Computers, Materials & Continua

Tech Science Press

DOI: 10.32604/cmc.2024.051379

ARTICLE

Knowledge Reasoning Method Based on Deep Transfer Reinforcement Learning: DTRLpath

Shiming Lin[1,2,3], Ling Ye[2], Yijie Zhuang[1], Lingyun Lu[2,*], Shaoqiu Zheng[2,*], Chenxi Huang[1] and Ng Yin Kwee[4]

[1]School of Informatics, Xiamen University, Xiamen, 361104, China

[2]Key Lab of Information System Requirement, Nanjing Research Institute of Electronics Engineering, Nanjing, 210007, China

[3]School of Information Engineering, Changji University, Changji, 831100, China

[4]School of Mechanical & Aerospace Engineering, Nanyang Technological University, Singapore, 639798, Singapore

*Corresponding Authors: Lingyun Lu. Email: jyzluhit@gmail.com; Shaoqiu Zheng. Email: zhengshaoqiu1214@foxmail.com

Received: 04 March 2024 Accepted: 03 June 2024 Published: 18 July 2024

ABSTRACT

In recent years, with the continuous development of deep learning and knowledge graph reasoning methods, more and more researchers have shown great interest in improving knowledge graph reasoning methods by inferring missing facts through reasoning. By searching paths on the knowledge graph and making fact and link predictions based on these paths, deep learning-based Reinforcement Learning (RL) agents can demonstrate good performance and interpretability. Therefore, deep reinforcement learning-based knowledge reasoning methods have rapidly emerged in recent years and have become a hot research topic. However, even in a small and fixed knowledge graph reasoning action space, there are still a large number of invalid actions. It often leads to the interruption of RL agents' wandering due to the selection of invalid actions, resulting in a significant decrease in the success rate of path mining. In order to improve the success rate of RL agents in the early stages of path search, this article proposes a knowledge reasoning method based on Deep Transfer Reinforcement Learning path (DTRLpath). Before supervised pre-training and retraining, a pre-task of searching for effective actions in a single step is added. The RL agent is first trained in the pre-task to improve its ability to search for effective actions. Then, the trained agent is transferred to the target reasoning task for path search training, which improves its success rate in searching for target task paths. Finally, based on the comparative experimental results on the FB15K-237 and NELL-995 datasets, it can be concluded that the proposed method significantly improves the success rate of path search and outperforms similar methods in most reasoning tasks.

KEYWORDS

Intelligent agent; knowledge graph reasoning; reinforcement; transfer learning

Nomenclature

Term 1	Interpretation 1
Term 2	Interpretation 2

e.g.,

$\varnothing$	Porosity
s	Skin factor

1 Introduction

In 2012, Google introduced Knowledge Graph (KG), a technology that quickly became a research hotspot in fields such as data mining, databases, and artificial intelligence. Knowledge Graph uses a graph structure to describe knowledge and model relationships between entities. It expresses information in a form closer to human cognition and provides capabilities for organizing, managing, and understanding vast amounts of information. Essentially, Knowledge Graph is a large-scale semantic network, representing entities or entity attribute values as nodes and relationships or attributes between entities as edges. Currently, knowledge acquisition, knowledge reasoning, knowledge representation, and knowledge fusion, related to knowledge graphs, have become indispensable components in fields such as search query answering [1,2], big data analysis, intelligent recommendation [3,4], and data integration. They are widely applied in multiple industries.

However, most current research on knowledge graphs employs supervised learning methods, which require a large amount of labeled data and increase the cost. Additionally, whether obtained through manual curation or automated entity relation extraction, knowledge graphs often suffer from missing links, which significantly affect the performance of downstream tasks. Knowledge inference, therefore, plays a crucial role in filling in the gaps by mining paths to complete the knowledge graph. Thus, optimizing knowledge inference is crucial. However, existing approaches mostly rely on manually predefined rules and prior knowledge or lack interpretable algorithms. In contrast, reinforcement learning is suited for sequential decision problems, assisting human decision-making by learning how to interact with the environment. When it comes to policy selection, reinforcement learning focuses more on the environmental state, enabling a better understanding and explanation of behavior choices. By modeling knowledge graph research problems as path or sequence-related problems, such as modeling the selection of clean samples in distant supervision-based named entity recognition as a sequence labeling task or modeling relation inference as a path-finding problem, applying reinforcement learning algorithms can avoid relying on manually predefined rules or prior knowledge, addressing the issues of lacking interpretable models or only providing post-hoc interpretability, thus providing important research and application value.

The core idea of reinforcement learning is to learn how to better accomplish tasks through trial and error. However, in the knowledge graph environment, the existence of numerous invalid actions leads to low success rates in path searching for most reinforcement learning methods. On one hand, in knowledge reasoning tasks, RL agents need to learn and reason from the knowledge graph to select effective actions. This involves choosing the next action for the agent based on its current state and the knowledge it has acquired. Searching for a unique target node in knowledge graph multi-hop subgraphs is a challenging task, requiring the agent to perform breadth-first search or depth-first search in the knowledge graph and update its current state and known path information at each step, presenting challenges to the learning process of reinforcement learning. On the other hand, within a fixed knowledge graph reasoning action space, there are numerous invalid actions. For example, as shown in Fig. 1, both "castActor" and "profession" are invalid actions for the entity "United States". When the RL agent searches for the "United States" node, it wastes efficiency on these invalid actions. These two issues result in the agent struggling to obtain rewards in the initial stages and a low success rate in path searching.

