Few-Shot Knowledge Graph Completion

Chuxu Zhang,[1] **Huaxiu Yao,**[2] **Chao Huang,**[3] **Meng Jiang,**[1] **Zhenhui Li,**[2] **Nitesh V. Chawla**[1]
[1]University of Notre Dame, [2]Pennsylvania State University, [3]JD Finance American Corporation
[1]{czhang11, mjiang2, nchawla}@nd.edu, [2]{huaxiuyao, jessieli}@psu.edu, [3]chaohuang75@gmail.com

Abstract

Knowledge graphs (KGs) serve as useful resources for various natural language processing applications. Previous KG completion approaches require a large number of training instances (i.e., head-tail entity pairs) for every relation. The real case is that for most of the relations, very few entity pairs are available. Existing work of one-shot learning limits method generalizability for few-shot scenarios and does not fully use the supervisory information; however, few-shot KG completion has not been well studied yet. In this work, we propose a novel few-shot relation learning model (FSRL) that aims at discovering facts of new relations with few-shot references. FSRL can effectively capture knowledge from heterogeneous graph structure, aggregate representations of few-shot references, and match similar entity pairs of reference set for every relation. Extensive experiments on two public datasets demonstrate that FSRL outperforms the state-of-the-art.

Introduction

Large-scale knowledge graphs (KGs) such as YAGO (Suchanek, Kasneci, and Weikum 2007), NELL (Carlson et al. 2010), and Wikidata (Vrandečić and Krötzsch 2014) usually represent facts in the form of relations (edges) between (head-tail) entity pairs (nodes). This kind of graph-structured knowledge is essential for many downstream applications such as search, question answering, and semantic web. However, KGs are known for their incompleteness. In order to automate the KG completion process, many work (Nickel, Tresp, and Kriegel 2011; Bordes et al. 2013; Socher et al. 2013; Yang et al. 2015; Trouillon et al. 2016; Schlichtkrull et al. 2018; Dettmers et al. 2018) have been proposed to infer missing relations by learning existing ones. For example, RESCAL (Nickel, Tresp, and Kriegel 2011) employs tensor factorization to capture inherent structure of multi-relational data in KGs. TransE (Bordes et al. 2013) interprets relations as translation operation on the low-dimensional embeddings of entities. And recently, G-GCN (Schlichtkrull et al. 2018) models relational structure by graph neural network.

The above methods need a good number of entity pairs for every relation. However, the frequency distributions of relations in real datasets often have long tails. A large portion of relations have only few entity pairs in KGs. It is important and challenging to deal with the relations with limited (few-shot) number of entity pairs. The few-shot scenario incurs the infeasibility of previous models which assume available, sufficient training instances for all relations.

In light of the above issue, Xiong *et al.* (2018) proposed GMatching which introduces a local neighbor encoder to learn entity embeddings. It achieves considerable performance in one-shot relation inference yet still has some limitations. First, GMatching assumes all local neighbors contribute equally to the entity embedding, whereas heterogeneous neighbors could have different impacts. For example, the embedding of "Nadella" may have more influence on the embedding of "Microsoft" than "Apple" as the company has only one CEO and a number of competitors. Thus the neighbor encoder of GMatching learns insufficient graph structure representation and impairs the model performance. Second, GMatching is designed under one-shot learning setting. Although it can be modified to few-shot case by adding a pooling layer over reference set, the general operation ignores the interaction among few-shot reference instances and limits the representation capability of reference set. Therefore, it is crucial to design a model to effectively complete relations with limited reference entity pairs.

To address the above weak points, we propose a Few-Shot Relation Learning model (FSRL) with the purpose of learning a matching function that can effectively infer the true entity pairs given the set of few-shot reference entity pairs for each relation. To be more specific, first, we propose a relation-aware heterogeneous neighbor encoder to learn entity embeddings based on the heterogeneous graph structure and attention mechanism. It captures both different relation types and impact differences of local neighbors. Next, we design a recurrent autoencoder aggregation network to model interactions of few-shot reference entity pairs and accumulate their expression capabilities for each relation. With the aggregated embedding of reference set, we finally employ a matching network to discover similar entity pairs of reference set. The meta-training based gradient descent approach is employed to optimize model parameters. The learned model can be further applied to infer true entity pairs for any new relation without any fine-tuning step.

To summarize, our main contributions are:

- We introduce a new few-shot KG completion problem which is different from previous work and more suitable for practical scenarios.
- We propose a few-shot relation learning model to solve the problem. The model performs joint optimization of several learnable neural network modules.
- We conduct extensive experiments on two public datasets. Results demonstrate that our model outperforms state-of-the-art baselines.

Related Work

Here we survey two topics relevant to this work: few-shot learning and relation learning for KGs.

Few-Shot Learning

Recent few-shot learning models have two categories: (1) metric based approaches (Koch, Zemel, and Salakhutdinov 2015; Vinyals et al. 2016; Snell, Swersky, and Zemel 2017; Mishra et al. 2018); (2) meta-optimizer based approaches (Ravi and Larochelle 2016; Finn, Abbeel, and Levine 2017; Li et al. 2017; Finn, Xu, and Levine 2018; Lee and Choi 2018; Yao et al. 2019b). The former one learns an effective metric and corresponding matching function among a set of training instances. For example, matching networks (Vinyals et al. 2016) make predictions by comparing the input example with a few-shot labeled support set. Prototypical networks (Snell, Swersky, and Zemel 2017) classify each sample by computing the distance to prototype representation of each class. The later one aims to quickly optimize the model parameters given the gradients on few-shot data instances. One example is the model-agnostic meta-learning (MAML) (Finn, Abbeel, and Levine 2017) which trains model via a small number of gradient updates and leads to fast learning on a new task. Another example is the LSTM-based meta-learner (Ravi and Larochelle 2016) that learns the exact optimization algorithm used to train another neural network classifier in the few-shot regime. Unlike the previous few-shot learning study that focus on vision (Yang et al. 2018), imitation learning (Duan et al. 2017), spatiotemporal analysis (Yao et al. 2019a), sentiment analysis (Li et al. 2019) domains, we leverage few-shot learning to complete KGs.

