Applied Intelligence (2022) 52:4715–4726
https://doi.org/10.1007/s10489-021-02672-0

RLPath: a knowledge graph link prediction method using reinforcement learning based attentive relation path searching and representation learning

Ling Chen[1] · Jun Cui[1] · Xing Tang[1] · Yuntao Qian[1] · Yansheng Li[2] · Yongjun Zhang[2]

Accepted: 9 July 2021 / Published online: 27 July 2021

Abstract
Due to containing rich patterns between entities, relation paths have been widely used in knowledge graph link prediction. The state-of-the-art link prediction methods considering relation paths obtain relation paths by reinforcement learning with an untrainable reward setting, and realize link prediction by path-ranking algorithm (PRA), which ignores information in entities. In this paper, we propose a new link prediction method RLPath to employ information in both relation paths and entities, which alternately trains a reinforcement learning model with a trainable reward setting to search high-quality relation paths, and a translation-based model to realize link prediction. Simultaneously, we propose a novel reward setting for the reinforcement learning model, which shares the parameters with the attention of the translation-based model, so that these parameters can not only measure the contributions of relation paths, but also guide agents to search relation paths that have high contributions for link prediction, forming mutual promotion. In experiments, we compare RLPath with the state-of-the-art link prediction methods. The results show that RLPath has competitive performance.

Keywords Knowledge graph link prediction · Representation learning · Reinforcement learning · Path searching

1 Introduction

Knowledge graphs (KGs), containing structural knowledge, have been a crucial part of various applications, e.g., information retrieval [1], knowledge extraction [2], fake-review detection [3], and recommendation [4]. Therefore, people are committed to building and developing large-scale KGs, e.g., Freebase [5] and DBpedia [6]. The facts in KGs are represented as triples (head entity, relation, tail entity), which can be abbreviated as (h, r, t). Although KGs have included a large number of triples, they are still far from complete, and link prediction becomes a hot research spot, which aims at completing a triple when h or t is missing. Traditional link prediction methods used to be popular, but with the growth of KG size, they are limited by low computation efficiency and data sparsity. Representation learning based link prediction (RLLP) methods solve these problems, which embed relations and entities into a low-dimensional vector space [7], and encode the semantics of relations and entities into their embeddings.

Most existing RLLP methods only consider single-hop relations, which can be mainly divided into two classes, i.e., translation-based models and semantic matching models. For example, TransE [8], as the most classical translation-based model, regards the single-hop relation as a translation between the head entity and the tail entity. TransE performs well with 1-to-1 relations, but has issues with 1-to-n, n-to-1, and n-to-n relations. To address the issue, a lot of improved models based on TransE are proposed [9–16]. For example, Wang et al. [16]

✉ Ling Chen
lingchen@cs.zju.edu.cn

Jun Cui
cuijun@cs.zju.edu.cn

Xing Tang
tangxing@cs.zju.edu.cn

Yuntao Qian
ytqian@zju.edu.cn

Yansheng Li
yansheng.li@whu.edu.cn

Yongjun Zhang
zhangyj@whu.edu.cn

[1] College of Computer Science and Technology, Zhejiang University, Hangzhou 310027, China

[2] School of Remote Sensing and Information Engineering, Wuhan University, Wuhan 430079, China

proposed TransH, which differentiates relations by projecting entities into different hyperplanes decided by relations; Lin et al. [13] proposed TransR, which supposes relations and entities belonging to different semantic spaces, and projects entities into relation-specific spaces; Sun et al. [12] proposed RotatE, which defines the relation as a rotation from head entity and tail entity in a complex vector space to better capture symmetric and asymmetry relations; Zhang et al. [14] proposed HAKE, which captures semantic hierarchies by mapping entities into the polar coordinate system; Chen et al. [9] proposed UGKE, which introduces the probabilistic soft logic to calculate the confidence score of triples; Tang et al. [15] proposed MKRL, which combines entity descriptions, hierarchical types, and textual relations to learn the representations of entities and relations. RESCAL [11] is a typical semantic matching model, which utilizes a bilinear model to capture the semantic correlations between entities and relations. To reduce the number of parameters, Yang et al. and Liu et al. proposed DistMult [17] and ANALOGY [10], which use diagonal matrices and normal matrices to represent relations, respectively.

In addition to single-hop relations, there are also multi-hop relations, i.e., relation paths, in KGs, which contain rich patterns between entities. Figure 1 shows an example of relation path, represented as (BornIn, StateLocatedIn, CountryLocatedIn), from which we can infer that the "Nationality" of "Barack Obama" is "United States" (with a high probability).

To utilize the rich pattern information in relation paths, some link prediction methods with relation paths have been proposed, which can be divided into four categories according to the way of obtaining relation paths and realizing link prediction.

The first one obtains relation paths by traversal and realizes link prediction by traditional link prediction methods [18–20]. For example, PRA [19] obtains relation paths by random walks and uses relation paths as features to train a per-relation classifier to realize link prediction; Gardner et al. [18] introduced feature similarity based on PRA; Neelakantan et al. proposed PathRNN [20], which uses a recurrent neural network (RNN) to model the path.

The second one obtains relation paths by traversal and realizes link prediction by RLLP methods [21, 22]. For example, based on TransE, Lin et al. [21] proposed PTransE, which obtains relation paths by traversing through the knowledge graph and utilizes both single-hop relations and relation paths to learn the embeddings of relations and entities. By introducing relation paths, PTransE enriches the semantics of relations and entities. However, the number of relation paths increases exponentially with the number of hops. Therefore, PTransE can only utilize short relation paths. Huang et al. [22] improved PTransE by replacing TransE with TransR.

Reinforcement learning, aiming at learning the optimal strategy by maximizing the cumulative reward that the agent obtains from the environment, has proven advance in many KG related tasks [23], due to the availability of better control and more flexibility over the path-searching process. In KG, the goal of reinforcement learning is to make a sequence of decisions on choosing relation paths to finally reach the correct entities, where the policy gradient is utilized for training agents to learn from the interactions with the environment derived from a KG. The third one integrates reinforcement learning with traditional RLLP methods to obtain relation paths and realize link prediction. For example, Xiong et al. [24] proposed DeepPath, which trains an agent to search relation paths between a given entity pair. DeepPath starts from the head entity, and walks through the KG. Its agent chooses a relation at each step to extend the relation path, until reaching the tail entity. Since DeepPath ignores the importance of choosing entities, Li et al. [25] proposed MARLPaR that consists of two agents, one for choosing relations, and the other for choosing entities. However, the reward settings of both DeepPath and MARLPaR are artificial and untrainable: DeepPath gives shorter relation paths higher rewards, ignoring rich information in longer relation paths; MARLPaR gives all relation paths the same reward, therefore it can hardly distinguish high-quality ones from all relation paths. In addition, both DeepPath and MARLPaR use the same way of link prediction as PRA, which fails to take the information in entities into consideration.