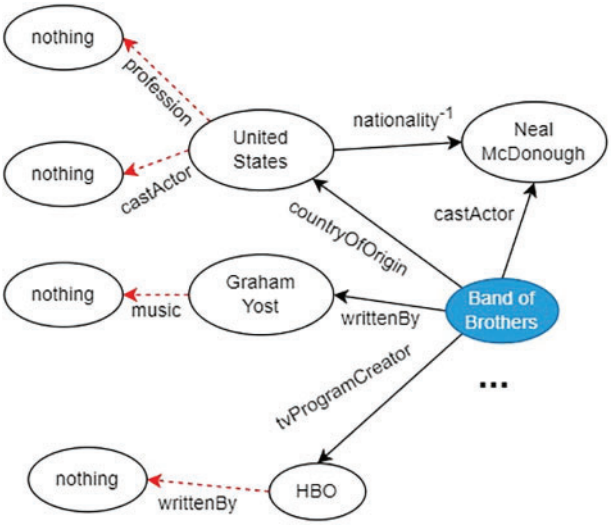

Figure 1: Invalid actions in the reasoning action space of a knowledge graph

Therefore, we believe that the learning process of intelligent agents should be progressive, that is, before learning complex multi-step reasoning, the intelligent agents should first learn how to select effective actions in a single step. In the exploration process, we discovered that transfer learning is an effective approach to solve this problem. Transfer learning is a machine learning method that transfers knowledge learned from one domain (referred to as the source domain) to another domain (referred to as the target domain). The goal of transfer learning is to improve the learning performance in the target domain by leveraging the knowledge from the source domain. Therefore, this paper applies transfer learning to the knowledge graph reasoning scenario and proposes a deep transfer reinforcement learning-based knowledge reasoning method called DTRLpath (Deep Transfer Reinforcement Learning path). Before supervised pre-training and retraining, a pre-task of searching for effective actions in a single step is added. The RL agent is first trained in the pre-task to improve its ability to search for effective actions. Then, the agent is transferred to the target reasoning task for fine-tuning, achieving path search training and improving its success rate in searching for paths in the target task.

The main contributions of this article are as follows:

(1) It proposes an effectiveness-driven pre-task to enhance the ability of RL agents to select effective actions in a single-step walking. The results of ablation experiments show that this pre-task effectively improves the agent's ability to choose effective actions.

(2) It proposes a Deep Transfer Reinforcement Learning path (DTRLpath), which is a knowledge reasoning method based on deep transfer reinforcement learning. It trains RL agents to perform single-step search for targets in the pre-task and then fine-tunes the agent on the target reasoning task. The results of fact prediction and link prediction experiments show that this algorithm outperforms similar methods in most reasoning tasks.

(3) Comparative experiments on the publicly available benchmark datasets FB15K-237 [5] and NELL-995 [5] demonstrate that the proposed model not only significantly improves the success rate of path mining but also achieves optimal performance in most knowledge reasoning tasks.

2 Related Work

Knowledge graph reasoning [6] can be broadly divided into three categories: embedding-based, neural network-based, and path-based methods. The premise of all three methods is the embedding of entities and relationships into vector spaces. However, the proposed method in this paper, called Deep Transfer Reinforcement Learning (DTRLpath), combines deep reinforcement learning with transfer learning to effectively address the problem of the agent making numerous ineffective actions during the early stages of reasoning. This section will focus on introducing the relevant work of deep reinforcement learning and transfer learning in the field of knowledge graph in both domestic and international settings.

2.1 Embedding-Based Knowledge Graph Reasoning Method

In the field of embedding-based knowledge graph reasoning methods, the TransE model proposed by Bordes et al. [7] is considered a classic model. This type of model represents entities and relationships as vectors in the same low-dimensional space, where the relationships between entities are interpreted as translation distances embedded in the low-dimensional space. The embedding vector of the head entity needs to be close to the embedding vector of the tail entity after the transfer of the relationship. Compared to traditional knowledge reasoning methods, TransE is simple, efficient, and effectively handles asymmetric relationships and one-to-one semantic relationships, which quickly sparked a wave of related research [8]. However, this model has some shortcomings that need to be optimized, and many variants of TransE have been proposed as a result. For example, TransD [9], TransR [10], TransH [11], etc. These models have shown good performance in knowledge graph reasoning and completion tasks, but most of these translation distance-based models are shallow representation learning models and are still not capable of multi-hop reasoning tasks in knowledge graphs.

2.2 Knowledge Reasoning Model Based on Tensor Decomposition

The knowledge reasoning model based on tensor decomposition first models the knowledge graph as a tensor, and then learns the latent representations of entities and relations through tensor decomposition, measuring the plausibility of the latent semantics of triples in the graph. Initially proposed by Nickel et al. [12], the RESCAL model associates each entity with a vector to capture its latent semantics. The DisMult model proposed by Yang et al. [13] constrains the relation matrix to be a diagonal matrix, thus alleviating the problem of excessive parameters and greatly enhancing learning efficiency. The ComplEx model proposed by Trouillon et al. [14] further extends the DistMult model by introducing complex-valued embeddings to model asymmetric and inverse relations. The ANALOGY model proposed by Liu et al. [15] is based on the RESCAL model and assumes that the linear mapping of relations is mutually exchangeable, which better models the analogy properties of entities and relations. Compared to distance-based models, tensor decomposition-based models can utilize tensor transformations to represent relation transformations in knowledge graphs and provide a reasonable geometric representation for their semantics, thus providing better interpretability.

2.3 Knowledge Graph Reasoning Model Based on Reinforcement Learning

Xiong et al. [5] first proposed a reinforcement learning model called DeepPath in 2017 for knowledge graph reasoning. It uses reinforcement learning to solve multi-hop reasoning problems in knowledge graphs. The model utilizes reinforcement learning to evaluate sampled paths, significantly reducing the search space. In the process of path reasoning, the reinforcement learning environment is modeled as a Markov Decision Process (MDP), transforming the multi-hop reasoning problem

in knowledge inference into a serialized decision problem. In this problem, each reasoning chain can be represented as a sequence of decisions. The reinforcement learning agent learns how to find paths that yield high rewards through interaction and feedback with the environment. It continuously optimizes its strategy during the path exploration process, enabling modeling and inference of multi-hop decision problems in knowledge graphs. Using this method, the intelligent agent autonomously learns how to select the next action based on previous decisions and received feedback, gradually mastering multi-hop reasoning in a knowledge graph. Das et al. [16] proposed a deep reinforcement learning approach called MINERVA, which guides the intelligent agent to find predicted paths based on input queries and obtains the correct answer entity given the specified head entity and query relationship. Wang et al. [17] proposed a rule-guided knowledge graph multi-hop reasoning model called RuleGuider, which improves the interpretability of reasoning. AttnPath [17] proposed by Wang et al. extends DeepPath with Long Short Term Memory(LSTM) [18] and graph attention [19], adding memory units and a forced backtracking reasoning mechanism to enhance the agent's ability to acquire rewards and success rate in reasoning.