Relation Learning for KGs

Many work have been proposed to model relational structure in KGs and automate KG completion. For example, Nickel *et al.* (2011) designed RESCAL to model inherent structure of dyadic relational data by tensor factorization. Bordes *et al.* (2013) proposed TransE that interprets relationships as translation operating on the low-dimensional embeddings of the entities. Unlike representing entities with single vectors, Socher *et al.* (2013) developed NTN that represents entities as an average of their constituting word vectors. Later, more sophisticated models have been proposed, such as DistMul (Yang et al. 2015) and ComplEx (Trouillon et al. 2016). Recently, deep neural network based models like R-GCG (Schlichtkrull et al. 2018) and ConvE (Dettmers et al. 2018) have been presented for further improvement. Different from those models that assume sufficient training instances are available, Xiong *et al.* (2018) presented GMatching model for one-shot relation learning in KGs. In this work, we study a practical few-shot scenario which deals with long tail or newly added relations with few-shot reference instances.

Preliminaries

In this section, we formally define the few-shot knowledge graph completion problem and detail the corresponding few-shot learning settings.

Problem Definition

A KG G is represented as a collection of triples $\{(h, r, t)\} \subseteq \mathcal{E} \times \mathcal{R} \times \mathcal{E}$, where $\mathcal{E}$ and $\mathcal{R}$ denote the entity set and relation set, respectively. The KG completion task is to either predict the tail entity t given the head entity h and the query relation r: $(h, r, ?)$, or predict unseen relation r between head entity and tail entity: $(h, ?, t)$. In this work, we focus on the former case as we want to predict the unseen facts of a given relation. Unlike previous studies that assume enough entity pairs are available for each relation, this work considers a practical scenario that few-shot entity pairs (reference set) are given. The purpose is to rank the true entity t_{true} higher than false candidate entities t_{false}, given few-shot reference pairs $(h_k, t_k) \in R_r$ of relation r. Formally, the problem is defined as follows:

Problem 1 Few-Shot Knowledge Graph Completion *Given the relation* r *and its few-shot reference entity pairs* $(h_k, t_k) \in R_r$*, the task is to design a machine learning model which ranks all tail candidate entities* t *for each new head entity* h*, such that the top ranked* t *are true tail entities of* h*.*

The candidate entities set is constructed based on the entity type constraint (Xiong et al. 2018), and we only consider a closed set of entities which excludes the unseen entities when predicting facts of new relations in test period.

Few-Shot Learning Settings

The purpose of this work is to design a machine learning model which could be utilized to predict the new facts with few-shot reference instances. Following the standard few-shot learning settings (Ravi and Larochelle 2016; Snell, Swersky, and Zemel 2017), we can access to a set of training tasks. In the problem, each training task corresponds to a KG relation $r \in \mathcal{R}$ with its own training/testing entity pairs data: $D_r = \left\{P_r^{train}, P_r^{test}\right\}$. We denote this kind of task set as meta-training set, $\mathcal{T}_{mtr}$. To imitate the few-shot relation prediction in evaluation period, each P_r^{train} only contains few-shot entity pairs $(h_k, t_k) \in R_r$. Besides, $P_r^{test} = \{(h_i, t_i, C_{h_i,r}) | (h_i, r, t_i) \in G\}$ contains all testing entity pairs of r, including true tail entities t_i of each query (h_i, r) and the remaining candidate entities $t_j \in C_{h_i,r}$ where t_j is an entity in G. The proposed model thus could be tested on this set by ranking all candidate entities given the test query (h_i, r) and the few-shot reference pairs in P_r^{train}. We denote the ranking loss of relation r as $\mathcal{L}_\Theta(h_i, t_i|C_{h_i,r}, P_r^{train})$, where Θ is the set of model parameters. Thus, the objective of model training is defined as:

$$min_\Theta \mathbb{E}_{\mathcal{T}_{mtr}} \left[\sum_{(h_i,t_i,C_{h_i,r})\in P_r^{test}} \frac{\mathcal{L}_\Theta(h_i, t_i|C_{h_i,r}, P_r^{train})}{|P_r^{test}|} \right] \tag{1}$$

where $|P_r^{test}|$ represents the number of tuples in P_r^{test}. In next section, we will detail how to formulate and optimize the above objective function.

After sufficient training, the learned model can be utilized to predict facts of each new relation $r' \in \mathcal{R}'$. This step is called the meta-testing. The relations in meta-testing are unseen from meta-training, i.e., $\mathcal{R} \cap \mathcal{R}' = \phi$. The same as meta-training relations, each relation r' in meta-testing has its own few-shot training data $P_{r'}^{train}$ and testing data $P_{r'}^{test}$. These relations form a meta-testing set which is denoted as $\mathcal{T}_{mte}$. In addition, we leave out a subset of relations in $\mathcal{T}_{mtr}$ as the meta-validation set $\mathcal{T}_{mtv}$. Furthermore, the model can access to a background KG G', which is a subset of G that excludes all the relations in $\mathcal{T}_{mtr}$, $\mathcal{T}_{mte}$ and $\mathcal{T}_{mtv}$.