The fourth one integrates reinforcement learning with end-to-end learning models to obtain relation paths and realize link prediction, where the reward is trainable and learnt automatically [23, 26–29]. For example, Das et al. [23] proposed MINERVA, which trains an agent to directly realize link prediction. Given a head entity and a relation, the agent of MINERVA walks through the KG starting from the head entity, and stops when it reaches the tail entity. Since MINERVA has the problem of sparse rewards, Shen et al. [29] proposed M-Walk, which consists of a deep recurrent neural network (RNN) and Monte Carlo Tree Search (MCTS). Lin et al. [27] improves MINERVA via reward shaping and action drop-out. By leveraging text corpus with the sentence bag of current entity denoted, Fu et al. proposed CPL [26], which introduces collaborative policy learning for path searching and fact extraction from text. Lv et al. [28]

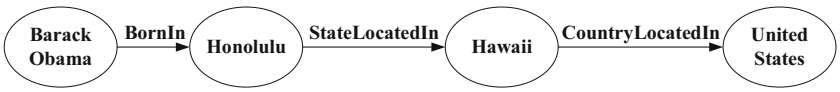


Fig. 1 A relation path between the entity pair (Barack Obama, United States)

proposed Meta-KGR, which adopts meta-learning to learn effective meta parameters from high-frequency relations that could easily adapt to few-shot relations. Despite the promising results of introducing end-to-end learning, there still exist some deficiencies needed to be improved.

Firstly, existing methods [18, 19, 24] only consider relation paths, ignoring the importance of choosing entities in relation path searching. However, the information in entities is important for an agent, ignoring which makes the agent easily "get lost". For example, as shown in the upper portion of Fig. 2, suppose that the agent is searching a relation path between entity pair (Barack Obama, United States), after finding relation path (LanguageSpoken, LanguageSpokenIn), ignoring the information in entity "United States", the agent may choose entity "United Kingdom", which may fail to reach entity "United States" before reaching predefined max step, i.e., the agent "gets lost". Secondly, existing methods [4, 21, 24] calculate the weights of each relation path by the path-constraint resource allocation (PCRA) algorithm [30], which is untrainable. Thirdly, existing methods [21, 24] only realize link prediction on entity pairs whose length of shortest relation path is less than a certain threshold, which ignores rich information in longer relation paths.

To address the aforementioned problems, we propose a link prediction method RLPath combining reinforcement learning based attentive relation path searching with representation learning, which obtains relation paths by reinforcement learning and realizes link prediction by RLLP methods. RLPath alternately trains a reinforcement learning model consisting of two agents, one for choosing relations, and the other for choosing entities, and a translation-based model. Given a triple, the reinforcement learning model searches high-quality relation paths based on the embeddings learned by the translation-based model, and uses a trainable reward setting. The translation-based model utilizes these relation paths to learn the embeddings of relations and entities, and uses attention to measure the contributions of relation paths. The main contributions of this paper are summarized as follows:

- We propose RLPath, which alternately trains a reinforcement learning model to search high-quality relation paths, and a translation-based model to realize link prediction.

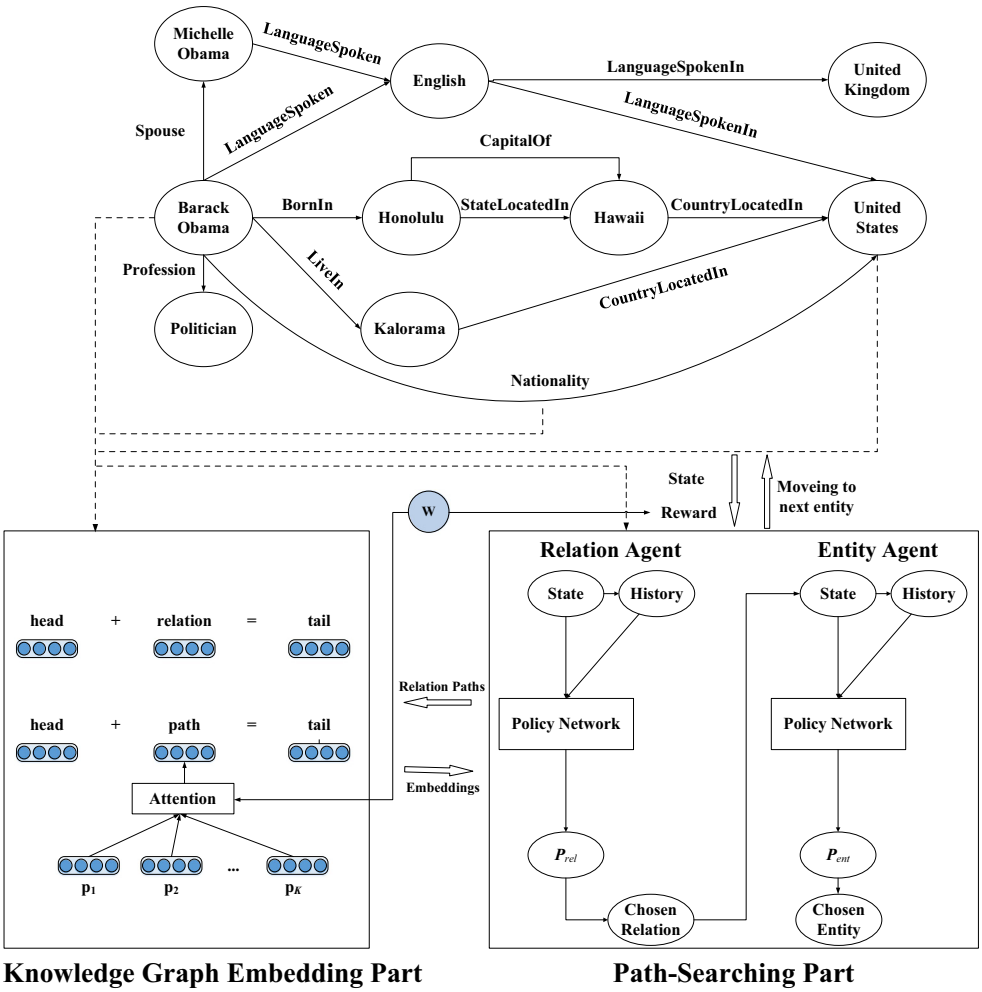


Fig. 2 The overall architecture of RLPath, which includes the path-searching part (at the bottom right of Fig. 2) and the representation learning part (at the bottom left of Fig. 2). The path-searching part searches relation paths between entity pairs based on the embeddings learned by the representation learning part. The path-searching part trains two agents, i.e., the relation agent and the entity agent. The representation learning part utilizes both single-hop relations and relation paths obtained by the path-searching part to learn embeddings, and uses the shared attention to measure the contributions of relation paths

- We propose a novel reward setting for the reinforcement learning model, which shares the parameters with the attention of the translation-based model, so that these parameters can not only measure the contributions of relation paths, but also guide agents to search relation paths that have high contributions for link prediction, forming mutual promotion.
- We evaluate RLPath on FB15K-237 and NELL-995 datasets, and compare it with the state-of-the-art link prediction methods. Experimental results show that RLPath achieves a competitive performance.

2 Methodology

In this section, we present the technical details of our method, including two parts: the representation learning part, which trains a translation-based model to learn the embeddings of relations and entities, and the path-searching part, which trains a reinforcement learning model consisting of two agents to search high-quality relation paths.

