shot entities.

REFERENCES

[1] C. Wang, H. Yu, and F. Wan, "Information retrieval technology based on knowledge graph," in *Proc. 3rd Int. Conf. Adv. Mater., Mechatronics Civil Eng. (ICAMMCE)*, 2018, pp. 291–296.

[2] Z. Chen, S. Yin, and X. Zhu, "Research and implementation of QA system based on the knowledge graph of Chinese classic poetry," in *Proc. IEEE 5th Int. Conf. Cloud Comput. Big Data Analytics (ICCCBDA)*, Apr. 2020, pp. 495–499.

[3] S. Lv, D. Guo, J. Xu, D. Tang, N. Duan, M. Gong, L. Shou, D. Jiang, G. Cao, and S. Hu, "Graph-based reasoning over heterogeneous external knowledge for commonsense question answering," in *Proc. 34th AAAI Conf. Artif. Intell.*, 2020, vol. 34, no. 5, pp. 8449–8456.

[4] H. Li, Y. Liu, N. Mamoulis, and D. S. Rosenblum, "Translation-based sequential recommendation for complex users on sparse data," *IEEE Trans. Knowl. Data Eng.*, vol. 32, no. 8, pp. 1639–1651, Aug. 2020.

[5] V. N. Ioannidis, D. Zheng, and G. Karypis, "Few-shot link prediction via graph neural networks for COVID-19 drug-repurposing," 2020, *arXiv:2007.10261*.

[6] E. B. Myklebust, E. Jiménez-Ruiz, J. Chen, R. Wolf, and K. E. Tollefsen, "Prediction of adverse biological effects of chemicals using knowledge graph embeddings," *Semantic Web*, vol. 13, no. 3, pp. 299–338, Apr. 2022.

[7] 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. Manag. Data (SIGMOD)*, 2008, pp. 1247–1250.

[8] D. Vrandečić and M. Krötzsch, "Wikidata: A free collaborative knowledgebase," *Commun. ACM*, vol. 57, no. 10, pp. 78–85, 2014.

[9] 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. (NIPS)*, 2013, pp. 2787–2795.

[10] Z. Wang, J. Zhang, J. Feng, and Z. Chen, "Knowledge graph embedding by translating on hyperplanes," in *Proc. 28th AAAI Conf. Artif. Intell.*, 2014, vol. 28, no. 1, pp. 1112–1119.

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

[12] Z. Sun, Z.-H. Deng, J.-Y. Nie, and J. Tang, "RotatE: Knowledge graph embedding by relational rotation in complex space," in *Proc. Int. Conf. Learn. Represent. (ICLR)*, 2019, pp. 1–18.

[13] D. Song, F. Zhang, M. Lu, S. Yang, and H. Huang, "DTransE: Distributed translating embedding for knowledge graph," *IEEE Trans. Parallel Distrib. Syst.*, vol. 32, no. 10, pp. 2509–2523, Oct. 2021.

[14] Y. Hao, X. Cao, Y. Fang, X. Xie, and S. Wang, "Inductive link prediction for nodes having only attribute information," in *Proc. 29th Int. Joint Conf. Artif. Intell.*, Jul. 2020, pp. 1209–1215.

[15] C. Fu, T. Chen, M. Qu, W. Jin, and X. Ren, "Collaborative policy learning for open knowledge graph reasoning," in *Proc. EMNLP-IJCNLP*, Hong Kong, 2019, pp. 2672–2681.

[16] M. Ali, M. Berrendorf, M. Galkin, V. Thost, T. Ma, V. Tresp, and J. Lehmann, "Improving inductive link prediction using hyper-relational facts," in *The Semantic Web—ISWC* (Lecture Notes in Computer Science). Cham, Switzerland: Springer, Oct. 2021, pp. 74–92.

[17] N. Reimers and I. Gurevych, "Sentence-BERT: Sentence embeddings using Siamese BERT-networks," in *Proc. EMNLP-IJCNLP*, Hong Kong, Nov. 2019, pp. 3982–3992.

[18] K. Teru, E. Denis, and W. Hamilton, "Inductive relation prediction by subgraph reasoning," in *Proc. Int. Conf. Mach. Learn. (ICML)*, 2020, pp. 9448–9457.