Model

In this section, we present the detail of FSRL. FSRL consists of three major parts: (1) encoding heterogeneous neighbors for each entity; (2) aggregating few-shot reference entity pairs for each relation; (3) matching query pairs with reference set for relation prediction. Figure 1 shows the framework of FSRL.

Encoding Heterogeneous Neighbors

Although many work (Nickel, Tresp, and Kriegel 2011; Bordes et al. 2013; Yang et al. 2015) have been proposed to learn entity embeddings by using relational information, Xiong *et al.* (Xiong et al. 2018) demonstrated that explicitly encoding graph local structure (i.e., one-hop neighbors) can benefit relation prediction. The proposed neighbor encoder takes the average of feature representations of all relational neighbors as the embedding of given entity. Despite the desirable performance, it neglects the different impacts of heterogeneous neighbors which may help improve entity embedding (Zhang et al. 2019). In light of this issue, we design a relation-aware heterogeneous neighbor encoder. Specifically, we denote the set of relational neighbors $(relation, entity)$ of given head entity h as $\mathcal{N}_h = \{(r_i, t_i)|(h, r_i, t_i) \in G'\}$, where G' is the background knowledge graph, r_i and t_i represent the i-th relation and corresponding tail entity of h, respectively. The heterogeneous neighbor encoder should be able to encode $\mathcal{N}_h$ and output a feature representation of h by considering different impacts of relational neighbors $(r_i, t_i) \in \mathcal{N}_h$. To achieve this goal, we introduce an attention module and formulate the embedding of h as follows:

$$\begin{aligned} f_\theta(h) &= \sigma\left(\sum_i \alpha_i e_{t_i}\right) \\ \alpha_i &= \frac{exp\left\{u_{rt}^T\left(\mathcal{W}_{rt}(e_{r_i} \oplus e_{t_i}) + b_{rt}\right)\right\}}{\sum_j exp\left\{u_{rt}^T\left(\mathcal{W}_{rt}(e_{r_j} \oplus e_{t_j}) + b_{rt}\right)\right\}} \end{aligned} \tag{2}$$

where σ denotes activation unit (we use Tanh), $\oplus$ represents concatenation operator, $e_{t_i}, e_{r_i} \in \mathbb{R}^{d\times 1}$ are pre-trained embeddings of t_i and r_i. Besides, $u_{rt} \in \mathbb{R}^{d\times 1}$, $\mathcal{W}_{rt} \in \mathbb{R}^{d\times 2d}$ and $b_{rt} \in \mathbb{R}^{d\times 1}$ (d: pre-trained embedding dimension) are learnable parameters. Figure 1(b) illustrates the detail of heterogeneous neighbor encoder. According to Eq. 2, the formulation of $f_\theta(h)$ considers the different impacts of heterogeneous relational neighbors via attention weight α_i and leverages both embeddings of entity t_i and relation r_i to compute α_i.

Aggregating Few-Shot Reference Set

The current models (e.g., GMatching) are not able to model the interactions of few-shot instances in reference set, which limits model capability. Thus, we need to design a module to effectively formulate the aggregated embedding of reference set R_r for each relation r. By applying the neighbor encoder $f_\theta(h)$ to each entity pair $(h_k, t_k) \in R_r$, we can obtain the representation of (h_k, t_k) in the form $\mathcal{E}_{h_k,t_k} = [f_\theta(h_k) \oplus f_\theta(t_k)]$. Learning the representation of a reference set R_r with few-shot entity pairs is challenging as it requires modeling interactions among different entity pairs and accumulating their expression capability. Inspired by the common practices in learning sentence embeddings (Conneau et al. 2017) in natural language processing and aggregating node embeddings (Hamilton, Ying, and Leskovec 2017) in graph neural networks, we tackle the challenge and formulate the embedding of R_r by aggregating representations of all entity pairs in R_r:

$$f_\epsilon(R_r) = \mathcal{AG}_{(h_k,t_k)\in R_r}\left\{\mathcal{E}_{h_k,t_k}\right\} \tag{3}$$

where $\mathcal{AG}$ is an aggregation function which can be pooling operation, feed-forward neural network, etc. Motivated by the recent success of recurrent neural network aggregator in order-invariant problems such as graph embedding (Hamilton, Ying, and Leskovec 2017), we design a recurrent autoencoder aggregator which achieves good capability. Specifically, the entity pair embeddings $\mathcal{E}_{h_k,t_k} \in R_r$ are sequentially fed into a recurrent autoencoder by:

$$\mathcal{E}_{h_1,t_1} \to m_1 \to \cdots \to m_K \to d_K \to \cdots \to d_1 \tag{4}$$

where K is the size of reference set (i.e., few-shot size). The hidden states m_k and d_k of encoder and decoder are computed by:

$$\begin{aligned} m_k &= \text{RNN}_{encoder}(\mathcal{E}_{h_k,t_k}, m_{k-1}) \\ d_{k-1} &= \text{RNN}_{decoder}(d_k) \end{aligned} \tag{5}$$

where $\text{RNN}_{encoder}$ and $\text{RNN}_{decoder}$ represent recurrent encoder and decoder (e.g., LSTM (Hochreiter and Schmidhuber 1997)), respectively. The reconstruction loss for optimizing autoencoder is defined as:

$$\mathcal{L}_{re}(R_r) = \sum_k \|d_k - \mathcal{E}_{h_k,t_k}\|_2^2 \tag{6}$$

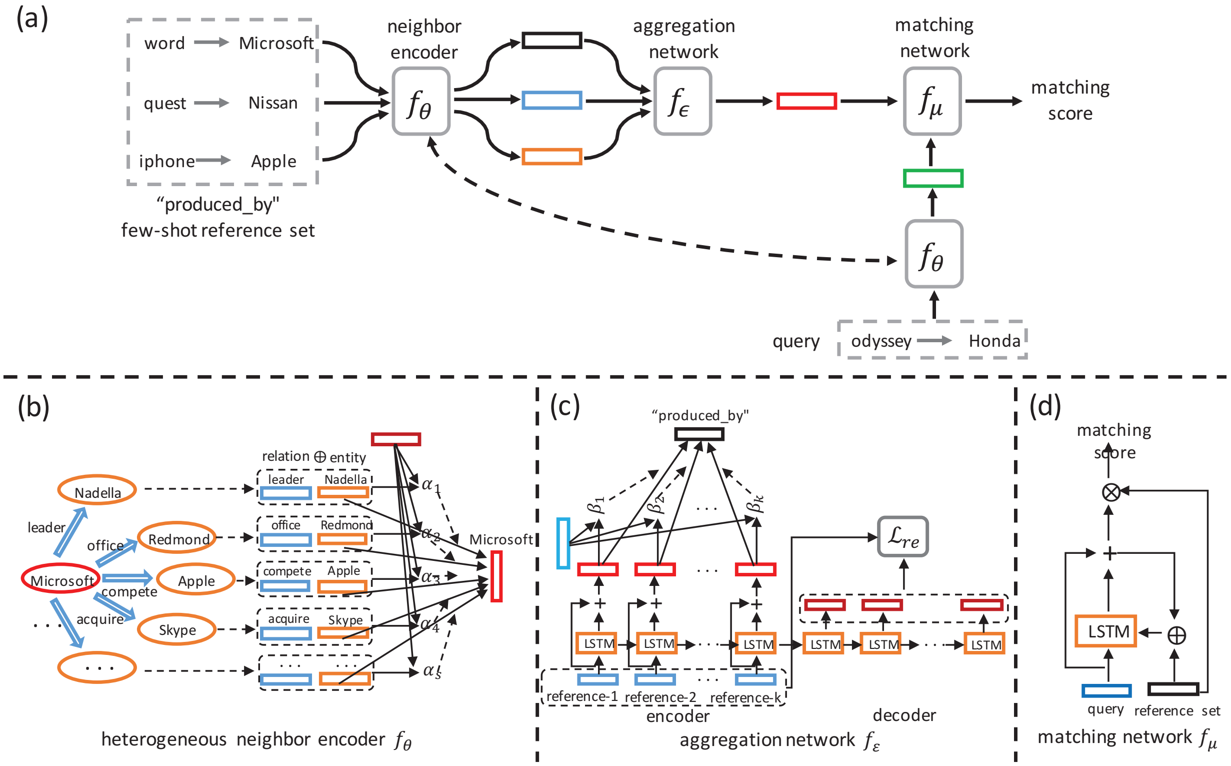

Figure 1: (a) The framework of FSRL: it first generates entity embedding via heterogeneous neighbor encoder, then aggregates few-shot reference entity pairs and generate reference set embedding, finally employs a matching network to compute similarity score between query pair and reference set; (b) the relation-aware heterogeneous neighbor encoder for entity; (c) the recurrent autoencoder aggregation network for reference set; (d) the recurrent matching network for query pair and reference set.

$\mathcal{L}_{re}$ will be incorporated to the relational ranking loss for refining the representation of each entity pair, as we will describe later. In order to formulate the embedding of reference set, we aggregate all hidden states of encoder and extend them by adding residual connection (He et al. 2016) and attention weight. Formally, $f_\epsilon(R_r)$ is computed by:

$$\begin{aligned} m'_k &= m_k + \mathcal{E}_{h_k,t_k} \\ \beta_k &= \frac{exp\left\{u_R^T\left(\mathcal{W}_R m'_k + b_R\right)\right\}}{\sum_{k'} exp\left\{u_R^T\left(\mathcal{W}_R m'_{k'} + b_R\right)\right\}} \\ f_\epsilon(R_r) &= \sum_k \beta_k m'_k \end{aligned} \tag{7}$$

where $u_R \in \mathbb{R}^{d\times 1}$, $\mathcal{W}_R \in \mathbb{R}^{d\times 2d}$ and $b_R \in \mathbb{R}^{d\times 1}$ (d: aggregated embedding dimension) are learnable parameters. Figure 1(c) illustrates the detail of recurrent autoencoder aggregator. The formulation of $f_\epsilon(R_r)$ aggregates all representations of $\mathcal{E}_{h_k,t_k} \in R_r$ and each component in this module will make effect for better performance, as we will show in the ablation study experiment.

Matching Query and Reference Set

With the heterogeneous neighbor encoder f_θ and the reference set aggregator f_ϵ, we now present how to effectively match each query entity pair $(h_l, t_l) \in Q_r$ (Q_r is the set of all query pairs of relation r) with the reference set R_r. By applying f_θ and f_ϵ to the query entity pair (h_l, t_l) and the reference set R_r, we can obtain two embedding vectors $\mathcal{E}_{h_l,t_l} = [f_\theta(h_l) \oplus f_\theta(t_l)]$ and $f_\epsilon(R_r)$, respectively. In order to measure the similarity between two vectors, we employ a recurrent processor (Vinyals et al. 2016) f_μ to perform multiple steps matching. The t-th process step is formulated as:

$$\begin{aligned} g'_t, c_t &= \mathrm{RNN}_{match}(\mathcal{E}_{h_l,t_l}, [g_{t-1} \oplus f_\epsilon(R_r)], c_{t-1}) \\ g_t &= g'_t + \mathcal{E}_{h_l,t_l} \end{aligned} \tag{8}$$

where RNN_{match} is LSTM cell (Hochreiter and Schmidhuber 1997) with input $\mathcal{E}_{h_l,t_l}$, hidden state g_t and cell state c_t. The last hidden state g_T after T "processing" step is the refined embedding of query pair (h_l, t_l): $\mathcal{E}_{h_l,t_l} = g_T$. We use the inner product between $\mathcal{E}_{h_l,t_l}$ and $f_\epsilon(R_r)$ as the similarity score for later ranking optimization procedure. Figure 1(d) shows the detail of matching processor. This module is effective for improving model performance, as we will demonstrate in the ablation study experiment.