2.1 Problem definition

We use $\mathcal{E}$ and $\mathcal{Q}$ to represent the entities and relations in a KG, respectively. The facts in a KG are represented as triples (h, r, t), where $h, t \in \mathcal{E}$, $r \in \mathcal{Q}$. Then, a knowledge graph can be represented as $\mathrm{KG} = \{(h, r, t)\}$. Our task is link prediction, i.e., completing a triple when h or t is missing.

2.2 Overall architecture

Figure 2 shows the overall architecture of RLPath. Given a triple (h, r, t), the path-searching part searches relation paths between (h, t) based on the embeddings learned by the representation learning part. The path-searching part trains two agents: the relation agent and the entity agent. From head entity h, at step τ, the relation agent chooses a relation r_τ among all possible relations of entity e_τ, and the entity agent chooses an entity $e_{\tau+1}$ in the condition of entity e_τ and chosen relation r_τ. The process stops until reaching target entity t or reaching predefined max step $\tau_{\max}$. Then, relation paths that reach target entity t are sent to the representation learning part. The representation learning part utilizes both single-hop relations and relation paths to learn the embeddings of relations and entities, and uses attention to measure the contributions of relation paths. The attention of relation paths in the representation learning part and the reward of the path-searching part share the weight matrix $\mathbf{W}$, forming mutual promotion.

2.3 Representation learning part

The representation learning part is based on TransE [8]. Given a triple (h, r, t), TransE regards r as a translation between h and t, i.e., $\mathbf{h} + \mathbf{r} \approx \mathbf{t}$, where $\mathbf{h},\mathbf{r},\mathbf{t} \in \mathbb{R}^d$ are the embeddings of h, r, and t, respectively. The energy function of TransE is defined as follows:

$$E_{\mathrm{TransE}}(h, r, t) = |\mathbf{h}+\mathbf{r}-\mathbf{t}|. \tag{1}$$

RLPath utilizes both single-hop relations and relation paths. A single-hop relation is a direct relation between an entity pair, i.e., r is a single-hop relation of (h, t). A relation path is defined as follows: given n triples (h, r_1, e_1), (e_1, r_2, e_2), …, (e_{n-1}, r_n, t), in which the tail entity of the former one is the same as the head entity of the latter one, then, $p = (r_1, r_2, \ldots, r_n)$ is a relation path between (h, t).

The semantic relevance of each relation path to the corresponding single-hop relation is different, i.e., the contributions of relation paths are different. Therefore, we use attention [31] to measure the contributions of relation paths. Suppose $P = \{p_1, \ldots, p_K\}$ is the relation path set of (h, r, t), the semantics of relation paths in P should be similar to the corresponding single-hop relation r, i.e., $\mathbf{p}_i \approx \mathbf{r}$, where $\mathbf{p}_i$ is the embedding of $p_i = (r_{i1}, r_{i2}, \ldots, r_{in})$, which is defined as follows:

$$\mathbf{p}_i = \sum_{j=1}^{n} \mathbf{r}_{ij}, \tag{2}$$

where $\mathbf{r}_{ij}$ is the embedding of r_{ij}.

Therefore, the energy function of the representation learning part is defined as follows:

$$E(h, r, t) = |\mathbf{h} + \mathbf{r}-\mathbf{t}| + \sum_{p_i \in P} \alpha_i |\mathbf{p}_i-\mathbf{r}|, \tag{3}$$

where α_i is the attention of relation path p_i, which is defined as follows:

$$\eta_i = \tanh(\mathbf{W}\mathbf{p}_i), \tag{4}$$

$$\alpha_i = \frac{\exp(\mathbf{r} \cdot \eta_i)}{\sum_{j=1}^{K} \exp\left(\mathbf{r} \cdot \eta_j\right)}, \tag{5}$$

where $\mathbf{W} \in \mathbb{R}^{d \times d}$ is the weight matrix.

2.4 Path-searching part

In the path-searching part, we train a reinforcement learning model having two agents to search high-quality relation paths, one for choosing relations (called relation agent), and the other for choosing entities (called entity agent). In this way, RLPath takes both relation choosing and entity choosing into consideration in relation path searching.The reinforcement learning model consists of external environment and policy networks.

2.4.1 External environment

External environment is a Markov Decision Process (MDP), which represents the interactions between the KG and the agents. The MDP consists of four parts: state $\mathcal{S}$, action $\mathcal{A}$, transition $\mathcal{T}$, and reward $\mathcal{R}$, which can be represented as a tuple $(\mathcal{S}, \mathcal{A}, \mathcal{T}, \mathcal{R})$. In the following, we suppose the agents are searching relation paths given a triple (h, r, t).

Action At step τ, the relation agent chooses a relation r_τ from its action space $\mathcal{A}_{rel,\tau} = \{r|(e_\tau, r, e) \in \mathrm{KG}\}$, and the entity agent chooses an entity $e_{\tau+1}$ from its action space $\mathcal{A}_{ent,\tau} = \{e|(e_\tau, r_\tau, e) \in \mathrm{KG}\}$. We call relations in $\mathcal{A}_{rel,\tau}$ as candidate relations, entities in $\mathcal{A}_{ent,\tau}$ as candidate entities.

State The state of the agents should record the current position, the single-hop relation, and the target entity of relation path searching, so that the agents can choose an action according to the semantics of the single-hop relation and the target entity. The state vectors of the relation agent and the entity agent at step τ are defined as $\mathbf{s}_{rel,\tau} = (\mathbf{e}_\tau, \mathbf{r}, \mathbf{t}) \in \mathbb{R}^{3d}$ and $\mathbf{s}_{ent,\tau} = (\mathbf{e}_\tau, \mathbf{r}, \mathbf{t}, \mathbf{r}_\tau) \in \mathbb{R}^{4d}$, respectively, where $\mathbf{e}_\tau$ is the embedding of entity e_τ and $\mathbf{r}_\tau$ is the embedding of chosen relation r_τ.

Transition From step τ to step $\tau+1$, the states of the two agents are updated from $(\mathbf{e}_\tau, \mathbf{r}, \mathbf{t})$ and $(\mathbf{e}_\tau, \mathbf{r}, \mathbf{t}, \mathbf{r}_\tau)$ to $(\mathbf{e}_{\tau+1}, \mathbf{r}, \mathbf{t})$ and $(\mathbf{e}_{\tau+1}, \mathbf{r}, \mathbf{t}, \mathbf{r}_{\tau+1})$, respectively. Therefore, transition $\mathcal{T}$ can be represented as $\mathcal{T}_{rel}((\mathbf{e}_\tau, \mathbf{r}, \mathbf{t}), r_\tau) = (\mathbf{e}_{\tau+1}, \mathbf{r}, \mathbf{t})$ and $\mathcal{T}_{ent}((\mathbf{e}_\tau, \mathbf{r}, \mathbf{t}, \mathbf{r}_\tau), e_{\tau+1}) = (\mathbf{e}_{\tau+1}, \mathbf{r}, \mathbf{t}, \mathbf{r}_{\tau+1})$.

