Meta Relational Learning for Few-Shot Link Prediction in Knowledge Graphs

Mingyang Chen[1,*], **Wen Zhang**[1,*], **Wei Zhang**[2], **Qiang Chen**[2], **Huajun Chen**[1,3,†]

[1]College of Computer Science and Technology, Zhejiang University
[2]Alibaba Group
[3]AZFT Joint Lab for Knowledge Engine
{mingyangchen, wenzhang2015, huajunsir}@zju.edu.cn
{lantu.zw, lapu.cq}@alibaba-inc.com

Abstract

Link prediction is an important way to complete knowledge graphs (KGs), while embedding-based methods, effective for link prediction in KGs, perform poorly on relations that only have a few associative triples. In this work, we propose a Meta Relational Learning (MetaR) framework to do the common but challenging few-shot link prediction in KGs, namely predicting new triples about a relation by only observing a few associative triples. We solve few-shot link prediction by focusing on transferring relation-specific meta information to make model learn the most important knowledge and learn faster, corresponding to relation meta and gradient meta respectively in MetaR. Empirically, our model achieves state-of-the-art results on few-shot link prediction KG benchmarks.

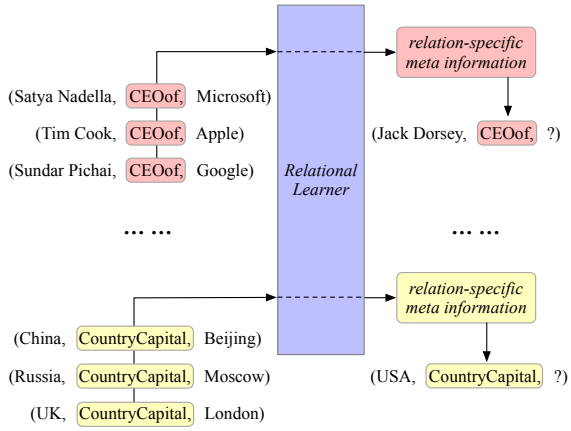

Figure 1: An example of 3-shot link prediction in KGs. One task represents observing only three instances of one specific relation and conducting link prediction on this relation. Our model focuses on extracting relation-specific meta information by a kind of relational learner which is shared across tasks and transferring this meta information to do link prediction within one task.

1 Introduction

A knowledge graph is composed by a large amount of triples in the form of $(head\ \ entity, relation, tail\ \ entity)$ ((h, r, t) in short), encoding knowledge and facts in the world. Many KGs have been proposed (Vrandei and Krtzsch, 2014; Bollacker et al., 2008; Carlson et al., 2010) and applied to various applications (Bordes et al., 2014; Zhang et al., 2016, 2019a).

Although with huge amount of entities, relations and triples, many KGs still suffer from incompleteness, thus knowledge graph completion is vital for the development of KGs. One of knowledge graph completion tasks is link prediction, predicting new triples based on existing ones. For link prediction, KG embedding methods (Bordes et al., 2013; Nickel et al., 2011; Trouillon et al., 2016; Yang et al., 2015) are promising ways. They learn latent representations, called embeddings, for entities and relations in continuous vector space and accomplish link prediction via calculation with embeddings.

The effectiveness of KG embedding methods is promised by sufficient training examples, thus results are much worse for elements with a few instances during training (Zhang et al., 2019c). However, few-shot problem widely exists in KGs. For example, about 10% of relations in Wikidata (Vrandei and Krtzsch, 2014) have no more than 10 triples. Relations with a few instances are called few-shot relations. In this paper, we devote to discuss *few-shot link prediction in knowledge graphs*, predicting tail entity t given head entity h and relation r by only observing K triples about r, usually K is small. Figure 1 depicts an example of 3-shot link prediction in KGs.

To do few-shot link prediction, Xiong et al.

* Equal contribution.
† Corresponding author.

Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing, pages 4217–4226, Hong Kong, China, November 3–7, 2019.

(2018) made the first trial and proposed GMatching, learning a matching metric by considering both learned embeddings and one-hop graph structures, while we try to accomplish few-shot link prediction from another perspective based on the intuition that *the most important information to be transferred from a few existing instances to incomplete triples should be the common and shared knowledge within one task.* We call such information *relation-specific meta information* and propose a new framework Meta Relational Learning (MetaR) for few-shot link prediction. For example, in Figure 1, relation-specific meta information related to the relation *CEOof* or *CountryCapital* will be extracted and transferred by MetaR from a few existing instances to incomplete triples.

The relation-specific meta information is helpful in the following two perspectives: 1) transferring common relation information from observed triples to incomplete triples, 2) accelerating the learning process within one task by observing only a few instances. Thus we propose two kinds of relation-specific meta information: *relation meta* and *gradient meta* corresponding to afore mentioned two perspectives respectively. In our proposed framework MetaR, relation meta is the high-order representation of a relation connecting head and tail entities. Gradient meta is the loss gradient of relation meta which will be used to make a rapid update before transferring relation meta to incomplete triples during prediction.