Objective and Model Training

For the query relation r, we randomly sample a set of few positive (true) entity pairs $\{(h_k, t_k)|(h_k, r, t_k) \in G\}$ and regard them as the reference set R_r. The remaining positive entity pairs $\mathcal{PE}_r = \{(h_l, t_l)|(h_l, r, t_l) \in G \cap$

Algorithm 1: FSRL Meta-Training

```
input : Meta-training task (relation) set $\mathcal{T}_{mtr}$
        Pre-trained KG embeddings
        Initial model parameters $\theta$, $\epsilon$ and $\mu$
while not done do
    Shuffle tasks (relations) in $\mathcal{T}_{mtr}$
    for $\mathcal{T}_r \in \mathcal{T}_{mtr}$ do
        Sample few-shot entity pairs as reference set
        Sample a batch of query entity pairs $(h_l, t_l)$
        Pollute the tail entity of $(h_l, t_l)$ to get $(h_l, t_l^-)$
        Accumulate the loss by Eq. 10
        Update parameters by Adam optimizer
    end
end
return Optimal model parameters $\theta^*$, $\epsilon^*$ and $\mu^*$
```

$(h_l, t_l) \notin R_r\}$ are utilized as positive query pairs. Besides, we construct a group of negative (false) entity pairs $\mathcal{NE}_r = \{(h_l, t_l^-)|(h_l, r, t_l^-) \notin G\}$ by polluting the tail entities. Therefore the ranking loss is formulated as:

$$\mathcal{L}_{rank} = \sum_{r} \sum_{(h_l,t_l)\in\mathcal{PE}_r} \sum_{(h_l,t_l^-)\in\mathcal{NE}_r} \left[\xi + s_{(h_l,t_l^-)} - s_{(h_l,t_l)}\right]_+ \tag{9}$$

where $[x]_+ = max[0, x]$ is standard hinge loss and ξ is safety margin distance, $s_{(h_l,t_l)}$ and $s_{(h_l,t_l^-)}$ are similarity scores between query pairs $(h_l, t_l/t_l^-)$ and reference set R_r. By leveraging the reconstruction loss $\mathcal{L}_{re}$ of reference set aggregator, we define the final objective function as:

$$\mathcal{L}_{joint} = \mathcal{L}_{rank} + \gamma\mathcal{L}_{re} \tag{10}$$

where γ is trade-off factor between $\mathcal{L}_{rank}$ and $\mathcal{L}_{re}$. To minimize $\mathcal{L}_{joint}$ and optimize model parameters, we take each relation as a task and design a batch sampling based meta-training procedure. The detail of this process is summarized in Algorithm 1.

Experiments

In this section, we conduct extensive experiments to evaluate the performance of proposed model and verify the effectiveness of each component in the model. Few-shot size impact analysis and embedding visualization are also provided.

Experimental Design

Datasets We use two public datasets for experiments. The first one is based on NELL (Mitchell et al. 2018), a system that continuously collects structured knowledge from webs. The second one is based on Wikidata (Vrandečić and Krötzsch 2014). Table 1 lists the statistics of two datasets. The same as (Xiong et al. 2018), we select the relations with less than 500 but more than 50 triples as few-shot tasks. There are 67 and 183 tasks in NELL and Wiki data, respectively. In addition, we use 51/5/11 task relations for training/validation/testing in NELL and the division is set to 133/16/34 in Wiki.

Table 1: Statistics of datasets. # Ent. denotes the number of all unique entities and # triples denotes the number of all relational triples. # Rel. represents the number of all relations and # Tasks represents the number of relations selected as few-shot tasks.

Dataset	# Ent.	# Triples	# Rel.	# Tasks
NELL	68,545	181,109	358	67
Wiki	4,838,244	5,859,240	822	183

Baseline Methods We consider two categories of baseline methods for comparison:

- **Relational embedding methods.** This type of model learns entity/relation embeddings by modeling relational structure in KG. We employ four widely used methods: RESCAL (Nickel, Tresp, and Kriegel 2011), TransE (Bordes et al. 2013), DistMul (Yang et al. 2015), and ComplEx (Trouillon et al. 2016). All entity pairs of background relations and training relations, as well as few-shot training entity pairs of validate and test relations are used to train models.
- **Graph neighbor encoder methods.** This type of model joints graph local neighbor encoder and matching network to learn entity embeddings and predict facts of new relations. We employ state-of-the-art model GMatching (Xiong et al. 2018) for comparison. Note that there are few-shot embeddings of entity pairs in reference set, we use max/mean pooling (denoted as MaxP and MeanP) to obtain the general embedding of reference set. Moreover, we also consider taking the maximum of similarity scores between a query and all K references as the final ranking score of this query. Thus in total, this type of model includes three baseline methods which are denoted as GMatching (MaxP), GMatching (MeanP), and GMatching (Max).