Reward The path-searching part aims to search high-quality relation paths. Therefore, the reward should not only measure whether a relation path reaches the target entity, but also measure the quality of the relation path. Our reward setting consists of the following two components:

Global accuracy Global accuracy measures whether a relation path reaches the target entity, which is defined as follows:

$$\mathbf{R}_g = \begin{cases} 1, \text{if the relation path reaches } t \\ 0, \text{otherwise} \end{cases}. \tag{6}$$

Path weight Path weight measures the quality of a relation path, i.e., the semantic relevance between the relation path and the corresponding single-hop relation. In the representation learning part, we use attention to measure the contributions of relation paths. In the path-searching part, we use the same parameters to measure the qualities of relation paths. The path weight is defined as follows:

$$\mathbf{R}_w = \begin{cases} \mathbf{rWp}, \text{if the relation path reaches } t \\ 0, \quad \text{otherwise} \end{cases}, \tag{7}$$

where $\mathbf{W}$ is the weight matrix, which is shared with the representation learning part.

Finally, the total reward is defined as follows:

$$\mathbf{R}_{total} = \mathbf{R}_g + \mathbf{R}_w. \tag{8}$$

2.4.2 Policy networks

Each of the two agents can be represented as a policy network with parameters θ: $\pi_\theta(a|\mathbf{X}) = P(a|\mathbf{X}; \theta)$, which maps input $\mathbf{X}$ to the probability distribution of candidate relations or entities. We build history-dependent policy networks whose inputs consist of history and state. History records the relations and entities that have been chosen in a process of relation path searching. It is the same for the two agents. The embedding of history at step τ is represented as $\mathbf{d}_\tau \in \mathbb{R}^{2d}$. We use one layer RNN to encode chosen relations and entities into $\mathbf{d}_\tau$:

$$\mathbf{d}_\tau = \mathrm{RNN}(\mathbf{d}_{\tau-1}, [\mathbf{e}_{\tau-1}, \mathbf{r}_{\tau-1}]), \tag{9}$$

where $\mathbf{e}_{\tau-1}$ and $\mathbf{r}_{\tau-1}$ are the embeddings of the chosen entity and relation at step $\tau-1$, respectively. [,] is the concatenation of two embeddings. The inputs of the two policy networks at step τ are represented as $\mathbf{X}_{rel,\tau} = [\mathbf{d}_\tau, \mathbf{s}_{rel,\tau}]$ and $\mathbf{X}_{ent,\tau} = [\mathbf{d}_\tau, \mathbf{s}_{ent,\tau}]$, respectively.

The structure of the two policy networks is a fully-connected neural network with two hidden layers, each followed by a ReLU nonlinearity layer. Since the embeddings of relations and entities contain features, we include this information in the following way:

$$\pi_\theta\left(\mathcal{A}_{rel,\tau}\middle|\mathbf{X}_{rel,\tau}\right) = \mathrm{softmax}\left(\mathbf{A}_{rel,\tau}\mathbf{O}_{rel,\tau}\right), \tag{10}$$

$$\pi_\theta\left(\mathcal{A}_{ent,\tau}\middle|\mathbf{X}_{ent,\tau}\right) = \mathrm{softmax}\left(\mathbf{A}_{ent,\tau}\mathbf{O}_{ent,\tau}\right), \tag{11}$$

where $\pi_\theta\left(\mathcal{A}_{rel,\tau}\middle|\mathbf{X}_{rel,\tau}\right)$ and $\pi_\theta\left(\mathcal{A}_{ent,\tau}\middle|\mathbf{X}_{ent,\tau}\right)$ are the probability distributions of all candidate relations and entities at step τ, respectively; $\mathbf{O}_{rel,\tau} \in \mathbb{R}^d$ and $\mathbf{O}_{ent,\tau} \in \mathbb{R}^d$ are the outputs of the second ReLU nonlinearity layer of the relation agent and the entity agent, respectively. $\mathbf{A}_{rel,\tau} \in \mathbb{R}^{|\mathcal{A}_{rel,\tau}| \times d}$ and $\mathbf{A}_{ent,\tau} \in \mathbb{R}^{|\mathcal{A}_{ent,\tau}| \times d}$ are matrixes consisting of the embeddings of all candidate relations and entities at step τ, respectively.

2.5 Training

In this section, we describe the parameter updating in our method RLPath, including the representation learning part and the path-searching part, and present the complete training procedure of RLPath.

2.5.1 Parameter update

In the representation learning part, we update parameters by minimizing a margin-based loss function [8], which is defined

as follows:

$$L_{\mathrm{KG}} = \sum_{(h,r,t)\in T} \sum_{\left(h,r',t\right)\in T^{-}} \left[\gamma + E(h,r,t) - E\left(h,r',t\right)\right]^{+}, \quad (12)$$

where $[x]^{+} = max\,(0,x)$ represents the maximum between 0 and x; γ is the margin; T is the set of all the triples in the KG, i.e., positive triples; T^{-} is the set of negative triples, which is defined as follows:

$$T^{-} = \left\{\left(h,r',t\right) \notin \mathrm{KG} \middle| \left(h,r,t\right) \in \mathrm{KG}, r' \in \mathcal{Q}\right\}. \quad (13)$$

In the path-searching part, we update parameters by maximizing expected total reward for all searched relation paths, which is defined as follows:

$$J(\theta) = \sum_{M} \mathbb{E}_{a\sim\pi_\theta}\left(\sum_{\tau} \mathbf{R}_{s_\tau,a_\tau}\right), \quad (14)$$

where $J(\theta)$ is the expected total reward for all searched relation paths; M is the set of all searched relation paths; $\mathbf{R}_{s_\tau,a_\tau}$ is the reward in condition of state s_τ and action a_τ.

We use Monte-Carlo Policy Gradient [32] to calculate the gradient of $J(\theta)$:

$$\nabla_\theta J(\theta) = \nabla_\theta \sum_{M} \mathbb{E}_{a\sim\pi_\theta}\left(\sum_{\tau} \mathbf{R}_{s_\tau,a_\tau}\right) \approx \nabla_\theta \sum_{M}\sum_{\tau} \log\pi_\theta(a = a_\tau|\mathbf{X}_\tau)\mathbf{R}_{s_\tau,a_\tau}. \quad (15)$$

For the relation agent, $a_\tau = r_\tau$ and $\pi_\theta(a = a_\tau|\,\mathbf{X}_\tau) = \pi_\theta(r_\tau|\,\mathbf{X}_{rel,\,\tau})$. For the entity agent, $a_\tau = e_{\tau+1}$ and $\pi_\theta(a = a_\tau|\,\mathbf{X}_\tau) = \pi_\theta(e_{\tau+1}|\,\mathbf{X}_{ent,\,\tau})$.