Compared with GMatching (Xiong et al., 2018) which relies on a background knowledge graph, our MetaR is independent with them, thus is more robust as background knowledge graphs might not be available for few-shot link prediction in real scenarios.

We evaluate MetaR with different settings on few-shot link prediction datasets. MetaR achieves state-of-the-art results, indicating the success of transferring relation-specific meta information in few-shot link prediction tasks. In summary, main contributions of our work are three-folds:

- Firstly, we propose a novel meta relational learning framework (MetaR) to address few-shot link prediction in knowledge graphs.
- Secondly, we highlight the critical role of relation-specific meta information for few-shot link prediction, and propose two kinds of relation-specific meta information, *relation meta* and *gradient meta*. Experiments show that both of them contribute significantly.
- Thirdly, our MetaR achieves state-of-the-art results on few-shot link prediction tasks and we also analyze the facts that affect MetaR's performance.

2 Related Work

One target of MetaR is to learn the representation of entities fitting the few-shot link prediction task and the learning framework is inspired by knowledge graph embedding methods. Furthermore, using loss gradient as one kind of meta information is inspired by MetaNet (Munkhdalai and Yu, 2017) and MAML (Finn et al., 2017) which explore methods for few-shot learning by meta-learning. From these two points, we regard knowledge graph embedding and meta-learning as two main kinds of related work.

2.1 Knowledge Graph Embedding

Knowledge graph embedding models map relations and entities into continuous vector space. They use a score function to measure the truth value of each triple (h, r, t). Same as knowledge graph embedding, our MetaR also need a score function, and the main difference is that representation for r is the learned relation meta in MetaR rather than embedding of r as in normal knowledge graph embedding methods.

One line of work is started by TransE (Bordes et al., 2013) with distance score function. TransH (Wang et al., 2014) and TransR (Lin et al., 2015b) are two typical models using different methods to connect head, tail entities and their relations. DistMult (Yang et al., 2015) and ComplEx (Trouillon et al., 2016) are derived from RESCAL (Nickel et al., 2011), trying to mine latent semantics in different ways. There are also some others like ConvE (Dettmers et al., 2018) using convolutional structure to score triples and models using additional information such as entity types (Xie et al., 2016) and relation paths (Lin et al., 2015a). Wang et al. (2017) comprehensively summarize the current popular knowledge graph embedding methods.

Traditional embedding models are heavily rely on rich training instances (Zhang et al., 2019b; Xiong et al., 2018), thus are limited to do few-shot link prediction. Our MetaR is designed to fill this vulnerability of existing embedding models.

2.2 Meta-Learning

Meta-learning seeks for the ability of learning quickly from only a few instances within the same concept and adapting continuously to more concepts, which are actually the rapid and incremental learning that humans are very good at.

Several meta-learning models have been proposed recently. Generally, there are three kinds of meta-learning methods so far: (1) *Metric-based* meta-learning (Koch et al., 2015; Vinyals et al., 2016; Snell et al., 2017; Xiong et al., 2018), which tries to learn a matching metric between query and support set generalized to all tasks, where the idea of matching is similar to some nearest neighbors algorithms. Siamese Neural Network (Koch et al., 2015) is a typical method using symmetric twin networks to compute the metric of two inputs. GMatching (Xiong et al., 2018), the first trial on one-shot link prediction in knowledge graphs, learns a matching metric based on entity embeddings and local graph structures which also can be regarded as a metric-based method. (2) *Model-based* method (Santoro et al., 2016; Munkhdalai and Yu, 2017; Mishra et al., 2018), which uses a specially designed part like memory to achieve the ability of learning rapidly by only a few training instances. MetaNet (Munkhdalai and Yu, 2017), a kind of memory augmented neural network (MANN), acquires meta information from loss gradient and generalizes rapidly via its fast parameterization. (3) *Optimization-based* approach (Finn et al., 2017; Lee and Choi, 2018), which gains the idea of learning faster by changing the optimization algorithm. Model-Agnostic Meta-Learning (Finn et al., 2017) abbreviated as MAML is a model-agnostic algorithm. It firstly updates parameters of task-specific learner, and meta-optimization across tasks is performed over parameters by using above updated parameters, it's like "a gradient through a gradient".

As far as we know, work proposed by Xiong et al. (2018) is the first research on few-shot learning for knowledge graphs. It's a metric-based model which consists of a neighbor encoder and a matching processor. Neighbor encoder enhances the embedding of entities by their one-hop neighbors, and matching processor performs a multi-step matching by a LSTM block.

Training	
Task #1 (CountryCapital)	
Support	(China, CountryCapital, Beijing)
Query	(France, CountryCapital, Paris)
Task #2 (CEOof)	
Support	(Satya Nadella, CEOof, Microsoft)
Query	(Jack Dorsey, CEOof, Twitter)
Testing	
Task #1 (OfficialLanguage)	
Support	(Japan, OfficialLanguage, Japanese)
Query	(Spain, OfficialLanguage, Spanish)

Table 1: The training and testing examples of 1-shot link prediction in KGs.