2.4 Knowledge Graph Reasoning Model Based on Random Walk

Based on the random walk, the method uses the path as a feature to predict the inference result, and the feature of the path is interpretable, making it easier for people to understand its meaning. Path Ranking Algorithm (PRA), proposed by Lao et al. [20], is a classic random walk-based knowledge inference method. PRA first walks to discover the edge type sequences and uses logistic regression model to predict the missing edges in the graph structure based on these edge types as features. In order to further learn the semantic relationship and improve the performance of inference, Wang et al. [21] proposed a new PRA multi-task learning framework called Coupling PRA (CPRA). CPRA uses a multi-task mechanism for inference, consisting of two modules: relationship clustering and relationship coupling. The former is used to automatically discover highly correlated relationships, while the latter is used to couple these relationships for learning. Knowledge reasoning methods based on random walks can obtain specific reasoning paths for prediction results, thereby enhancing interpretability. However, in most of these methods, the path selection process is heuristic and the random walk for mining valuable reasoning paths lacks purpose, resulting in low efficiency and even introducing erroneous reasoning rules. Moreover, in the case of sparse and low connectivity knowledge graphs, extracting path features is inefficient and time-consuming [8].

2.5 Applications of Transfer Learning in Reinforcement Learning [22]

Transfer learning [23] refers to the application of existing knowledge or models to new tasks or environments. Through transfer learning, part of the knowledge of a model can be applied to a new task or environment without retraining the entire model, improving the performance of the model. Yaser et al. [24] applied transfer learning to the text summarization scenario in 2018 and proposed a reinforcement learning framework based on self-critical policy gradient method, which achieved optimal performance with only a few fine-tuning samples after pre-training. Ammanabrolu et al. [25] applied transfer learning to text adventure games based on knowledge graphs in 2019, and were able to learn reinforcement learning strategies faster in both computer-generated and human-created games, improving the quality of agent strategies. Liu et al. [26] applied transfer learning to multi-agent reinforcement learning in 2019 and proposed a scalable transfer learning method based on a novel MDP similarity concept, significantly speeding up multi-agent reinforcement learning while achieving better performance. From the above research work, it can be observed that transfer learning is suitable for scenarios where there is an abundance of pre-task samples but scarcity of target task samples. In

the context of knowledge graphs, although it is difficult for agents to obtain successful samples for the target task, each triple (h, r, t) in the knowledge graph contains two successful samples for single-step random walks, (h, r) and (t, r-1). Therefore, this paper treats single-step random walks as the pre-task and multi-step reasoning as the target task and proposes a deep transfer reinforcement learning method called DTRLpath.

3 Knowledge Reasoning Based on Deep Transfer Reinforcement Learning

In order to address the problem of invalid actions in a knowledge graph environment, this paper proposes a knowledge reasoning method called DTRLpath. Fig. 2 shows the overall framework of the DTRLpath method. Firstly, a validity-driven pre-training is conducted to improve the single-step traversal ability of the reinforcement learning agent, helping the agent learn to choose valid actions. The goal of the pre-training stage is to enhance the efficiency and accuracy of the agent in the traversal process. Next, the training continues with multi-step reasoning on the target task to improve the agent's ability to search for multi-step paths in the goal reasoning task. The goal of this training phase is to teach the agent how to reason through multiple steps in a knowledge graph to solve complex problems. Through the training in the two stages mentioned above, the DTRLpath method can effectively address the issue of invalid actions in the knowledge graph environment.

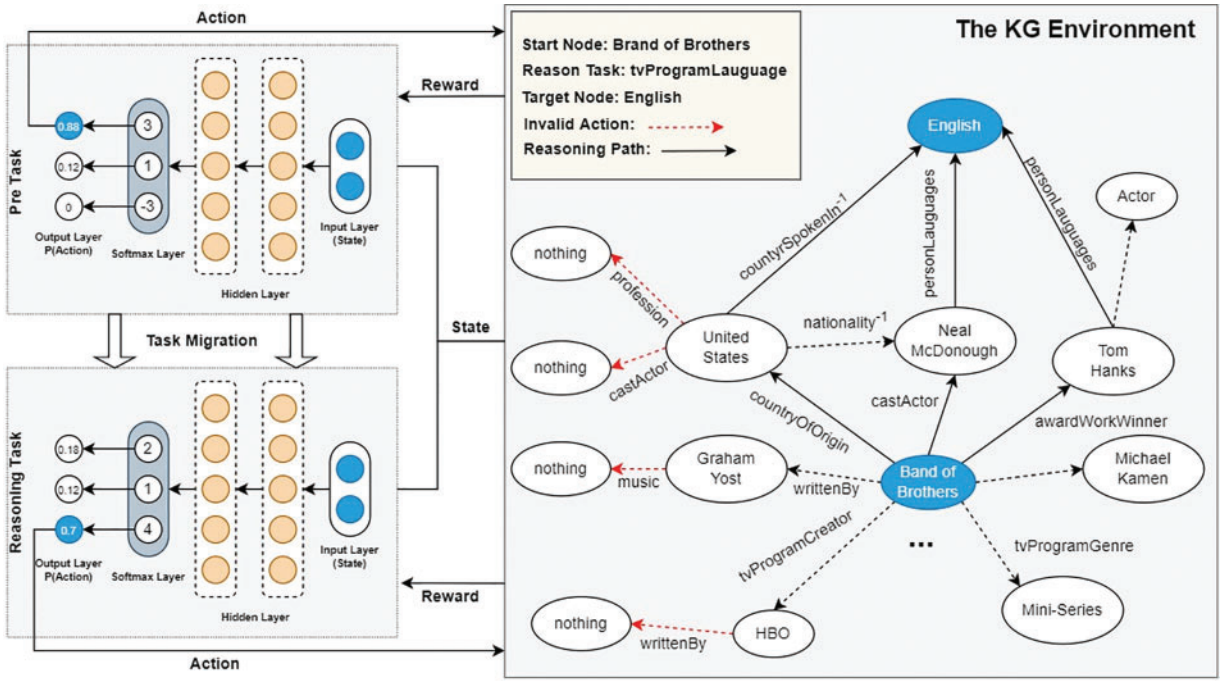

Figure 2: Framework diagram of knowledge reasoning model based on deep transfer reinforcement learning

3.1 Deep Reinforcement Learning

In this section, a knowledge graph will be modeled in a reinforcement learning environment. A knowledge graph K consists of a set of entities E, a set of relations R, and a set of RDF fact triples V. The knowledge graph K can be modeled as KE = <S, A, Y, P>, where S is the state space of the agent, A

is the action space of the agent, Y is the reward function of the agent, and P is the state transition policy of the agent. In the reinforcement learning environment, the agent can interact with the knowledge graph by executing actions. Each state s ∈ S represents the position of the agent in the knowledge graph and the state of the environment, while the action set A contains the operations that the agent can perform. The goal of an agent is to maximize cumulative rewards by selecting appropriate actions. The actions of the agent can be adding new facts to the knowledge graph, deleting existing facts, or querying specific information. Based on the actions performed by the agent, the state transition policy P moves the agent from one state to the next. The reward function Y can be designed according to the task requirements, providing positive or negative rewards based on the behavior of the agent. For example, in a question-answering task on a knowledge graph, positive rewards can be given when the agent provides correct answers, while negative rewards can be given for incorrect answers. Next, let's provide a detailed definition of the reinforcement learning space.