2.5.2 Training procedure

Algorithm 1 shows the complete training procedure of RLPath. First, we use TransE to pre-train the embeddings of relations and entities (Line 1). In some KGs, the number of relations is large, so training the two agents without supervision would be time-consuming and parameters are difficult to converge. Therefore, we exploit supervised learning (Line 2–5), where the reward of each step $\boldsymbol{R}_{s_\tau,a_\tau}$ is set to 1. Then, we implement reinforcement learning to further update parameters (Line 7–24), where the reward of each step $\boldsymbol{R}_{s_\tau,a_\tau}$ is set to $\boldsymbol{R}_{total}$. After that, we use the relation paths that reach the tail entity to retrain the embeddings (Line 25). Reinforcement learning and embedding retraining are alternated for a predefined number of iterations (Line 26–27). We use Adam [33] with a learning rate of 0.001 to update embeddings, and the updating process is defined as follows:

$$g_t = \nabla_\theta J(\theta_{t-1}), \quad (16)$$

$$m_t = \beta_1 m_{t-1} + (1-\beta_1) g_t \quad (17)$$

$$v_t = \beta_2 v_{t-1} + (1-\beta_2) g_t^2, \quad (18)$$

$$\widehat{m}_t = \frac{m_t}{1-\beta_1^t}, \quad (19)$$

$$\widehat{v}_t = \frac{v_t}{1-\beta_2^t}, \quad (20)$$

$$\theta_t = \theta_{t-1} - \frac{\eta}{\sqrt{\widehat{v}_t} + \epsilon}\widehat{m}_t, \quad (21)$$

where $\beta_1 = 0.9$, $\beta_2 = 0.999$, and $\epsilon = 10^{-8}$. Eq. 16 gets the gradient of $J(\theta)$ at step t. Equations 17 and 18 update the biased first moment estimate and the biased second raw moment estimate, respectively. Equations 19 and 20 compute the bias-corrected first moment estimate and the bias-corrected second raw moment estimate, respectively. Equation 21 updates parameters.

3 Experiments

We evaluate the performance of RLPath on two datasets: FB15K-237 [34] and NELL-995 [24]. We choose TransE [8], PTransE [21] (we only consider the add operation, which has the best performance in [21]), DeepPath [24], MARLPaR [25], MINERVA [23], M-Walk [29], and other state-of-the-art link prediction methods, i.e., DistMult [17] (which is a simple variant of bilinear method), ComplEx [35] (which is an extension of DistMult that uses complex-valued vectors instead of real-valued vectors), and ConvE [36] (which uses 2D convolutions to realize link prediction) as our baselines. We compare RLPath with baselines on the link prediction task. We implement our model with TensorFlow and implement training, validation, and test tasks on a server with one GPU (NVIDIA RTX 2080Ti).

3.1 Datasets

To evaluate the performance of RLPath, following [23], we conduct experiments on two more common and realistic KG datasets, i.e., FB15K-237 [34] and NELL-995 [24]. FB15K-237 is constructed from the original FB15K [8] by removing highly redundant relations, which is a subset of Freebase [5]. NELL-995 is constructed from the 995th iteration of the NELL system by removing two relations (generalizations and haswikipediaurl), and selecting triples with

Table 1 The statistics of datasets

Dataset	#Rel	#Ent	#Train	#Valid	#Test
FB15K-237	237	14,541	272,115	17,535	20,466
NELL-995	200	75,492	154,213	-	3992

top-200 relations. Table 1 shows the statistics of the datasets. Similar to DeepPath, MARLPaR, MINERVA, and M-Walk, we add inverse relations to training sets (for each triple (h, r, t), we add a triple (t, r^{-1}, h), where r^{-1} is the inverse relation of r).

3.2 Parameter settings

We investigate the influence of several important parameters on the model performance, including the dimension of embeddings and the margin γ in the representation learning part.

To explore the dimension of embeddings in the representation learning part, we select it among {50, 100, 200}, The experimental results on dataset FB15K-237 are shown in Fig. 3, from which we can find that with the increasing of the dimension of embeddings in the representation learning part, first the model performance increases and then decreases slightly. When the number is 100, the model achieves the highest Hits@1, Hits@3, Hits@10, and MRR. Therefore, we set the dimension of embeddings in the representation learning part as 100 in the following experiments.

For the margin γ in the representation learning part, we select it among {1, 2, 3}. The experimental results on dataset FB15K-237 are shown in Fig. 4, from which we can find that with the increasing of the margin γ in the representation learning part, the model performance decreases slightly. When the number is 1, the model achieves the highest Hits@1, Hits@3, Hits@10, and MRR. Therefore, we set the margin γ in the representation learning part as 1 in the following experiments.

On both FB15K-237 and NELL-995, the learning rates for the parameters are 0.001. For the dimension of embeddings in the representation learning part, the optimal setting is 100. For the margin γ in the representation learning part, the optimal setting is 1. We set the sizes of the two hidden layers of the policy networks =1024 and 100 in the path-searching part. On FB15K-237, we choose max step $\tau_{\max} = 10$ and the max length of the relation paths in supervised learning $l_{max} = 10$.

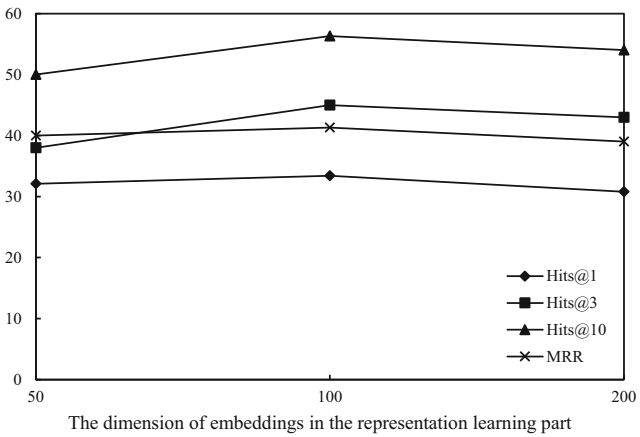


Fig. 3 The impact of the dimension of embeddings in the representation learning part on the performance of RLPath

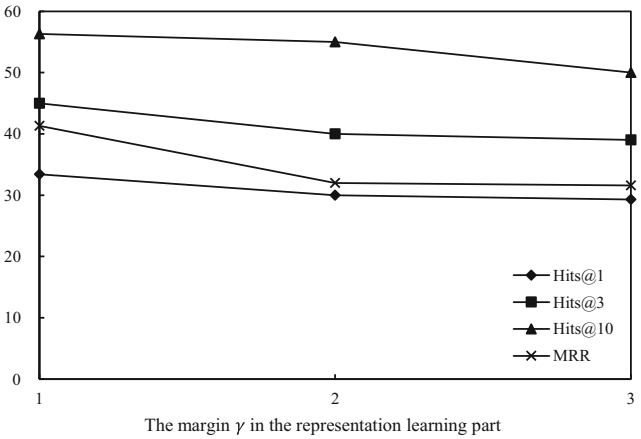


Fig. 4 The impact of the margin γ in the representation learning part on the performance of RLPath

On NELL-995, we choose $\tau_{\max} = 20$ and $l_{max} = 40$. We use Adam optimizer [33] to update parameters.