3 Task Formulation

In this section, we present the formal definition of a knowledge graph and few-shot link prediction task. A knowledge graph is defined as follows:

Definition 3.1 *(Knowledge Graph $\mathcal{G}$) A knowledge graph $\mathcal{G} = \{\mathcal{E}, \mathcal{R}, \mathcal{TP}\}$. $\mathcal{E}$ is the entity set. $\mathcal{R}$ is the relation set. And $\mathcal{TP} = \{(h, r, t) \in \mathcal{E} \times \mathcal{R} \times \mathcal{E}\}$ is the triple set.*

And a few-shot link prediction task in knowledge graphs is defined as:

Definition 3.2 *(Few-shot link prediction task $\mathcal{T}$) With a knowledge graph $\mathcal{G} = \{\mathcal{E}, \mathcal{R}, \mathcal{TP}\}$, given a support set $\mathcal{S}_r = \{(h_i, t_i) \in \mathcal{E} \times \mathcal{E} | (h_i, r, t_i) \in \mathcal{TP}\}$ about relation $r \in \mathcal{R}$, where $|\mathcal{S}_r| = K$, predicting the tail entity linked with relation r to head entity h_j, formulated as $r : (h_j, ?)$, is called K-shot link prediction.*

As defined above, a few-shot link prediction task is always defined for a specific relation. During prediction, there usually is more than one triple to be predicted, and with *support set* $\mathcal{S}_r$, we call the set of all triples to be predicted as *query set* $\mathcal{Q}_r = \{r : (h_j, ?)\}$.

The goal of a few-shot link prediction method is to gain the capability of predicting new triples about a relation r with only observing a few triples about r. Thus its training process is based on a set of tasks $\mathcal{T}_{train} = \{\mathcal{T}_i\}_{i=1}^{M}$ where each task $\mathcal{T}_i = \{\mathcal{S}_i, \mathcal{Q}_i\}$ corresponds to an individual few-shot link prediction task with its own support and query set. Its testing process is conducted on a set of new tasks $\mathcal{T}_{test} = \{\mathcal{T}_j\}_{j=1}^{N}$ which is similar to $\mathcal{T}_{train}$, other than that $\mathcal{T}_j \in \mathcal{T}_{test}$ should be about relations that have never been seen in $\mathcal{T}_{train}$.

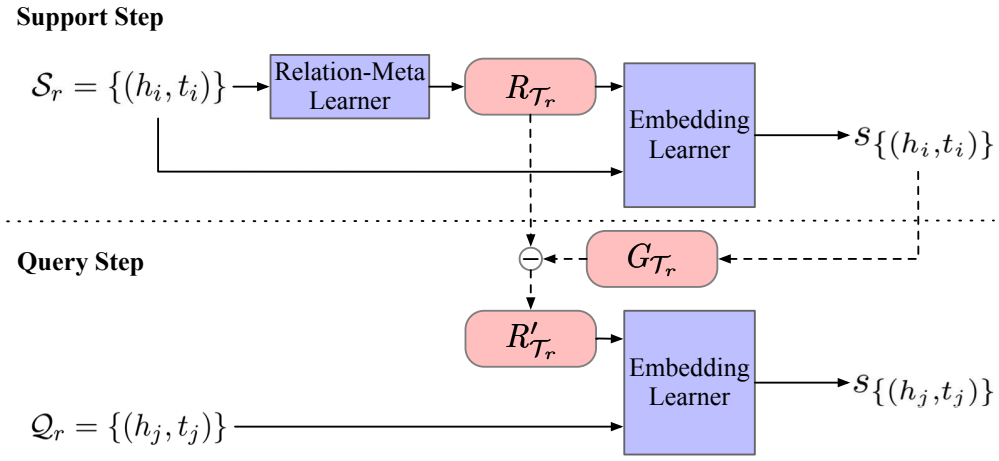

Figure 2: Overview of MetaR. $\mathcal{T}_r = \{\mathcal{S}_r, \mathcal{Q}_r\}$, $R_{\mathcal{T}_r}$ and $R'_{\mathcal{T}_r}$ represent relation meta and updated relation meta, and $G_{\mathcal{T}_r}$ represents gradient meta.

Table 1 gives a concrete example of the data during learning and testing for few-shot link prediction.

4 Method

To make one model gain the few-shot link prediction capability, the most important thing is transferring information from support set to query set and there are two questions for us to think about: (1) what is the most transferable and common information between support set and query set and (2) how to learn faster by only observing a few instances within one task. For question (1), within one task, all triples in support set and query set are about the same relation, thus it is naturally to suppose that relation is the key common part between support and query set. For question (2), the learning process is usually conducted by minimizing a loss function via gradient descending, thus gradients reveal how the model's parameters should be changed. Intuitively, we believe that gradients are valuable source to accelerate learning process.