[19] W. Xiong, M. Yu, S. Chang, X. Guo, and W. Y. Wang, "One-shot relational learning for knowledge graphs," in *Proc. Conf. Empirical Methods Natural Lang. Process.*, 2018, pp. 1980–1990.

[20] Y. Li, K. Yu, Y. Zhang, and X. Wu, "Learning relation-specific representations for few-shot knowledge graph completion," 2022, *arXiv:2203.11639*.

[21] S. Mai, S. Zheng, Y. Yang, and H. Hu, "Communicative message passing for inductive relation reasoning," in *Proc. 35th AAAI Conf. Artif. Intell.*, 2021, pp. 4294–4302.

[22] Q. Zeng, J. M. Patel, and D. Page, "QuickFOIL: Scalable inductive logic programming," *Proc. VLDB Endowment*, vol. 8, no. 3, pp. 197–208, Nov. 2014.

[23] L. A. Galárraga, C. Teflioudi, K. Hose, and F. Suchanek, "AMIE: Association rule mining under incomplete evidence in ontological knowledge bases," in *Proc. 22nd Int. Conf. World Wide Web (WWW)*, 2013, pp. 413–422.

[24] C. Meilicke, M. Fink, Y. Wang, D. Ruffinelli, R. Gemulla, and H. Stuckenschmidt, "Fine-grained evaluation of rule- and embedding-based systems for knowledge graph completion," in *The Semantic Web—ISWC* (Lecture Notes in Computer Science). Cham, Switzerland: Springer, 2018, pp. 3–20.

[25] W. W. Cohen, "TensorLog: A differentiable deductive database," 2016, *arXiv:1605.06523*.

[26] F. Yang, Z. Yang, and W. W. Cohen, "Differentiable learning of logical rules for knowledge base reasoning," in *Proc. Adv. Neural Inf. Process. Syst. (NIPS)*, 2017, pp. 2319–2328.

[27] A. Sadeghian, M. Armandpour, P. Ding, and D. Z. Wang, "DRUM: End-to-end differentiable rule mining on knowledge graphs," in *Proc. Adv. Neural Inf. Process. Syst. (NIPS)*, 2019, pp. 15347–15357.

[28] J. Zhou, G. Cui, S. Hu, Z. Zhang, C. Yang, Z. Liu, L. Wang, C. Li, and M. Sun, "Graph neural networks: A review of methods and applications," *AI Open*, vol. 1, pp. 57–81, Jan. 2020.

[29] K. Chen, H. Che, X. Li, and M.-F. Leung, "Graph non-negative matrix factorization with alternative smoothed L_0 regularizations," *Neural Comput. Appl.*, pp. 1–15, Jun. 2022, doi: 10.1007/s00521-022-07200-w.

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

[31] M. Schlichtkrull, T. N. Kipf, P. Bloem, R. V. D. Berg, I. Titov, and M. Welling, "Modeling relational data with graph convolutional networks," in *Proc. Eur. Semantic Web Conf.* Cham, Switzerland: Springer, 2018, pp. 593–607.

[32] W. L. Hamilton, Z. Ying, and J. Leskovec, "Inductive representation learning on large graphs," in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 30, 2017, pp. 1024–1034.

[33] J. Chen, H. He, F. Wu, and J. Wang, "Topology-aware correlations between relations for inductive link prediction in knowledge graphs," in *Proc. 35th AAAI Conf. Artif. Intell.*, 2021, pp. 6271–6278.

[34] C. Finn, P. Abbeel, and S. Levine, "Model-agnostic meta-learning for fast adaptation of deep networks," in *Proc. Int. Conf. Mach. Learn. (ICML)*, 2017, pp. 1126–1135.

[35] A. Nichol, J. Achiam, and J. Schulman, "On first-order meta-learning algorithms," 2018, *arXiv:1803.02999*.

[36] O. Vinyals, C. Blundell, T. Lillicrap, K. Kavukcuoglu, and D. Wierstra, "Matching networks for one shot learning," in *Proc. Adv. Neural Inf. Process. Syst. (NIPS)*, vol. 29, Barcelona, Spain, 2016, pp. 3637–3645.

[37] Z. Wang, K. Lai, P. Li, L. Bing, and W. Lam, "Tackling long-tailed relations and uncommon entities in knowledge graph completion," in *Proc. EMNLP-IJCNLP*, Hong Kong, 2019, pp. 250–260.