3.3 Computational complexity discussion

In KGs, the number of relations is quite large. Considering the complexity of the triple set, training the two agents without supervision would be time-consuming and parameters are difficult to converge. Therefore, we exploit supervised learning with a randomized breadth-first search (BFS). In spite of this, compared with other methods, our method still has higher computational complexity due to the need of searching relation paths for all triples.

3.4 Evaluation protocol

Given a test triple as ground truth, we use a common evaluation method of translation-based models [8] to evaluate the performance of RLPath on the link prediction task: first, replace the head entity (or tail entity) of the test triple with all entities in $\mathcal{E}$, forming candidate triples; and then, rank candidate triples with energy function (Eq. 3) in ascending order; at last, record the ranking of the ground truth. Through the above process, a head entity prediction (or tail entity prediction) is accomplished.

On FB15K-237, we compare RLPath with TransE, PTransE, MINERVA, DistMult, ComplEx, and ConvE by link prediction on Hits@1, Hits@3, Hits@10, MRR (Mean Reciprocal Rank), and MR (Mean Rank) metrics. MRR denotes the average of the reciprocal ranks of correct entities. MR denotes the ranks of correct entities. Hits@k denotes the percentage of correct entities ranked at top k. Higher MRR and Hits@k and lower MR indicate better performance. Since PRA trains a model for each relation, DeepPath and MARLPaR cannot be evaluated on FB15K-237, which contains a few relations. To reduce the amount of calculation, we use a re-rank method [21]: first, rank all candidate triples with the energy function of TransE (Eq. 1); and then, rank the

top-200 candidate triples with Eq. 3. On NELL-995, the number of entities is large, while the triples are sparse, and training the methods on the whole dataset results in poor results. Therefore, we choose the same 10 relations as [23]. For each relation, we use triples that contain it in the training set to train the methods, and evaluate the methods by tail entity prediction. We compare RLPath with DeepPath, MARLPaR, MINERVA, and M-Walk on MAP metric.

To justify the effectiveness of the components of RLPath, we also evaluate three variants models. RLPath (CON) replaces the reward setting of RLPath, and only considers global reward, which is the same as MARLPaR; RLPath (DP) replaces the reward setting of RLPath, and considers global reward, as well as the length and diversity of relation paths, which is the same as DeepPath; RLPath(PRA) replaces the way of realizing link prediction of RLPath, and uses the same way of link prediction as PRA.

3.5 Result

Table 2 shows the performances of link prediction on FB15K-237, where "Head" and "Tail" are the results of head entity prediction and tail entity prediction, respectively, from which we could find out (Since MINERVA can only realize tail entity prediction, we do not give its results on "Head"):

(1) Although PTransE utilizes relation paths, it has a worse performance on most metrics than the state-of-the-art link prediction methods DistMult, ComplEx, and ConvE, which do not consider relation paths. It is because the number of relation paths increases exponentially with the number of hops, using traversal, PTransE can only utilize short relation paths. In addition, PTransE uses PCRA to calculate the weights of each relation path, which is untrainable.

(2) RLPath has a better performance on most metrics than the state-of-the-art methods DistMult, ComplEx, and ConvE, which might be because RLPath can effectively search high-quality and longer relation paths, measure the contributions of relation paths, and utilize the information in relation path.

(3) RLPath has a better performance than MARLPaR, which might be because MARLPaR uses PRA, which only considers relation paths, ignoring the information in entities, to realize link prediction, and gives all relation paths the same reward, which makes it difficult for MARLPaR to distinguish high-quality ones from all relation paths; while RLPath realizes link prediction by the translation-based model, considering the information in both relation paths and entities, and uses a trainable reward setting.

(4) MINERVA has a poor performance, partly because of the problem of sparse reward as mentioned in [29], and partly because MINERVA can only realize link prediction on entity pairs whose length of shortest relation path is less than a certain threshold.

(5) RLPath has a better performance than RLPath(CON) and RLPath(DP), which might be because the reward settings of MARLPaR and DeepPath can hardly distinguish high-quality ones from all relation paths. The reward setting of RLPath is trainable and shared with the attention of the translation-based model, therefore, RLPath can search high-quality relation paths, justifying the effectiveness of the reward setting of RLPath.

Table 3 shows the MAP of link prediction on NELL-995, from which we could find out:

(1) RLPath has a better performance on all metrics than DeepPath, which might be because RLPath realizes link

Table 2 The performances of link prediction on FB15K-237, in which "Head" means predicting head entity and "Tail" means predicting tail entity

Metric(%)	Head					Tail				
	MR	MRR	Hits@1	Hits@3	Hits@10	MR	MRR	Hits@1	Hits@3	Hits@10
TransE	969	16.4	9.1	17.4	31.1	855	33.4	23.3	37.5	53.2
PTransE	486	20.8	12.9	22.5	37.1	453	29.5	19.1	33.2	50.8
MINERVA	–	–	–	–	–	447	29.8	21.8	33.7	46.2
DistMult	503	19.2	11.9	20.9	34.1	404	38.7	29.7	42.6	56.7
ComplEx	526	18.7	11.4	20.3	33.7	412	37.7	28.5	41.4	56.2
ConvE	458	20.8	12.9	22.7	37.6	397	41.0	31.7	44.6	59.8
RLPath(CON)	321	22.6	15.9	26.9	38.5	293	39.5	32.1	43.8	54.9
RLPath(DP)	297	23.8	16.3	27.3	37.9	288	38.8	31.4	43.9	55.4
RLPath	264	24.9		27.4	40.1	242	41.3	33.4	45.0	56.3

Table 3 The MAP of entity prediction on NELL-995

Task	DeepPath	MARLPaR	MINERVA	M-Walk	RLPath (CON)	RLPath (DP)	RLPath (PRA)	RLPath
athletePlaysInLeague	95.1	96.2	95.9	97.6	96.4	93.2	96.9	98.6
worksFor	70.3	72.9	82.7	84.6	84.4	83.4	76.3	85.3
orgHiredPerson	74.4	77.7	86.8	89.2	77.7	78.1	77.1	82.1
athletePlaysSport	93.1	96.5	98.2	98.2	96.7	97.5	96.8	98.4
teamPlaysSport	73.5	83.8	87.5	88.6	76.8	79.0	85.8	80.5
PersonBornInLoc	75.2	80.3	78.9	81.2	83.7	85.7	81.4	82.3
PersonLeadsOrg	80.8	82.5	85.4	89.1	85.4	84.9	83.6	
athleteHomeStadium	84.6	87.9	92.0	91.6	86.9	88.7	89.2	92.1
OrgHeadquartereInCity	79.4	79.5	94.1	95.2	82.6	80.7	81.0	81.9
athletePlaysForTeam	73.3	76.2	82.5	83.9	78.2	76.8	77.9	80.5

prediction by the translation-based model, and uses a trainable reward setting.

(2) M-Walk shows a significantly better performance than DeepPath, MARLPaR, and MINERVA, as it solves the problem of sparse rewards by combing MCTS and deep RNN. RLPath has a comparable performance to M-Walk, indicating that RLPath can search high-quality relation paths, and encode this information into the embeddings of relations and entities.