Based on these thoughts, we propose two kinds of meta information which are shared between support set and query set to deal with above problems:

- **Relation Meta** represents the relation connecting head and tail entities in both support and query set and we extract relation meta for each task, represented as a vector, from support set and transfer it to query set.
- **Gradient Meta** is the loss gradient of relation meta in support set. As gradient meta shows how relation meta should be changed in order to reach a loss minima, thus to accelerate the learning process, relation meta is updated through gradient meta before being transferred to query set. This update can be viewed as the rapid learning of relation meta.

In order to extract relation meta and gradient mate and incorporate them with knowledge graph embedding to solve few-shot link prediction, our proposal, **MetaR**, mainly contains two modules:

- **Relation-Meta Learner** generates relation meta from heads' and tails' embeddings in the support set.
- **Embedding Learner** calculates the truth values of triples in support set and query set via entity embeddings and relation meta. Based on the loss function in embedding learner, gradient meta is calculated and a rapid update for relation meta will be implemented before transferring relation meta to query set.

The overview and algorithm of MetaR are shown in Figure 2 and Algorithm 1. Next, we introduce each module of MetaR via one few-shot link prediction task $\mathcal{T}_r = \{\mathcal{S}_r, \mathcal{Q}_r\}$.

4.1 Relation-Meta Learner

To extract the relation meta from support set, we design a relation-meta learner to learn a mapping from head and tail entities in support set to relation meta. The structure of this relation-meta learner can be implemented as a simple neural network.

In task $\mathcal{T}_r$, the input of relation-meta learner is head and tail entity pairs in support set $\{(h_i, t_i) \in \mathcal{S}_r\}$. We firstly extract entity-pair specific relation

Algorithm 1 Learning of MetaR

Require: Training tasks $\mathcal{T}_{train}$
Require: Embedding layer emb; Parameter of relation-meta learner ϕ
while not done **do**
  Sample a task $\mathcal{T}_r = \{\mathcal{S}_r, \mathcal{Q}_r\}$ from $\mathcal{T}_{train}$
  Get R from $\mathcal{S}_r$ (Equ. 1, Equ. 2)
  Compute loss in $\mathcal{S}_r$ (Equ. 4)
  Get G by gradient of R (Equ. 5)
  Update R by G (Equ. 6)
  Compute loss in $\mathcal{Q}_r$ (Equ. 8)
  Update ϕ and emb by loss in $\mathcal{Q}_r$
end while

meta via a L-layers fully connected neural network,

$$\begin{aligned} \mathbf{x}^0 &= \mathbf{h}_i \oplus \mathbf{t}_i \\ \mathbf{x}^l &= \sigma(\mathbf{W}^l \mathbf{x}^{l-1} + b^l) \\ R_{(h_i,t_i)} &= \mathbf{W}^L \mathbf{x}^{L-1} + b^L \end{aligned} \tag{1}$$

where $\mathbf{h}_i \in \mathbb{R}^d$ and $\mathbf{t}_i \in \mathbb{R}^d$ are embeddings of head entity h_i and tail entity t_i with dimension d respectively. L is the number of layers in neural network, and $l \in \{1, \ldots, L-1\}$. $\mathbf{W}^l$ and $\mathbf{b}^l$ are weights and bias in layer l. We use LeakyReLU for activation σ. $\mathbf{x} \oplus \mathbf{y}$ represents the concatenation of vector $\mathbf{x}$ and $\mathbf{y}$. Finally, $R_{(h_i,t_i)}$ represent the relation meta from specific entity pare h_i and t_i.

With multiple entity-pair specific relation meta, we generate the final relation meta in current task via averaging all entity-pair specific relation meta in current task,

$$R_{\mathcal{T}_r} = \frac{\sum_{i=1}^{K} R_{(h_i,t_i)}}{K} \tag{2}$$

4.2 Embedding Learner

As we want to get gradient meta to make a rapid update on relation meta, we need a score function to evaluate the truth value of entity pairs under specific relations and also the loss function for current task. We apply the key idea of knowledge graph embedding methods in our embedding learner, as they are proved to be effective on evaluating truth value of triples in knowledge graphs.

In task $\mathcal{T}_r$, we firstly calculate the score for each entity pairs (h_i, t_i) in support set $\mathcal{S}_r$ as follows:

$$s_{(h_i,t_i)} = \|\mathbf{h}_i + R_{\mathcal{T}_r} - \mathbf{t}_i\| \tag{3}$$

where $\|\mathbf{x}\|$ represents the L2 norm of vector $\mathbf{x}$. We design the score function inspired by TransE (Bordes et al., 2013) which assumes the head entity embedding $\mathbf{h}$, relation embedding $\mathbf{r}$ and tail entity embedding $\mathbf{t}$ for a true triple (h, r, t) satisfying $\mathbf{h}+\mathbf{r} = \mathbf{t}$. Thus the score function is defined according to the distance between $\mathbf{h} + \mathbf{r}$ and $\mathbf{t}$. Transferring to our few-show link prediction task, we replace the relation embedding $\mathbf{r}$ with relation meta $R_{\mathcal{T}_r}$ as there is no direct general relation embeddings in our task and $R_{\mathcal{T}_r}$ can be regarded as the relation embedding for current task $\mathcal{T}_r$.