3.1.1 State Space

To represent the semantics of the entity set E in the knowledge graph, this paper adopts the embedding representation model TransE [7]. In the TransE model, entities are represented as continuous embedding vectors. This embedding representation model can capture the semantic associations between entities by learning the relationships between them. By representing entities as embedding vectors, we can better handle entities in the knowledge graph and incorporate them as part of the state space of intelligent agents:

$$s_t = Trans\ (e_t) \tag{1}$$

where e_t is the current entity, and s_t is the state representation vector of the current entity.

3.1.2 Action Space

The process of action selection for the intelligent agent is to achieve the transition from the current state to the next state. In this article, we consider the relationship set R in the knowledge graph as the action space of the intelligent agent. However, in order to enable the agent to perform reverse reasoning, we need to let the agent consider the inverse relationships as well. The inverse relationship refers to the fact that if there exists a relationship (e.g., A is related to B through relationship R) in the knowledge graph, there also exists its inverse relationship (e.g., B is related to A through the inverse relationship R-1). The inclusion of inverse relationships can help the intelligent agent consider different possibilities more comprehensively during the reasoning process. Therefore, in the action space, in addition to the original relationship set R, we also need to include the inverse relationships for all relationships. This way, the intelligent agent can consider reverse reasoning paths when selecting the next action, thereby searching the solution space more comprehensively and improving the reasoning ability of the intelligent agent in the knowledge graph. The action space is defined as follows:

$$A = \left\{r_1, r_1^{-1}, r_2, r_2^{-1}, \ldots, r_{|R|}, r_{|R|}^{-1}\right\} \tag{2}$$

where $r_x{}^{-1}$ represents the inverse relation of relation r_x.

3.1.3 Reward Setting

The environment evaluates the performance of an agent in completing or failing tasks and updates the agent's strategy based on positive or negative feedback in order to maximize rewards. The reward mechanisms applicable to the pre-training and fine-tuning stages differ due to the different

tasks involved. In the following Sections 3.2 and 3.3, we will discuss in detail the different reward mechanisms applicable to these two stages.

3.1.4 Policy Neural Network

The policy network maps the input state representation et to a probability vector for selecting different actions using a Fully-Connected Network (FCN). This neural network consists of two hidden layers and one output layer, which is normalized using the softmax function. For an input state s_t, the policy network will output the corresponding probability distribution:

$$d(s_t) = softmax(f(f(s_1 \times w_1 + b_1) \times w_2 + b_2)) \tag{3}$$

Among them, f is the activation function, w_x and b_x are the weights and biases of the hidden layer. $d(s_t)$ is a $|A| \times 1$ matrix, where each element represents the probability of choosing an action.

3.1.5 Parameter Optimization

The DTRLpath model uses policy gradient descent algorithm [27] to update the parameters of the policy network:

$$\nabla_\theta J(\theta) \approx \nabla_\theta \sum_{t=1}^{T} \log \pi(a = r_t | s_t; \theta) \gamma \tag{4}$$

where θ is the parameter that needs to be updated, $\log \pi(a = r_t | s_t; \theta)$ represents the probability that the policy network selects action r_t given state s_t, and γ represents the reward obtained from executing this action.

3.2 Pre-Training for Task Translation

Prior to training the path inference task, this paper first conducts effectiveness-driven pre-training for the reinforcement learning agent. The purpose of this pre-task is to assist the agent in learning to choose effective actions, thereby improving the success rate of single-step navigation.

3.2.1 Training Data Preprocessing

In the state space of a knowledge graph, we can obtain all combinations of states and valid actions by using the set of fact triplets V. Each triplet (e_{head}, r, e_{tail}) in V is split into two state-action pairs, (e_{head}, r) and (e_{tail}, r − 1), and then the identical pairs are merged together, resulting in the pre-trained valid action training set T_{valid}. This training set can be used to train a model for predicting valid actions given a specific state.

3.2.2 Reward Function for Pre-Tasks

The objective of pre-training task is to learn to select effective actions. When the agent selects action a_i in state e_t, if the tuple (e_t, a_i) is included in T_{valid}, the agent is rewarded with a positive reward; otherwise, no reward is given. The reward is defined as follows:

$$Y(e_t, a_i) = \begin{cases} 1, & if\,(e_t, a_t) \in T_{valid} \\ 0, & if\,(e_t, a_t) \notin T_{valid} \end{cases} \tag{5}$$

3.2.3 Pre-Training Algorithm

As the knowledge graph already contains all combinations of states and valid actions, the agent does not need to interact with the environment to obtain rewards. Therefore, this paper will use T_{valid} as the training dataset to train the agent offline. The pre-training algorithm is as follows:

Algorithm 1: Pre-training algorithm

Data: Preprocessed training set T_{valid}
Result: Policy network parameters for RL agents
1. *for* $(\boldsymbol{e_i}, \boldsymbol{a_i})$ *to* $\boldsymbol{T_{valid}}$ *do*
2. $\boldsymbol{S_t} \leftarrow \boldsymbol{Tans}(\boldsymbol{e_t})$
3. *Update Policy Network Parameters:*
$\nabla_{\theta} \boldsymbol{log\pi}\ (\boldsymbol{a} = \boldsymbol{a_i} | \boldsymbol{S_t}; \boldsymbol{\theta})\ \boldsymbol{\gamma}\ (\boldsymbol{e_t}, \boldsymbol{a_t})$;
4. *end*

3.3 Fine-Tuning Training of the Target Task

During the pre-training phase, the AI agent learns how to select effective actions and, after fine-tuning training, can be transferred to specific reasoning tasks. In the knowledge graph environment, the AI agent continues to learn how to complete multi-step reasoning tasks.