(3) RLPath has a better performance in most tasks than RLPath(RPA). RLPath and RLPath(RPA) have the same reinforcement learning framework. However, RLPath (RPA) uses PRA to realize link prediction, which only utilizes relation paths, ignoring the information in entities, while RLPath uses a translation-based model, considering the information in both relation paths and entities, justifying the effectiveness of realizing link prediction by RLLP methods.

(4) RLPath(DP) does not have significantly better performance than RLPath(CON), which might be because RLPath(DP) gives short relation paths higher rewards, ignoring the information contained in long relation paths. RLPath has a better performance than RLPath(CON) and RLPath(DP), which might be because we use a trainable reward setting that shares the parameters with the attention of the translation-based model, therefore, RLPath can distinguish high-quality ones from all relation paths, justifying the effectiveness of the reward setting of RLPath.

3.6 Case study

In Table 4, we show top-3 relation paths in confidence searched by RLPath for some single-hop relations on NELL-995 in the last iteration. "Co-occur" represents the number of times a relation path co-occurs with its corresponding single-hop relation; "HC" and "Conf" represent the ratios of co-occurrences to the occurrences of the relation path and

Table 4 The top-3 relation paths in confidence searched by RLPath for some single-hop relations on NELL-995 in the last iteration

Single-hop relation	Relation path	Co-occur	Conf (%)	HC (%)
personBornIn Loc	(personBornInCity, subpartOf^{-1}, locationLocatedWithinLoc)	351	64.0	28.6
	(personGraduatedSchool, synonymFor, personBornInCity, subpartOf^{-1}, locationLocatedWithinLocation)	159	24.8	23.7
	(personGraduatedUniversity, synonymFor, personBornInCity, subpartOf^{-1}, locationLocatedWithinLocation)	159	24.8	23.9
athletePlays Sport	(athletePlaysForTeam, teamPlaysSport)	453	22.6	38.2
	(athletePlaysInLeague, personBelongsToOrganization^{-1}, athletePlaysSport)	250	16.1	51.7
	(athleteLedSportsTeam, teamPlaysSport)	192	9.6	50.9
athleteHome Stadium	(athletePlaysForTeam, teamHomeStadium)	461	69.1	54.9
	(athletePlaysForTeam, athleteLedSportsTeam^{-1}, athleteHomeStadium)	365	54.7	40.2
	(athleteLedSportsTeam, teamHomeStadium)	98	14.7	32.1

the single-hop relation, respectively. Some interesting facts can be obtained. For example, from the relation path (athletePlaysForTeam, teamHomeStadium), if athlete A plays for team B, and C is the home stadium of team B, then, C is the home stadium of athlete A with a high probability. We can observe that RLPath can search relation paths that not only have high confidence but also are in line with common sense in reality.

4 Conclusion and future work

In this paper, we propose a link prediction method RLPath to employ information in both relation paths and entities, which alternately trains a reinforcement learning model with a trainable reward setting to search high-quality relation paths, and a translation-based model to realize link prediction. Simultaneously, we propose a novel reward setting for the reinforcement learning model, which shares the parameters with the attention of the translation-based model, so that these parameters can not only measure the contributions of relation paths, but also guide agents to search relation paths that have high contributions for link prediction, forming mutual promotion. In addition, our method RLPath can be applied to plenty of fields, e.g., information retrieval and recommendation, by utilizing the embeddings of entities and relations learnt by RLPath.

In the experiments, RLPath shows competitive performance, justifying that RLPath can effectively search and utilize high-quality relation paths by the trainable reward setting of RLPath and the shared attention of the translation-based model, which justifies the effectiveness of the reward setting of RLPath. In the future, we will continue our study in the following aspects: (1) we will try to improve the representation learning part based on the state-of-the-art link prediction methods to train RLPath, as they have better performances than TransE. (2) We will attempt to adaptively adjust training times and weights based on the characteristics of training samples, as RLPath is vulnerable to the number of training samples.

Acknowledgments This work was funded by the National Key Research and Development Program of China (No. 2018YFB0505000) and the Fundamental Research Funds for the Central Universities (No.2020QNA5017).

References

1. Han B, Chen L, Tian X (2018) Knowledge based collection selection for distributed information retrieval. Inf Process Manag 54(1): 116–128
2. Romadhony A, Widyantoro D, Purwarianti A (2019) Utilizing structured knowledge bases in open IE based event template extraction. Appl Intell 49:206–219
3. Fang Y, Wang H, Zhao L, Yu F, Wang C (2020) Dynamic knowledge graph based fake-review detection. Appl Intell 50(12):4281–4295
4. Lin L, Liu J, Lv Y, Guo F (2020) A similarity model based on reinforcement local maximum connected same destination structure oriented to disordered fusion of knowledge graphs. Appl Intell 50: 2867–2886
5. Bollacker, K., Evans, C., Paritosh, P., Sturge, T., and Taylor, J. (2008). Freebase: a collaboratively created graph database for structuring human knowledge. In *proceedings of the 14th ACM SIGMOD international conference on Management of Data*, 1247–1250
6. Bizer C, Lehmann J, Kobilarov G, Auer S (2009) Dbpedia-a crystallization point for the web of data. J Web Semantics 7(3):154–165
7. Dong, X., Gabrilovich, E., Heitz, G., Horn, W., and Lao, N. (2014). Knowledge vault: a web-scale approach to probabilistic knowledge fusion. In *proceedings of the 20th ACM SIGKDD international conference on knowledge discovery and data mining*, 601–610
8. Bordes, A., Usunier, N., García-Durán, A., and Yakhnenko, O. (2013). Translating Embeddings for Modeling Multi-Relational Data. In *Proceedings of the 27th Annual Conference on Neural Information Processing Systems*, 2787–2795
9. Chen, X., Chen, M., Shi, W., Sun, Y., and Zaniolo, C. (2019). Embedding uncertain knowledge graphs. In *Proceedings of the 33rd AAAI Conference on Artificial Intelligence*, 33(01), 3363–3370
10. Liu, H., Wu, Y., and Yang, Y. (2017). Analogical Inference for Multi-Relational Embeddings," In *Proceedings of the 34th International Conference on Machine Learning*, 70, 2168–2178
11. Nickel, M., Tresp, V., and Kriegel, H. (2011). A three-way model for collective learning on multi-relational data. In *proceedings of the 28th international conference on international conference on machine learning*, 809–816
12. Sun, Z., Deng, Z., Nie, J., and Tang, J. (2019). RotatE: knowledge graph embedding by relational rotation in complex space. In *Proceedings of the 7th International Conference on Learning Representations*
13. Lin, Y., Liu, Z., Sun, M., Liu, Y., and Zhu, X. (2015). Learning entity and relation Embeddings for knowledge graph completion. In *proceedings of the 29th AAAI conference on artificial intelligence*, 2181-2187
14. Zhang, Z., Cai, J., Zhang, Y., and Wang, J. (2020). Learning hierarchy-aware knowledge graph Embeddings for link prediction. In *proceedings of the 34th AAAI conference on artificial intelligence*, 3065-3072
15. Tang X, Chen L, Cui J, Wei B (2019) Knowledge representation learning with entity descriptions, hierarchical types, and textual relations. Inf Process Manag 56(3):809–822
16. Wang, Z., Zhang, J., Feng, J., and Chen, Z. (2014). Knowledge graph embedding by translating on Hyperplanes. In *proceedings of the 28th AAAI conference on artificial intelligence*, 1112-1119
17. Yang, B., Yih, W., He, X., Gao, J., and Deng, L. (2015). Embedding entities and relations for learning and inference in knowledge bases. In *Proceedings of the 3rd International Conference on Learning Representation*
18. Gardner, M., Talukdar, P., Krishnamurthy, J., and Mitchell, T. (2014). Incorporating vector space similarity in random walk inference over knowledge bases. In *proceedings of the* 18th conference on empirical methods in natural language processing, 397–406
19. Lao N, Cohen W (2010) Relational retrieval using a combination of path-constrained random walks. Mach Learn 81(1):53–67