With score function for each triple, we set the following loss,

$$L(\mathcal{S}_r) = \sum_{(h_i,t_i) \in \mathcal{S}_r} [\gamma + s_{(h_i,t_i)} - s_{(h_i,t_i')}]_+ \tag{4}$$

where $[x]_+$ represents the positive part of x and γ represents margin which is a hyperparameter. $s_{(h_i,t_i')}$ is the score for negative sample (h_i, t_i') corresponding to current positive entity pair $(h_i, t_i) \in \mathcal{S}_r$, where $(h_i, r, t_i') \notin \mathcal{G}$.

$L(\mathcal{S}_r)$ should be small for task $\mathcal{T}_r$ which represents the model can properly encode truth values of triples. Thus gradients of parameters indicate how should the parameters be updated. Thus we regard the gradient of $R_{\mathcal{T}_r}$ based on $L(\mathcal{S}_r)$ as gradient meta $G_{\mathcal{T}_r}$:

$$G_{\mathcal{T}_r} = \nabla_{R_{\mathcal{T}_r}} L(\mathcal{S}_r) \tag{5}$$

Following the gradient update rule, we make a rapid update on relation meta as follows:

$$R'_{\mathcal{T}_r} = R_{\mathcal{T}_r} - \beta G_{\mathcal{T}_r} \tag{6}$$

where β indicates the step size of gradient meta when operating on relation meta.

When scoring the query set by embedding learner, we use updated relation meta. After getting the updated relation meta R', we transfer it to samples in query set $\mathcal{Q}_r = \{(h_j, t_j)\}$ and calculate their scores and loss of query set, following the same way in support set:

$$s_{(h_j,t_j)} = \|\mathbf{h}_j + R'_{\mathcal{T}_r} - \mathbf{t}_j\| \tag{7}$$

$$L(\mathcal{Q}_r) = \sum_{(h_j,t_j) \in \mathcal{Q}_r} [\gamma + s_{(h_j,t_j)} - s_{(h_j,t_j')}]_+ \tag{8}$$

where $L(\mathcal{Q}_r)$ is our training objective to be minimized. We use this loss to update the whole model.

Dataset	Fit	# Train	# Dev	# Test
NELL-One	Y	321	5	11
Wiki-One	Y	589	16	34
NELL-One	N	51	5	11
Wiki-One	N	133	16	34

Table 2: Statistic of datasets. Fit denotes fitting background into training tasks (Y) or not (N), # Train, # Dev and # Test denote the number of relations in training, validation and test set.

Background Conf.	Description
BG:Pre-Train	Use background to train entity embedding in advance.
BG:In-Train	Fit background graph into training tasks.
BG:Discard	Discard the background graph.

Table 3: Three forms of datasets in our experiments.

4.3 Training Objective

During training, our objective is to minimize the following loss L which is the sum of query loss for all tasks in one minibatch:

$$L = \sum_{(\mathcal{S}_r, \mathcal{Q}_r) \in \mathcal{T}_{train}} L(\mathcal{Q}_r) \tag{9}$$

5 Experiments

With MetaR, we want to figure out following things: 1) can MetaR accomplish few-shot link prediction task and even perform better than previous model? 2) how much relation-specific meta information contributes to few-shot link prediction? 3) is there any requirement for MetaR to work on few-shot link prediction? To do these, we conduct the experiments on two few-shot link prediction datasets and deeply analyze the experiment results [1].

5.1 Datasets and Evaluation Metrics

We use two datasets, NELL-One and Wiki-One which are constructed by Xiong et al. (2018). NELL-One and Wiki-One are derived from NELL (Carlson et al., 2010) and Wikidata (Vrandei and Krtzsch, 2014) respectively. Furthermore, because these two benchmarks are firstly tested on GMatching which consider both learned embeddings and one-hop graph structures, a background graph is constructed with relations out of training/validation/test sets for obtaining the pre-train entity embeddings and providing the local graph for GMatching.

Unlike GMatching using background graph to enhance the representations of entities, our MetaR can be trained without background graph. For NELL-One and Wiki-One which have background graph originally, we can make use of such background graph by fitting it into training tasks or using it to train embeddings to initialize entity representations. Overall, we have three kinds of dataset settings, shown in Table 3. For setting of *BG:In-Train*, in order to make background graph included in training tasks, we sample tasks from triples in background graph and original training set, rather than sampling from only original training set.

Note that these three settings don't violate the task formulation of few-shot link prediction in KGs. The statistics of NELL-One and Wiki-One are shown in Table 2.

We use two traditional metrics to evaluate different methods on these datasets, MRR and Hits@N. MRR is the mean reciprocal rank and Hits@N is the proportion of correct entities ranked in the top N in link prediction.