3.3.1 Reasoning Task

The goal of an inference task is to find the path that connects two entities, which is more complex than pre-training single-step tasks. Given a fact, including the start node (e_{start}), the inference task (r_{task}), and the target node (e_{target}), the agent needs to start searching from the start node and find the paths to all reachable target nodes except for r_{task}. By optimizing this objective, the agent can improve its performance in inference tasks.

3.3.2 Reward Function

To optimize the performance of the agent, three reward functions were designed for fine-tuning training of the target task:

1) Global accuracy: For our environment setup, the number of possible actions the agent can take can be very large. In other words, there are significantly more incorrect sequence decisions than correct ones. The quantity of these incorrect decision sequences grows exponentially with the length of the path. Given this challenge, the first reward function we added to the RL model is defined as follows:

$$r_{GLOBAL} = \begin{cases} +1, & if\ the\ path\ reaches\ e_{target} \\ -1, & if\ otherwise \end{cases} \tag{6}$$

There are three possible scenarios for the end of a path. The first scenario is when the agent reaches the target entity, and the reward +1 is given by the external source rather than the environment. The reward stored in the training samples used to train the agent remains 0, which is the reward given by the environment. The "otherwise" statement in the code implementation only includes one scenario, which is when the environment discovers that the selected relationship by the agent does not exist. In this case, the environment returns −1. However, it is important to note that the non-existence of the selected relationship by the agent does not mean the end of the search. The agent will discard the

current selection and choose another action until the maximum number of steps is reached. The last scenario is when the agent's search reaches the specified maximum number of steps. In this case, the reward returned by the environment is still 0, but the external source sets $r_{GLOBAL} = 0.05$.

2) Since fine-tuning training is a multi-step task, the environment cannot immediately provide immediate rewards for every action taken by the agent. To address this issue, this paper employs Monte Carlo methods [28]. When the agent successfully reaches a target node or reaches the predetermined step limit, a reward is given for each state-relation tuple(e_t, a_i) on the path p. Specifically, the rewards are set as follows:

$$\gamma = \begin{cases} \dfrac{1}{length\,(p)}, & if\ success \\ 1, & otherwise \end{cases} \tag{7}$$

The length of the path is represented by $length\,(p)$. For relational reasoning tasks, we observe that shorter paths often provide more reliable evidence for inference than longer paths. Limiting the interaction length between RL and the environment can also improve the efficiency of reasoning.

3) Path diversity: We train the agent with positive samples to find paths for each relationship. These training samples have similar state representations in vector space. The agent tends to find paths with similar syntax and semantics. These paths often contain redundant information as some of them may be related. To encourage the agent to find different paths, we define a diversity reward function based on the cosine similarity between the current path and existing paths:

$$r_{DIVERSITY} = -\frac{1}{|F|}\sum_{i=1}^{|F|}\cos\,(P, P_i) \tag{8}$$

where F is the set of existing paths, P is a newly found path instance, and P_i is an existing path instance. We hope to find more diverse path instances.

3.3.3 Fine-Tuning Training Algorithm

Fine-tuning is a key technique in transfer learning. By leveraging the knowledge of a pre-trained model on related tasks, it is possible to accelerate and enhance the performance of the model on a specific task. This is because the pre-trained model has already learned to extract useful features from its original training data, which can be transferred to new tasks. Moreover, training a deep learning model from scratch usually requires substantial computational resources and time. By using a pre-trained model and fine-tuning it, the required training time and computational costs can be significantly reduced.

After completing pre-training, the agent needs to transfer to the target task. Each target task represents a relationship in the knowledge graph. The agent learns to search for paths that connect the target relationship through fine-tuning training. To perform fine-tuning training, this paper extracts all triplets containing the target task from the dataset to form the training set, called trainset. The following is the fine-tuning training algorithm for the target task:

Algorithm 2: Fine-tuning the training algorithm for the target task

Data: The trainset for the target task
Result: Policy network parameters for RL agents
1. Overload the pre-trained RL policy network;

(Continued)

Algorithm 2 (continued)

2. For $(e_{start}, r_{target}, e_{target})$ to trainset do
3. $s_t \leftarrow Trans\,(e_t)$;
4. Steps = 0, success = False;
5. While steps < maxsteps and no success do
6. $d\,(s_t) = softmax(f\,(f\,(s_t \times w_1 + b_1) \times w_2 + b_2))$;
7. Randomly select action a_i based on $d\,(s_t)$, terminate if invalid;
8. A set of state-action tuples T records (e_t, a_i);
9. Execute action a_i, jump to next entity e_{next};
10. If $e_{next} = e_{target}$ then
11. Success = True;
12. End
13. $s_t \leftarrow Trans(e_t)$;
14. Compute the reward for each (e_t, a_i) in T;
15. Update Policy Network Parameters: $\nabla_\theta \sum_{i=1}^{|T|} log\pi\,(a = a_i|\,S_t; \theta)\,\gamma\,(e_t, a_i)$;
16. end
17. end

4 Experimental Design and Analysis

4.1 Dataset Selection and Analysis

When conducting experimental performance analysis, this study selected two widely used benchmark datasets in the knowledge reasoning field: FB15K-237 and NELL-995. These datasets provide rich information about knowledge graphs and are commonly used in knowledge reasoning research. By using these two datasets, this paper compares and evaluates different reasoning models. FB15K-237 dataset consists of 14.5 k entities, 237 relations, 310.1 k triple facts, and 20 reasoning tasks. It is derived from FB15K by removing redundant triple facts. NELL-995 dataset contains 7.5 k entities, 200 relations, 154.2 k triple facts, and 12 reasoning tasks. Taking FB15K-237 as an example, it is divided into three subsets: train, valid, and test, for training and validation purposes. The statistical data is as follows (see Table 1):

Table 1: FB15K-237 statistical data

Properties	Train	Valid	Test
Headings	13781	7652	8171
Triad	272115	17535	20466
Relationship	237	223	224
AverTriple	19.75	2.29	2.50
AverRelationship	6.78	1.58	1.68
AverEntity	2.91	1.45	1.49

During the training of the model in this paper, the softmax function is used as the activation function for the last layer of the policy network. The two fully connected hidden layers use ReLU as the activation function, with node sizes set to 1024 and 2048, respectively. The Adam [27] algorithm is chosen as the training optimization algorithm, with an initial learning rate of 0.001. The TransE model

is trained in this paper following the method in Fast-TransX [29], with an embedding dimension of 100. The pre-training batch size on the source task is set to 1000, with 2000 epochs of training. The fine-tuning batch size on the target task is set to 500, with 500 epochs of training. The DTRLpath proposed in this paper is implemented based on the TensorFlow framework and trained on an NVIDIA 1080 GPU.