Reproducibility Settings The above relational embedding methods can be utilized to pre-train KG embeddings, which are further used as the input for GMatching and FSRL. We select ComplEx for pre-training as GMatching and FSRL with it achieve best performances in most cases. For the proposed model, we tune hyper-parameters on the validation dataset. The embedding dimension is set to 100 and 50 for NELL and Wiki dataset, respectively. The maximum number of local neighbors in heterogeneous neighbor encoder is set to 30 for both datasets. In addition, we use LSTM as the reference set aggregator and matching processor. The dimension of LSTM's hidden state is set to 200 and 100 for NELL and Wiki dataset, respectively. The number of recurrent steps equals 2 in matching network. We use the Adam optimizer (Kingma and Ba 2015) to update model parameters. The initial learning rate equals 0.001 and the weight decay is 0.25 for each 10k training steps. The margin distance and trade-off factor in the objective function are set to 5.0 and 0.0001, respectively. In entity candidate set construction, we set the maximum size to 1000 for both

Table 2: The overall results of all methods. GMatching is the best baseline. Our model has the best performances in all cases.

	Data: NELL				Data: Wiki			
Model	Hits@1	Hits@5	Hits@10	MRR	Hits@1	Hits@5	Hits@10	MRR
RESCAL	.069/.141	.160/.313	.204/.383	.119/.223	.259/.057	.297/.090	.309/.126	.279/.081
TransE	.056/.119	.112/.256	.189/.320	.104/.193	.186/.069	.352/.134	.431/.176	.273/.111
DistMult	.066/.164	.123/.306	.178/.375	.109/.231	.271/.069	.419/.156	.459/.195	.339/.112
ComplEx	.049/.129	.092/.223	.112/.273	.079/.185	.226/.085	.315/.117	.397/.145	.282/.106
GMatching (MaxP)	.244/.198	.418/.370	.524/.464	.331/.279	.313/.095	.402/.235	.468/.324	.346/.171
GMatching (MeanP)	.257/.186	.455/.360	.542/.453	.341/.267	.290/.128	407/.274	.484/.350	.352/.203
GMatching (Max)	.179/.152	.391/.335	.476/.445	.273/.241	.279/.135	.396/.284	.477/.374	.342/.214
FSRL (Ours)	**.345/.211**	**.502/.433**	**.570/.507**	**.421/.318**	**.338/.155**	**.430/.327**	**.486/.406**	**.390/.241**

datasets. We employ Pytorch[1] to implement our model and further conduct it on a server with GPU machines.

Evaluation Metrics Relations and their entity pairs in training data are utilized to train the model while those of validation and test data are respectively used to tune and evaluate model. We use the top-k hit ratio (Hits@k) and the mean reciprocal rank (MRR) to evaluate performances of different methods. The k is set to 1, 5, and 10. The few-shot size K is set to 3 for the following experiments. In addition, we also conduct experiment to analyze the impact of K.

Results Comparison

Overall Comparison with Baselines The performances of all models are reported in Table 2, where the best results are highlighted in bold and the best baseline results are indicated by underline. The former/later score denotes result in validation/test dataset. According to this table:

- The graph neighbor encoder methods (GMatching) outperform the relational embedding methods, showing that incorporating graph local structure and matching network is effective for learning entity embeddings and predicting facts of new relations.
- FSRL achieves the best performances in all cases. The average relative improvement (%) over the best baseline method is up to 34% and 15% in NELL and Wiki data, respectively. It demonstrates the effectiveness of our model. The heterogeneous neighbor encoder and recurrent autoencoder aggregation network benefit few-shot relation prediction in KGs.

Comparison Over Different Relations Besides the overall performance for all relations, we also conduct experiments to evaluate model performance for each relation in NELL test data. Table 3 reports the results of FSRL and GMatching. The better result for each case is highlighted in bold. According to this table:

- The results of both models on different relations are of high variance. It is reasonable since different relations have different sizes of candidate set for evaluation. The relations (e.g., relation 1) with small candidate set (or are easily to be predicted) have relative large scores and two models in these cases are compared.
- FSRL has better performances than GMatching in most cases. It demonstrates that our model is robust for different relations and outperforms GMatching for most relations.

Table 3: The results of GMatching and FSRL for each relation (RId) in NELL test data.

RId	Model	Hits@1	Hits@5	Hits@10	MRR
1	GMatching	0.946	**1.000**	**1.000**	0.970
	FSRL	**0.972**	0.986	**1.000**	**0.981**
2	GMatching	0.279	0.390	0.451	0.360
	FSRL	**0.972**	**0.972**	**0.986**	**0.975**
3	GMatching	0.069	0.115	0.152	0.108
	FSRL	**0.466**	**0.418**	**0.357**	**0.398**
4	GMatching	0.017	0.033	0.070	0.035
	FSRL	**0.044**	**0.215**	**0.343**	**0.132**
5	GMatching	**0.069**	0.115	0.151	0.107
	FSRL	0.044	**0.215**	**0.343**	**0.132**
6	GMatching	0.192	0.515	0.581	0.338
	FSRL	**0.342**	**0.549**	**0.617**	**0.442**
7	FSRL	**0.478**	0.692	0.804	**0.575**
	GMatching	0.438	**0.716**	**0.842**	0.562
8	GMatching	0.151	0.502	0.669	0.312
	FSRL	**0.201**	**0.543**	**0.681**	**0.347**
9	GMatching	**0.449**	0.707	0.737	**0.564**
	FSRL	0.163	**0.750**	**0.838**	0.408
10	GMatching	0.043	0.129	0.206	0.098
	FSRL	**0.069**	**0.192**	**0.291**	**0.139**
11	GMatching	0.076	**0.708**	0.736	0.341
	FSRL	**0.104**	0.631	**0.781**	**0.416**