5.2 Implementation

During training, mini-batch gradient descent is applied with batch size set as 64 and 128 for NELL-One and Wiki-One respectively. We use Adam (Kingma and Ba, 2015) with the initial learning rate as 0.001 to update parameters. We set $\gamma = 1$ and $\beta = 1$. The number of positive and negative triples in query set is 3 and 10 in NELL-One and Wiki-One. Trained model will be applied on validation tasks each 1000 epochs, and the current model parameters and corresponding performance will be recorded, after stopping, the model that has the best performance on Hits@10 will be treated as final model. For number of training epoch, we use early stopping with 30 patient epochs, which means that we stop the training when the performance on Hits@10 drops 30 times continuously. Following GMatching, the embedding dimension of NELL-One is 100 and Wiki-One is 50. The

[1]The source code of experiments is available at https://github.com/AnselCmy/MetaR

	MRR		Hits@10		Hits@5		Hits@1	
NELL-One	1-shot	5-shot	1-shot	5-shot	1-shot	5-shot	1-shot	5-shot
GMatching_RESCAL	.188	–	.305	–	.243	–	.133	–
GMatching_TransE	.171	–	.255	–	.210	–	.122	–
GMatching_DistMult	.171	–	.301	–	.221	–	.114	–
GMatching_ComplEx	.185	.201	.313	.311	.260	.264	.119	.143
GMatching_Random	.151	–	.252	–	.186	–	.103	–
MetaR (BG:Pre-Train)	.164	.209	.331	.355	.238	.280	.093	.141
MetaR (BG:In-Train)	**.250**	**.261**	**.401**	**.437**	**.336**	**.350**	**.170**	**.168**
Wiki-One	1-shot	5-shot	1-shot	5-shot	1-shot	5-shot	1-shot	5-shot
GMatching_RESCAL	.139	–	.305	–	.228	–	.061	–
GMatching_TransE	.219	–	.328	–	.269	–	.163	–
GMatching_DistMult	.222	–	.340	–	.271	–	.164	–
GMatching_ComplEx	.200	–	.336	–	.272	–	.120	–
GMatching_Random	.198	–	.299	–	.260	–	.133	–
MetaR (BG:Pre-Train)	**.314**	**.323**	**.404**	**.418**	**.375**	**.385**	**.266**	**.270**
MetaR (BG:In-Train)	.193	.221	.280	.302	.233	.264	.152	.178

Table 4: Results of few-shot link prediction in NELL-One and Wiki-One. **Bold** numbers are the best results of all and underline numbers are the best results of GMatching. The contents of (bracket) after MetaR illustrate the form of datasets we use for MetaR.

sizes of two hidden layers in relation-meta learner are 500, 200 and 250, 100 for NELL-One and Wiki-One.

5.3 Results

The results of two few-shot link prediction tasks, including 1-shot and 5-shot, on NELL-One and Wiki-One are shown in Table 4. The baseline in our experiment is GMatching (Xiong et al., 2018), which made the first trial on few-shot link prediction task and is the only method that we can find as baseline. In this table, results of GMatching with different KG embedding initialization are copied from the original paper. Our MetaR is tested on different settings of datasets introduced in Table 3.

In Table 4, our model performs better with all evaluation metrics on both datasets. Specifically, for 1-shot link prediction, MetaR increases by 33%, 28.1%, 29.2% and 27.8% on MRR, Hits@10, Hits@5 and Hits@1 on NELL-One, and 41.4%, 18.8%, 37.9% and 62.2% on Wiki-One, with average improvement of 29.53% and 40.08% respectively. For 5-shot, MetaR increases by 29.9%, 40.5%, 32.6% and 17.5% on MRR, Hits@10, Hits@5 and Hits@1 on NELL-One with average improvement of 30.13%.

Thus for the first question we want to explore, the results of MetaR are no worse than GMatching, indicating that MetaR has the capability of accomplishing few-shot link prediction. In parallel, the impressive improvement compared with GMatching demonstrates that the key idea of MetaR, transferring relation-specific meta information from support set to query set, works well on few-shot link prediction task.

Ablation Conf.	BG:Pre-Train	BG:In-Train
standard	.331	.401
-g	.234	.341
-g -r	.052	.052

Table 5: Results of ablation study on Hits@10 of 1-shot link prediction in NELL-One.

Furthermore, compared with GMatching, our MetaR is independent with background knowledge graphs. We test MetaR on 1-shot link prediction in partial NELL-One and Wiki-One which discard the background graph, and get the results of 0.279 and 0.348 on Hits@10 respectively. Such results are still comparable with GMatching in fully datasets with background.

5.4 Ablation Study

We have proved that relation-specific meta information, the key point of MetaR, successfully contributes to few-shot link prediction in previous section. As there are two kinds of relation-specific meta information in this paper, relation meta and gradient meta, we want to figure out how these two kinds of meta information contribute to the performance. Thus, we conduct an ablation study with three settings. The first one is our complete MetaR method denoted as *standard*. The second one is removing the gradient meta by transferring un-updated relation meta directly from support set to query set without updating it via gradient meta, denoted as *-g*. The third one is removing the relation meta further which makes the model rebase to a simple TransE embedding model, denoted as *-g -r*. The result under the third setting is copied from Xiong et al. (2018). It uses the triples from background graph, training tasks and one-shot training triples from validation/test set, so it's neither *BG:Pre-Train* nor *BG:In-Train*. We conduct the ablation study on NELL-one with metric Hit@10 and results are shown in Table 5.