[38] A. Joey Bose, A. Jain, P. Molino, and W. L. Hamilton, "Meta-graph: Few shot link prediction via meta learning," 2019, *arXiv:1912.09867*.

[39] X. Lv, Y. Gu, X. Han, L. Hou, J. Li, and Z. Liu, "Adapting meta knowledge graph information for multi-hop reasoning over few-shot relations," in *Proc. EMNLP-IJCNLP*, 2019, pp. 3374–3379.

[40] J. Baek, D. B. Lee, and S. J. Hwang, "Learning to extrapolate knowledge: Transductive few-shot out-of-graph link prediction," in *Proc. Adv. Neural Inf. Process. Syst. (NIPS)*, vol. 33, 2020, pp. 546–560.

[41] A. Raghu, M. Raghu, S. Bengio, and O. Vinyals, "Rapid learning or feature reuse? Towards understanding the effectiveness of MAML," in *Proc. Int. Conf. Learn. Represent. (ICLR)*, 2020, pp. 1–21.
[42] Z. Zhu, Z. Zhang, L.-P. Xhonneux, and J. Tang, "Neural Bellman-Ford networks: A general graph neural network framework for link prediction," in *Proc. Adv. Neural Inf. Process. Syst. (NIPS)*, vol. 34, 2021, pp. 29476–29490.
[43] K. Toutanova, D. Chen, P. Pantel, H. Poon, P. Choudhury, and M. Gamon, "Representing text for joint embedding of text and knowledge bases," in *Proc. Conf. Empirical Methods Natural Lang. Process.*, 2015, pp. 1499–1509.
[44] T. Dettmers, P. Minervini, P. Stenetorp, and S. Riedel, "Convolutional 2D knowledge graph embeddings," in *Proc. 22th AAAI Conf. Artif. Intell.*, 2018, vol. 32, no. 1, pp. 1–8.
[45] W. Xiong, T. Hoang, and W. Y. Wang, "DeepPath: A reinforcement learning method for knowledge graph reasoning," in *Proc. Conf. Empirical Methods Natural Lang. Process.*, 2017, pp. 564–573.
[46] K. Kong, G. Li, M. Ding, Z. Wu, C. Zhu, B. Ghanem, G. Taylor, and T. Goldstein, "Robust optimization as data augmentation for large-scale graphs," in *Proc. CVPR*, 2022, pp. 60–69.

RUITING YANG received the B.S. degree in the Internet of Things engineering from the Hebei University of Engineering, Handan, China, in 2019, where she is currently pursuing the master's degree in computer science and technology. Her research interests include knowledge graph reasoning and machine learning.

ZHONGCHENG WEI received the Ph.D. degree in information and communication engineering from the Beijing University of Posts and Telecommunications, Beijing, China, in 2016. He is currently an Associate Professor with the School of Information and Electrical Engineering, Hebei University of Engineering, and serves as a member for the Hebei Key Laboratory of Security and Protection Information Sensing and Processing. He has coauthored 20 publications, held five China national invention patents, and seven software copyrights. He has presided over or participated in more than ten projects on scientific research. His research interests include the Internet of Things, wireless communication, big data, and artificial intelligence.

YONGJIAN FAN received the B.S. degree from the Hebei University of Engineering (HUE), China, in 2009, and the Ph.D. degree from the Renmin University of China, China, in 2013. He is currently an Associate Professor and a B.S. Supervisor with the School of Information and Electrical Engineering, HUE, and serves as a member for the Hebei Key Laboratory of Security Protection Information Sensing and Processing. His research interests include data mining, wireless networks, and privacy preservation.

JIJUN ZHAO (Member, IEEE) received the Ph.D. degree in electromagnetic and microwave technique from the Beijing University of Posts and Telecommunications, Beijing, China, in 2003. He was a Postdoctoral Researcher with ZTE Corporation. He was the Dean of the School of Information and Electrical Engineering for nine years. He is currently a Full Professor with the Hebei University of Engineering and the Dean of the Graduate Department. He serves as the Vice Director for the Hebei Institute of Electronics. He is also the Vice Dean of the Hebei Key Laboratory of Security and Protection Information Sensing and Processing. He is a principle/co-investigator in several research projects funded by the National High Technology Research and Development Program of China. He has published more than 70 papers and applied the ten patents of invention. His current research interests include broadband communication networks, the Internet of Things, and smart security and protection.

He is a member of ACM.