Ablation Study

FSRL is a joint learning framework of several neural network modules. To investigate the contributions of different

[1] https://pytorch.org/

components, we conduct the following ablation studies from three perspectives in Table 4, where the results in NELL data are reported and best results are highlighted in bold:

- **(AS_1)** We investigate the effectiveness of relation-aware heterogeneous neighbor encoder. We replace it by a mean pooling layer over all neighbors' embeddings. As shown in the table, our model has much better performances than the variant (in AS_1), indicating the large benefit of heterogeneous neighbor encoder.
- **(AS_2)** We analyze the impacts of different modules of aggregation network in (AS_2a)-(AS_2c). In (AS_2a), we replace the recurrent autoencoder aggregation with mean pooling operation. In (AS_2b), we replace the attention weight of recurrent autoencoder with a mean pooling layer. In (AS_2c), we remove the decoder part, and only use encoder for aggregation. According to the results in table, our model outperforms all of three variants in most cases, demonstrating the effect of each component in aggregation network.
- **(AS_3)** We further analyze the effectiveness of matching network. We remove the LSTM cell and use inner-product between query embedding and reference embedding as similarity (ranking) score. From the results in table, our model largely outperforms the variant (in AS_3), showing that recurrent matching network has good capability in computing relevance between query and reference.

Table 4: Results of model variants in NELL data. Our model has better performance than all model variants.

Model	Hits@1	Hits@5	Hits@10	MRR
(AS_1)	.121/.179	.312/.379	.432/.464	.212/.272
(AS_2a)	.281/.191	.463/.414	.538/.504	.368/.297
(AS_2b)	.298/.219	.480/.420	.556/.498	.382/.315
(AS_2c)	.333/**.226**	.478/.418	.552/.499	.401/.313
(AS_3)	.282/.205	.463/.394	.548/.478	.370/.299
Ours	**.345**/.211	**.502/.433**	**.570/.507**	**.421/.318**

Analysis

Impact of Few-Shot Size This work studies few-shot relation learning in KGs, thus we conduct experiment to analyze the impact of few-shot size K. Figure 2 reports the performances of our model and GMatching (MaxP) in NELL test data with different settings of K. According to the figure:

- With the increment of K, performances of both models increase. It indicates that larger reference set can produce better reference set embedding for the relation.
- Our model consistently outperforms GMatching in different K, demonstrating the stability of the proposed model for few-shot relation completion in KGs.

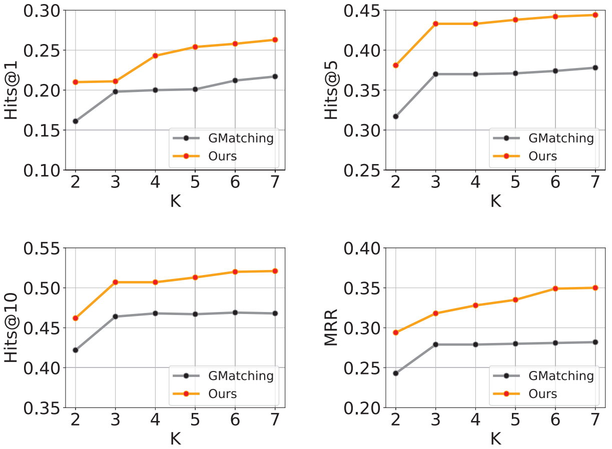

Figure 2: Impact of few-shot size K. Our model consistently outperforms GMatching.

Embedding Visualization To show a better performance comparison between our model and GMatching, we visualize the 2D embeddings of positive and negative candidate entity pairs for each relation. Figure 3 shows the visualization results of our model and GMatching for two test relations of NELL data, i.e., "produced_by" and "team_coach", which vary from each other in semantic meaning and size of positive/negative candidate set. According to the figure, both methods can distinguish embeddings of positive and negative candidates well. However, it is clear that our model achieves better performance and embeddings of two classes are clearly discriminated from each other, which further demonstrates the superior performance of our model in terms of visualization.

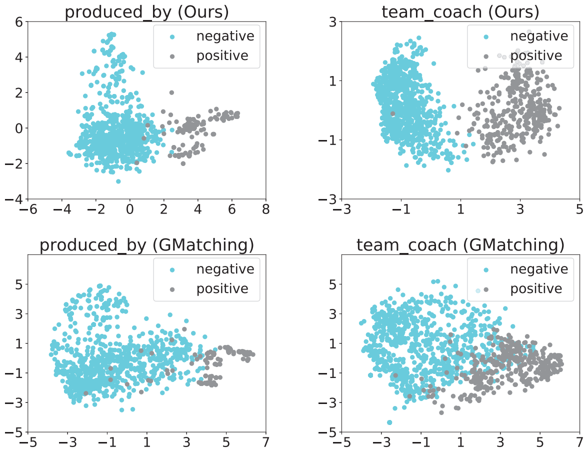

Figure 3: Embedding visualization of positive and negative candidates of two selected relations. Our model can clearly discriminate embeddings of these two types of candidates.

Conclusion

In this paper, we presented a new few-shot KG completion problem and proposed an innovative few-shot relation learning model, i.e., FSRL, to solve the problem. FSRL performs joint optimization of relation-aware heterogeneous neighbor encoder, recurrent autoencoder aggregation network and matching network. The extensive experiments on two public datasets demonstrate that FSRL can outperform state-of-the-art baseline methods. In addition, the ablation studies verify the effectiveness of each model component. As a new research problem, there are many opportunities for the next steps. The future work might consider utilizing a better model training process such as model-agnostic meta-learning or incorporating contextual information such as entity attributes or text description to improve the quality of entity embeddings.