4.2 Model Algorithm Evaluation Criteria

The commonly used metrics for evaluating the quality of knowledge reasoning based on deep reinforcement learning include Path Finding Success Rate (PFSR), Mean Average Precision (MAP) for Fact Prediction (FP), and Mean Average Precision (MAP) for Link Prediction (LP).

1) Path Finding Success Rate (PFSR): This metric mainly measures the ability of a reinforcement learning agent to explore paths. Specifically, it measures the ratio of the number of samples in each iteration of the training process where the reinforcement learning agent starts from an initial node, navigates, and eventually finds a path to the goal node, to the total number of samples.

$$PFSR = \frac{SuccessNum}{BatchSize} \tag{9}$$

SuccessNum represents the number of samples successfully searched for paths in each epoch, and *BatchSize* denotes the batch size. A higher *PFSR* indicates a stronger path search capability of the RL agent.

2) Mean Average Precision (MAP): The mean average precision for Fact Prediction (FP) and Link Prediction (LP) are used to measure the performance of fact prediction and link prediction, respectively. Fact prediction refers to determining whether a given triplet (eh, r, et) is correct or not. Link prediction refers to predicting the tail entity ex in a triplet (eh, r, ex) with a missing tail entity. The positive and negative samples used for testing on each dataset have a ratio of approximately 1:10, where the negative samples are generated by replacing the tail entity with a random entity.

To validate the effectiveness of the proposed method in this paper, we conducted a comparative experiment between the DTRLpath model and three other methods:

1) Path-based models. Path-based models have better ability for multi-step inference compared to embedding models. In this paper, we compared our model with the classical PRA algorithm [22] and the more effective DIVA(Variational Knowledge Graph Reasoner) method [30].

2) Embedding-based models. Embedding models have shown good performance in link prediction and fact prediction tasks. In this section, we compare our method with traditional embedding models TransE [7], TransD [9], TransR [10], and TransH [11].

3) RL-based models. DeepPath [16] is the earliest RL-based method proposed, and AttnPath [19] extends DeepPath with LSTM [20]and graph attention mechanism [21] as memory units. In addition, MINERVA [17] restructures the knowledge graph and proposes a query-based method, while DIVINE [31] introduces a generative adversarial reinforcement learning method.

4.3 Pathfinding Experiment

To analyze the path-searching ability of the model, this study compares the success rates of path searching between the DTRLpath model and similar methods such as DeepPath and AttnPath. After pre-training on a preliminary task, the intelligent agent underwent 500 epochs of training on the target task. The training results are shown in the graph below:

DTRLpath method, without the forced walking mechanism, achieves significantly higher success rates in path search compared to other methods, as shown in Table 2. Especially on the FB15K-237 dataset, the success rate is improved to 66.1%, outperforming other methods. This result is mainly attributed to the pre-training of the agent on the pre-task, which enables it to learn to select effective actions during single-step walking, thus significantly improving the success rate of single-step walking.

Table 2: Path finding success rate results (%)

Model	FBK15K-237	NELL-995
DeepPathNoPre	7.3	27.8
DeepPath	17.7	37.8
AttnPathForce	30.4	48.1
AttnPath	18.2	30.3
DTRLpath	**66.1**	**62.8**

We selected 12 inference tasks from NELL-995 and plotted the curve of average success rate of path search during fine-tuning training. The results are shown in Fig. 3:

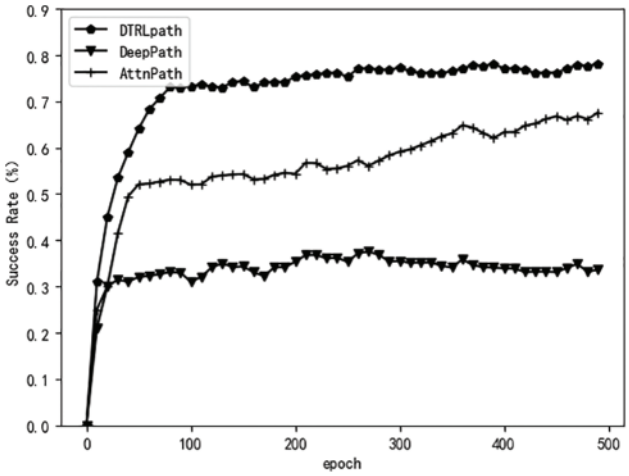

Figure 3: Average success rate of NELL-995 path search

From the results in Fig. 3, it can be seen that DTRLpath has a better success rate in path search on the 12 inference tasks of the NELL-995 dataset. This is determined by the pre-training on the pre-task. Moreover, due to the pre-training on the pre–task, our method can quickly achieve a high success rate in the first 0–100 epochs. Transfer learning not only improves the success rate of path search, but also accelerates the initial stage of training.

4.4 Fact Prediction Experiment

The fact prediction experiment aims to determine the truthfulness of a given triplet (eh, r, et). The model evaluates the triplet by scoring it, indicating whether it is a true fact. Traditional deep reinforcement learning-based methods continue the evaluation approach used in PRA, where the

number of paths that match the triplet is used as the score. Higher scores indicate a higher likelihood of the triplet being a positive sample.

This text describes the process of scoring a triplet (e_h, r, e_t) using a RL agent. The steps are as follows:

1) Start with eh as the initial node and input the state vector into the policy network.

2) The policy network maps the vector of the current state to the probabilities of choosing each action and selects one action accordingly.

3) The RL agent executes the action in the knowledge graph environment and moves to the next node.

4) If the chain of actions taken at this point forms a previously explored path, check if the current node is e_t. If it is, add 1 to the score and terminate. Otherwise, subtract 1 from the score and terminate.