Table 5 shows that removing gradient meta decreases 29.3% and 15% on two dataset settings, and further removing relation meta continuous decreases the performance with 55% and 72% compared to the *standard* results. Thus both relation meta and gradient meta contribute significantly and relation meta contributes more than gradient meta. Without gradient meta and relation meta, there is no relation-specific meta information transferred in the model and it almost doesn't work. This also illustrates that relation-specific meta information is important and effective for few-shot link prediction task.

5.5 Facts That Affect MetaR's Performance

We have proved that both relation meta and gradient meta surely contribute to few-shot link prediction. But is there any requirement for MetaR to ensure the performance on few-shot link prediction? We analyze this from two points based on the results, one is the sparsity of entities and the other is the number of tasks in training set.

The sparsity of entities We notice that the best result of NELL-One and Wiki-One appears in different dataset settings. With NELL-One, MetaR performs better on *BG:In-Train* dataset setting, while with Wiki-One, it performs better on *BG:Pre-Train*. Performance difference between two dataset settings is more significant on Wiki-One.

Most datasets for few-shot task are sparse and the same with NELL-One and Wiki-One, but the entity sparsity in these two datasets are still significantly different, which is especially reflected in the proportion of entities that only appear in one triple in training set, 82.8% and 37.1% in Wiki-One and NELL-One respectively. Entities only have one triple during training will make MetaR unable to learn good representations for them, because entity embeddings heavily rely on triples related to them in MetaR. Only based on one triple, the learned entity embeddings will include a lot of bias. Knowledge graph embedding method can learn better embeddings than MetaR for those one-shot entities, because entity embeddings can be corrected by embeddings of relations that connect to it, while they can't in MetaR. This is why the best performance occurs in *BG:Pre-train* setting on Wiki-One, pre-train entity embeddings help MetaR overcome the low-quality on one-shot entities.

The number of tasks From the comparison of MetaR's performance between with and without background dataset setting on NELL-One, we find that the number of tasks will affect MetaR's performance significantly. With *BG:In-Train*, there are 321 tasks during training and MetaR achieves 0.401 on Hits@10, while without background knowledge, there are 51, with 270 less, and MetaR achieves 0.279. This makes it reasonable that why MetaR achieves best performance on *BG:In-Train* with NELL-One. Even NELL-One has 37.1% one-shot entities, adding background knowledge into dataset increases the number of training tasks significantly, which complements the sparsity problem and contributes more to the task.

Thus we conclude that both the sparsity of entities and number of tasks will affect performance of MetaR. Generally, with more training tasks, MetaR performs better and for extremely sparse dataset, pre-train entity embeddings are preferred.

6 Conclusion

We propose a meta relational learning framework to do few-shot link prediction in KGs, and we design our model to transfer relation-specific meta information from support set to query set. Specif-

ically, using relation meta to transfer common and important information, and using gradient meta to accelerate learning. Compared to GMatching which is the only method in this task, our method MetaR gets better performance and it is also independent with background knowledge graphs. Based on experimental results, we analyze that the performance of MetaR will be affected by the number of training tasks and sparsity of entities. We may consider obtaining more valuable information about sparse entities in few-shot link prediction in KGs in the future.

Acknowledgments

We want to express gratitude to the anonymous reviewers for their hard work and kind comments, which will further improve our work in the future. This work is funded by NSFC 91846204/61473260, national key research program YS2018YFB140004, and Alibaba CangJingGe(Knowledge Engine) Research Plan.

References

Kurt Bollacker, Colin Evans, Praveen Paritosh, Tim Sturge, and Jamie Taylor. 2008. Freebase: A collaboratively created graph database for structuring human knowledge. In *Proceedings of the 2008 ACM SIGMOD International Conference on Management of Data*, pages 1247–1250. ACM.

Antoine Bordes, Nicolas Usunier, Alberto Garcia-Duran, Jason Weston, and Oksana Yakhnenko. 2013. Translating embeddings for modeling multi-relational data. In *Advances in neural information processing systems*, pages 2787–2795.

Antoine Bordes, Jason Weston, and Nicolas Usunier. 2014. Open question answering with weakly supervised embedding models. In *Joint European conference on machine learning and knowledge discovery in databases*, pages 165–180. Springer.

Andrew Carlson, Justin Betteridge, Bryan Kisiel, Burr Settles, Estevam R Hruschka, and Tom M Mitchell. 2010. Toward an architecture for never-ending language learning. In *Twenty-Fourth AAAI Conference on Artificial Intelligence*.

Tim Dettmers, Pasquale Minervini, Pontus Stenetorp, and Sebastian Riedel. 2018. Convolutional 2d knowledge graph embeddings. In *Thirty-Second AAAI Conference on Artificial Intelligence*.