Acknowledgments

This work was supported by the Army Research Laboratory under Cooperative Agreement Number W911NF-09-2-0053 and the National Science Foundation awards #1925607, #1629914, #1652525, #1618448 and #1849816.

References

Bordes, A.; Usunier, N.; Garcia-Duran, A.; Weston, J.; and Yakhnenko, O. 2013. Translating embeddings for modeling multi-relational data. In *NIPS*.

Carlson, A.; Betteridge, J.; Kisiel, B.; Settles, B.; Hruschka, E. R.; and Mitchell, T. M. 2010. Toward an architecture for never-ending language learning. In *AAAI*.

Conneau, A.; Kiela, D.; Schwenk, H.; Barrault, L.; and Bordes, A. 2017. Supervised learning of universal sentence representations from natural language inference data. In *EMNLP*.

Dettmers, T.; Minervini, P.; Stenetorp, P.; and Riedel, S. 2018. Convolutional 2d knowledge graph embeddings. In *AAAI*.

Duan, Y.; Andrychowicz, M.; Stadie, B.; Ho, O. J.; Schneider, J.; Sutskever, I.; Abbeel, P.; and Zaremba, W. 2017. One-shot imitation learning. In *NIPS*.

Finn, C.; Abbeel, P.; and Levine, S. 2017. Model-agnostic meta-learning for fast adaptation of deep networks. In *ICML*.

Finn, C.; Xu, K.; and Levine, S. 2018. Probabilistic model-agnostic meta-learning. In *NIPS*.

Hamilton, W.; Ying, Z.; and Leskovec, J. 2017. Inductive representation learning on large graphs. In *NIPS*.

He, K.; Zhang, X.; Ren, S.; and Sun, J. 2016. Deep residual learning for image recognition. In *CVPR*.

Hochreiter, S., and Schmidhuber, J. 1997. Long short-term memory. *Neural computation* 9(8):1735–1780.

Kingma, D. P., and Ba, J. 2015. Adam: A method for stochastic optimization. In *ICLR*.

Koch, G.; Zemel, R.; and Salakhutdinov, R. 2015. Siamese neural networks for one-shot image recognition. In *ICML deep learning workshop*.

Lee, Y., and Choi, S. 2018. Gradient-based meta-learning with learned layerwise metric and subspace. In *ICML*.

Li, Z.; Zhou, F.; Chen, F.; and Li, H. 2017. Meta-sgd: Learning to learn quickly for few shot learning. *arXiv preprint arXiv:1707.09835*.

Li, Z.; Li, X.; Wei, Y.; Bing, L.; Zhang, Y.; and Yang, Q. 2019. Transferable end-to-end aspect-based sentiment analysis with selective adversarial learning. In *EMNLP-IJCNLP*.

Mishra, N.; Rohaninejad, M.; Chen, X.; and Abbeel, P. 2018. A simple neural attentive meta-learner. *ICLR*.

Mitchell, T.; Cohen, W.; Hruschka, E.; Talukdar, P.; Yang, B.; Betteridge, J.; Carlson, A.; Dalvi, B.; Gardner, M.; Kisiel, B.; et al. 2018. Never-ending learning. *Communications of the ACM* 61(5):103–115.

Nickel, M.; Tresp, V.; and Kriegel, H.-P. 2011. A three-way model for collective learning on multi-relational data. In *ICML*.

Ravi, S., and Larochelle, H. 2016. Optimization as a model for few-shot learning. *ICLR*.

Schlichtkrull, M.; Kipf, T. N.; Bloem, P.; Van Den Berg, R.; Titov, I.; and Welling, M. 2018. Modeling relational data with graph convolutional networks. In *ESWC*.

Snell, J.; Swersky, K.; and Zemel, R. 2017. Prototypical networks for few-shot learning. In *NIPS*.

Socher, R.; Chen, D.; Manning, C. D.; and Ng, A. 2013. Reasoning with neural tensor networks for knowledge base completion. In *NIPS*.

Suchanek, F. M.; Kasneci, G.; and Weikum, G. 2007. Yago: a core of semantic knowledge. In *WWW*.

Trouillon, T.; Welbl, J.; Riedel, S.; Gaussier, É.; and Bouchard, G. 2016. Complex embeddings for simple link prediction. In *ICML*.

Vinyals, O.; Blundell, C.; Lillicrap, T.; Wierstra, D.; et al. 2016. Matching networks for one shot learning. In *NIPS*.

Vrandečić, D., and Krötzsch, M. 2014. Wikidata: a free collaborative knowledge base. *Communications of the ACM* 57(10):78–85.

Xiong, W.; Yu, M.; Chang, S.; Guo, X.; and Wang, W. Y. 2018. One-shot relational learning for knowledge graphs. In *EMNLP*.

Yang, B.; Yih, W.; He, X.; Gao, J.; and Deng, L. 2015. Embedding entities and relations for learning and inference in knowledge bases. In *ICLR*.

Yang, F. S. Y.; Zhang, L.; Xiang, T.; Torr, P. H.; and Hospedales, T. M. 2018. Learning to compare: Relation network for few-shot learning. In *CVPR*.

Yao, H.; Liu, Y.; Wei, Y.; Tang, X.; and Li, Z. 2019a. Learning from multiple cities: A meta-learning approach for spatial-temporal prediction. In *WWW*.

Yao, H.; Wei, Y.; Huang, J.; and Li, Z. 2019b. Hierarchically structured meta-learning. In *ICML*.

Zhang, C.; Song, D.; Huang, C.; Swami, A.; and Chawla, N. V. 2019. Heterogeneous graph neural network. In *KDD*.