5) Repeat steps 2) to 4) until the maximum number of steps is reached. Terminate and assign a score of 0.

The effectiveness of DTRLpath stems from the pre-training on pre-tasks, which greatly improves the success rate of path search for fact prediction. During testing, the retrained agent is used to find feature paths related to corresponding relations. Although it is a test, there is still one parameter update, which utilizes the diversity reward function (formula (8)). Therefore, the experiment compared the path search success rate of multiple tests of the DTRLpath model. The results of fact prediction are shown in Table 3, where the numbers after DTRLpath indicate the number of tests.

Table 3: Fact prediction MAP (mean average precision)

Model	FB15K-237	NELL-995
TransE	0.277	0.383
TransD	0.303	0.413
TransR	0.302	0.406
TransH	0.309	0.389
DeepPath	0.339	0.494
AttnPath	0.315	0.598
AttnPathForce	0.379	0.693
DTRLpath-1	0.341	0.632
DTRLpath-200	0.412	0.724
DTRLpath-500	**0.426**	**0.774**

Based on the experimental results in Table 3, it can be concluded that the proposed method in this paper achieves optimal factual prediction performance on both datasets. At the 500th testing iteration, our method achieves a result of approximately 0.426 on the FB15K-237 dataset and approximately 0.774 on the NELL-995 dataset, surpassing the DeepPath and AttnPath methods.

4.5 Link Prediction Experiment

Link prediction aims to predict missing entities. Specifically, given a test sample (eh, r, ex), our objective is to predict the missing tail entity ex. The model ranks candidate tail entities by scoring them.

In link prediction experiments, this paper generates negative samples and splits the dataset into training and testing sets. The testing method used in this paper is based on DeepPath, where the adaptability of each path in the sample is treated as a binary feature. Firstly, a classification model is pretrained on the training set and then used to score the tail entities in the testing set. Afterwards, the candidate tail entities are ranked based on the scores to obtain the most likely prediction for the missing entity.

As shown in Table 4, DTRLpath achieves the best performance in the link prediction experiment on the FB15K-237 dataset. It also performs well on the NELL-995 dataset. It is worth noting that our method shows more significant performance improvement on the FB15K-237 dataset compared to the performance on the NELL-995 dataset. This result can be explained by several factors. Firstly, the average path length in the FB15K-237 dataset is larger than that in the NELL-995 dataset, and the sparsity caused by invalid actions is more severe. Our method has certain advantages in alleviating the problem of invalid actions, thus showing more significant improvement on the FB15K-237 dataset. Secondly, our method is an improvement upon DeepPath, and the training and testing process is similar to the DeepPath method. Although it does not achieve the best performance on the NELL-995 dataset, our method has already shown significant improvement compared to the original DeepPath method.

Table 4: Link prediction MAP (mean average precision)

Model	FB15K-237	NELL-995
TransE	0.532	0.737
TransD	0.505	0.764
TransR	0.540	0.789
TransH	0.566	0.702
DeepPath	0.635	0.829
AttnPath	0.661	**0.858**
DTRLpath-1	0.655	0.821
DTRLpath-200	0.669	0.835
DTRLpath-500	**0.687**	0.849

4.6 Transfer Learning Ablation Experiments

In order to further investigate the impact of pre-task pre-training and target task fine-tuning training on knowledge reasoning in transfer learning, we conducted the following ablation experiments on the proposed DTRLpath method:

1) Pre-task deletion: We trained the RL agent directly on the target task to obtain the model onlyTarget. After training, we tested this model on the path search task and single-step walk task.

2) Target task fine-tuning deletion: To study the impact of target task fine-tuning training, we removed fine-tuning and the improved reward function on the target task. After pre-training on the pre-task, we obtained the model onlyPre and directly used it for the path search task and single-step walk task.

This paper conducts comparative experiments on the 12 inference tasks of the NELL-995 dataset, using the two generated incomplete models and the original model DTRLpath for path searching and fact prediction.

The results of the ablation experiments are shown in Table 5, where the testing frequency for the three models in the fact prediction task is set to 1.

Table 5: Ablation experiment results (MAP) of DTRLpath

Tasks	onlyTarget	onlyPre	DTRLpath
athletePlayersForTeam	0.663	0.218	**0.780**
athletePlayersInLeague	0.857	0.304	**0.965**
athleteHomeStadium	0.732	0.248	**0.890**
athletePlayersSports	0.829	0.331	**0.960**
teamPlaySports	0.716	0.249	**0.746**
orgHeadquaterCity	0.755	0.401	**0.814**
worksFor	0.694	0.332	**0.749**
bornLocation	0.683	0.272	**0.757**
personLeadsOrg	0.722	0.361	**0.795**
orgHiredPerson	0.668	0.257	**0.742**
...			
Overall	0.720	0.297	**0.820**

According to the results in Table 5 and Fig. 4, the onlyPre model performs poorly in the fact prediction experiment. In the path search experiment, it initially outperforms DTRLpath, but its success rate levels off and does not improve significantly during later training. This is primarily because the onlyPre model is trained only on single-step prediction tasks, while path search and fact prediction tasks require multi-step reasoning. Therefore, the onlyPre model can initially learn to choose effective tasks, but it lacks the ability to perform efficient multi-step reasoning.

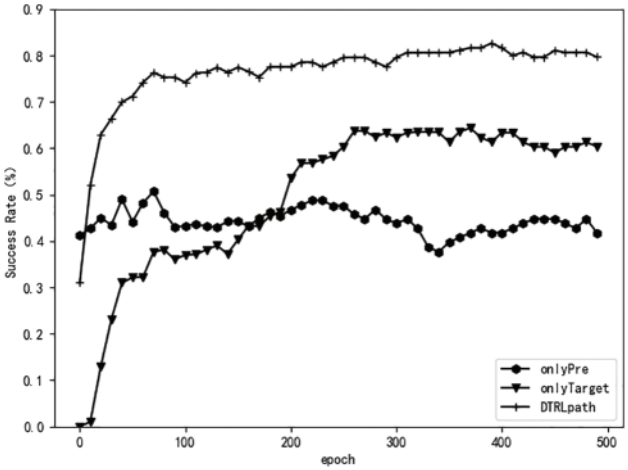

Figure 4: Average success rate of path search in DTRLpath ablation experiment

On the other hand, although the onlyTarget model is trained on the target task, it lacks training on pre-tasks, resulting in a lower ability to select effective actions and a lower path search success rate.