Chelsea Finn, Pieter Abbeel, and Sergey Levine. 2017. Model-agnostic meta-learning for fast adaptation of deep networks. In *International Conference on Machine Learning*, pages 1126–1135.

Diederik P. Kingma and Jimmy Ba. 2015. Adam: A method for stochastic optimization. In *International Conference on Learning Representations*.

Gregory Koch, Richard Zemel, and Ruslan Salakhutdinov. 2015. Siamese neural networks for one-shot image recognition. In *ICML deep learning workshop*, volume 2.

Yoonho Lee and Seungjin Choi. 2018. Gradient-based meta-learning with learned layerwise metric and subspace. In *International Conference on Machine Learning*, pages 2933–2942.

Yankai Lin, Zhiyuan Liu, Huanbo Luan, Maosong Sun, Siwei Rao, and Song Liu. 2015a. Modeling relation paths for representation learning of knowledge bases. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 705–714.

Yankai Lin, Zhiyuan Liu, Maosong Sun, Yang Liu, and Xuan Zhu. 2015b. Learning entity and relation embeddings for knowledge graph completion. In *Twenty-Ninth AAAI Conference on Artificial Intelligence*.

Nikhil Mishra, Mostafa Rohaninejad, Xi Chen, and Pieter Abbeel. 2018. A simple neural attentive meta-learner. In *International Conference on Learning Representations*.

Tsendsuren Munkhdalai and Hong Yu. 2017. Meta networks. In *International Conference on Machine Learning*, pages 2554–2563.

Maximilian Nickel, Volker Tresp, and Hans-Peter Kriegel. 2011. A three-way model for collective learning on multi-relational data. In *International Conference on Machine Learning*, volume 11, pages 809–816.

Adam Santoro, Sergey Bartunov, Matthew Botvinick, Daan Wierstra, and Timothy Lillicrap. 2016. Meta-learning with memory-augmented neural networks. In *International Conference on Machine Learning*, pages 1842–1850.

Jake Snell, Kevin Swersky, and Richard Zemel. 2017. Prototypical networks for few-shot learning. In *Advances in Neural Information Processing Systems*, pages 4077–4087.

Théo Trouillon, Johannes Welbl, Sebastian Riedel, Éric Gaussier, and Guillaume Bouchard. 2016. Complex embeddings for simple link prediction. In *International Conference on Machine Learning*, pages 2071–2080.

Oriol Vinyals, Charles Blundell, Timothy Lillicrap, Daan Wierstra, et al. 2016. Matching networks for one shot learning. In *Advances in Neural Information Processing Systems*, pages 3630–3638.

Denny Vrandei and Markus Krtzsch. 2014. Wikidata: A free collaborative knowledge base. *Communications of the ACM*, 57:78–85.

Quan Wang, Zhendong Mao, Bin Wang, and Li Guo. 2017. Knowledge graph embedding: A survey of approaches and applications. *IEEE Transactions on Knowledge and Data Engineering*, 29(12):2724–2743.

Zhen Wang, Jianwen Zhang, Jianlin Feng, and Zheng Chen. 2014. Knowledge graph embedding by translating on hyperplanes. In *Twenty-Eighth AAAI Conference on Artificial Intelligence*.

Ruobing Xie, Zhiyuan Liu, and Maosong Sun. 2016. Representation learning of knowledge graphs with hierarchical types. In *IJCAI*, pages 2965–2971.

Wenhan Xiong, Mo Yu, Shiyu Chang, Xiaoxiao Guo, and William Yang Wang. 2018. One-shot relational learning for knowledge graphs. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 1980–1990. Association for Computational Linguistics.

Bishan Yang, Wen-tau Yih, Xiaodong He, Jianfeng Gao, and Li Deng. 2015. Embedding entities and relations for learning and inference in knowledge bases. In *International Conference on Learning Representations*.

Fuzheng Zhang, Nicholas Jing Yuan, Defu Lian, Xing Xie, and Wei-Ying Ma. 2016. Collaborative knowledge base embedding for recommender systems. In *Proceedings of the 22nd ACM SIGKDD international conference on knowledge discovery and data mining*, pages 353–362. ACM.

Ningyu Zhang, Shumin Deng, Zhanlin Sun, Guanying Wang, Xi Chen, Wei Zhang, and Huajun Chen. 2019a. Long-tail relation extraction via knowledge graph embeddings and graph convolution networks. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 3016–3025.

Wen Zhang, Bibek Paudel, Liang Wang, Jiaoyan Chen, Hai Zhu, Wei Zhang, Abraham Bernstein, and Huajun Chen. 2019b. Iteratively learning embeddings and rules for knowledge graph reasoning. In *The World Wide Web Conference*, pages 2366–2377. ACM.

Wen Zhang, Bibek Paudel, Wei Zhang, Abraham Bernstein, and Huajun Chen. 2019c. Interaction embeddings for prediction and explanation in knowledge graphs. In *Proceedings of the Twelfth ACM International Conference on Web Search and Data Mining*, pages 96–104. ACM.