20. Neelakantan, A., Roth, B., McCallum, A. (2015). Compositional vector space models for Knowledge Base completion. Computer Ence, 1–16
21. Lin, Y., Liu, Z., Luan, H., and Sun, M. (2015). Modeling relation paths for representation learning of knowledge bases. In *proceedings of the* 19th conference on empirical methods in natural language processing, 705–714
22. Huang, W., Li, G., and Jin, Z. (2017). Improved Knowledge Base completion by the path-augmented TransR model. Knowledge Science, Engineering and Management, 149–159
23. Das, R., Dhuliawala, S., Zaheer, M., Vilnis, L., Durugkar, I., Krishnamurthy, A., Smola, A., and McCallum, A. (2018). Go for a walk and arrive at the answer: reasoning over paths in knowledge bases using reinforcement learning. In *Proceedings of the 6th International Conference on Learning Representations*
24. Xiong, W., Hoang, T., and Wang, W. (2017). Deeppath: a reinforcement learning method for knowledge graph reasoning. In *proceedings of the* 21th conference on empiraical methods in natural language processing, 564–573
25. Li, Z., Jin, X., Guan, S., Wang, Y., and Cheng, X. (2018). Path reasoning over knowledge graph: a multi-agent and reinforcement learning based method. In *proceedings of the 18th IEEE international conference on data mining workshops*, 929–936
26. Fu, C., Chen, T., Qu, M., Jin, W., and Ren, X. (2019). Collaborative policy learning for open knowledge graph reasoning. In *proceedings of the* 23th conference on empirical methods in natural language processing and the 9th international joint conference on natural language processing (EMNLP-IJCNLP), 2672–2681
27. Lin, X., Socher, R., and Xiong, C. (2018). Multi-hop knowledge graph reasoning with reward shaping. In *proceedings of the* 22th conference on empirical methods, 3243–3253
28. Lv, X., Gu, Y., Han, X., Hou, L., Li, J., and Liu, Z. (2019). Adapting Meta knowledge graph information for multi-hop reasoning over Fewshot relations. In *proceedings of the* 23th conference on empirical methods in natural language processing and the 9th international joint conference on natural language processing (EMNLP-IJCNLP), 3367–3372
29. Shen, Y., Chen, J., Huang, P., Guo, Y., and Gao, J. (2018). M-walk: learning to walk over graphs using Monte Carlo tree search. In *proceedings of the 32nd advances in neural information processing systems*, 6786-6797
30. Mnih V, Kavukcuoglu K, Silver D, Rusu A, Veness J, Bellemare M, Graves A, Riedmiller M, Fidjeland A, Ostrovski G (2015) Human-level control through deep reinforcement learning. Nature 518(7540):529–533
31. Lin, Y., Shen, S., Liu, Z., Luan, H., and Sun, M. (2016). Neural relation extraction with selective attention over instances, in *proceedings of the 54th annual meeting of the Association for Computational Linguistics*, 2124–2133
32. Williams R (1992) Simple statistical gradient-following algorithms for connectionist reinforcement learning. Mach Learn 8(3–4):229–256
33. Kingma, D. and Ba, J. (2014). Adam: a method for stochastic optimization. In *Proceedings of the 2nd International Conference on Learning Representations*
34. Toutanova, K., Chen, D., Pantel, P., Poon, H., Choudhury, P., and Gamon, M. (2015). Representing text for joint embedding of text and knowledge bases. In *proceedings of the* 19th conference on empirical methods in natural language processing, 1499–1509
35. Théo, T., Johannes, W., Sebastian, R., Éric, G., and Guillaume, B. (2016). Complex Embeddings for simple link prediction. In *proceedings of the 33th international conference on machine learning*, 2071–2080
36. Tim, D., Minervini, P., Stenetorp, P., and Riedel, S. (2018). Convolutional 2D knowledge graph Embeddings. In *proceedings of the 32th AAAI conference on artificial intelligence*, 1811-1818

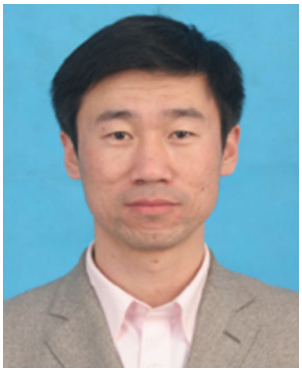
Ling Chen received his B.S. and Ph.D. degrees in computer science from Zhejiang University in 1999 and 2004, respectively. He is currently a professor with the College of Computer Science and Technology, Zhejiang University. His research interests include ubiquitous computing, AI, pattern recognition, and human computer interaction.

Jun Cui is currently a M.S. candidate in the College of Computer Science and Technology at Zhejiang University, Hangzhou, China. His research interest includes data mining and knowledge graph.

Xing Tang is currently a Ph.D. candidate in the College of Computer Science and Technology at Zhejiang University, Hangzhou, China. Her research interest includes machine learning, data mining, and knowledge graph.

Yuntao Qian received the B.E. and M.E. degrees in automatic control from Xi'an Jiaotong University, in 1989 and 1992, respectively, and the Ph.D. degree in signal processing from Xidian University in 1996. He is currently a professor with the College of Computer Science and Technology, Zhejiang University. His research interests include machine learning, signal and image processing, pattern recognition, and hyperspectral imaging.

Yansheng Li received the B.S. degree and the Ph.D. degree in information and computing science from Shandong University and in pattern recognition and intelligent system from Huazhong University of Science and Technology in 2010 and 2015, respectively. He is currently an associate professor with the School of Remote Sensing and Information Engineering, Wuhan University. His research interests include computer vision, deep learning, knowledge graph, and their applications in remote sensing big data analysis.



Yongjun Zhang received the B.S. degree in Geodesy, the M.S. degree in Geodesy and Surveying Engineering, and the Ph.D. degree in Geodesy and Photography from Wuhan University in 1997, 2000, and 2002, respectively. He is currently the Dean of the School of Remote Sensing and Information Engineering, Wuhan University. His research interests include aerospace and low-attitude photogrammetry, image matching, combined block adjustment with multisource data sets, artificial intelligence-driven remote sensing image interpretation, integration of LiDAR point clouds and images, and 3-D city reconstruction.