In contrast, the complete DTRLpath model starts with a higher path search success rate compared to the onlyTarget model, and improves more quickly, reaching near-optimal performance within the first 100 epochs. Furthermore, even after sufficient training, the DTRLpath model still outperforms the onlyTarget model significantly. Therefore, the experiment results indicate that pre-training with pre-tasks significantly accelerates agent reinforcement learning and improves model performance. It shows that pre-training with pre-tasks has a significant impact on improving the completion ability of the target task, and solely training on a single target task is insufficient to enhance model performance.

5 Conclusion and Future Work

The article proposes a knowledge reasoning method based on Deep Transfer Reinforcement Learning path (DTRLpath), which adds a pre-task of single-step search for effective actions before supervised pre-training and retraining. The RL agent is first trained to search for targets in the pre-task, improving its ability to search for effective actions. Then, the agent is transferred to the target reasoning task for path search training, improving its success rate in searching for target task paths. In the experimental section, based on the comparative experimental results of the FB15K-237 and NELL-995 datasets, it can be concluded that the proposed method significantly improves the success rate of path search and performs better than similar methods in most reasoning tasks. The final ablation experiment section verifies that pre-training of the pre-task and fine-tuning training of the target task have a significant improvement effect on the completion ability of the target task.The results of the erosion experiment show that onlyPath initially outperforms DTRLpath in the path planning task. This suggests that not only does the pretraining of the pretext task affect the fine-tuning training of the target task, but the fine-tuning training of the target task may also have a negative impact on the pretraining of the pretext task. It might be worth exploring a new direction by using a collaborative training approach with multiple agents to eliminate this impact.

The application of transfer learning in reinforcement learning is extremely widespread. Next, we will continue to study the application of transfer learning in knowledge reasoning in reinforcement learning. Traditional transfer learning is guided by experts on which transfer learning algorithm to use, as different algorithms excel in different domains and cannot be generalized. At the same time, we do not want too much human intervention, and we do not want experts to guide the intelligent system on which algorithm to choose every time a new scenario arises. Therefore, self-transfer learning (learning to transfer) is our next research goal.

Acknowledgement: The author sincerely appreciates the invaluable efforts of the editors and reviewers, as well as the invaluable support provided by the supervisor and family throughout the research process.

Funding Statement: This research was supported by Key Laboratory of Information System Requirement, No. LHZZ202202, Natural Science Foundation of Xinjiang Uyghur Autonomous Region (2023D01C55) and Scientific Research Program of the Higher Education Institution of Xinjiang (XJEDU2023P127).

Author Contributions: Conceptualization, Shiming Lin, Ling Ye, Yijie Zhuang, Lingyun Lu; methodology, Shiming Lin, Yijie Zhuang, Shaoqiu Zheng, Chenxi Huang; software, Ling Ye, Yijie Zhuang, Shaoqiu Zheng, Chenxi Huang; validation, Shiming Lin, Ling Ye, Shaoqiu Zheng; formal analysis, Shiming Lin, Yijie Zhuang and Chenxi Huang; investigation, Ling Ye, Yijie Zhuang, Shaoqiu Zheng; resources, Lingyun Lu, Shaoqiu Zheng, Chenxi Huang, Ng Yin Kwee; data curation, Shiming Lin,

Ling Ye, Yijie Zhuang; writing—original draft preparation, Shiming Lin, Yijie Zhuang; writing—review and editing, Shaoqiu Zheng, Chenxi Huang, Ng Yin Kwee; visualization, Shiming Lin, Ling Ye, Lingyun Lu, Shaoqiu Zheng; supervision, Chenxi Huang, Ng Yin Kwee; project administration, Lingyun Lu, Shaoqiu Zheng. All authors reviewed the results and approved the final version of the manuscript.

Availability of Data and Materials: All the study data are included in the article.

Conflicts of Interest: The authors declare that they have no conflicts of interest to report regarding the present study.

References

[1] X. Huang, J. Zhang, D. Li, and P. Li, "Knowledge graph embedding based question answering," in *Proc. 12th ACM Int. Conf. Web Search Data Min.*, Melbourne, VIC, Australia, 2019, pp. 105–113.

[2] L. Dong, F. Wei, M. Zhou, and K. Xu, "Question answering over freebase with multi-column convolutional neural networks," in *Proc. 53rd Annu. Meet. Assoc. Comput. Linguist. 7th Int. Joint Conf. Nat. Lang. Process.*, Beijing, China, 2015, pp. 260–269.

[3] Y. Gao, Y. Li, Y. Lin, H. Gao, and L. Khan, "Deep learning on knowledge graph for recommender system: A survey," arXiv preprint arXiv:2004.00387, 2020.

[4] Q. Guo *et al.*, "A survey on knowledge graph-based recommender systems," *IEEE Trans. Knowl. Data Eng.*, vol. 34, no. 8, pp. 3549–3568, 2020. doi: 10.1109/TKDE.2020.3028705.

[5] W. Xiong, T. Hoang, and W. Yang Wang, "DeepPath: A reinforcement learning method for knowledge graph reasoning," arXiv preprint arXiv:1707.06690, 2017.

[6] Y. Xu, J. Ou, H. Xu, and L. Fu, "Temporal knowledge graph reasoning with historical contrastive learning," *Proc. AAAI Conf. on Artif. Intell.*, vol. 37, pp. 4765–4773, 2023. doi: 10.1609/aaai.v37i4.25601.

[7] A. Bordes, N. Usunier, A. Garcia-Duran, J. Weston, and O. Yakhnenko, "Translating embeddings for modeling multi-relational data," in *Proc. 26th Conf. Neural Inf. Process. Syst.*, Lake Tahoe, Nevada, USA, 2013, pp. 2787–2795.

[8] X. Yi, M. Lan, X. C, J. Luo, G. Zhou, and P. He, "Survey on explainable knowledge graph reasoning methods," *Chinese J. Netw. Inf. Secur.*, vol. 8, no. 5, pp. 1–25 2022 (In Chinese).

[9] G. Ji, S. He, L. Xu, K. Liu, and J. Zhao, "Knowledge graph embedding via dynamic mapping matrix," in *Proc. 53rd Annu. Meet. Assoc. Comput. Linguist. 7th Int. Joint Conf. Nat. Lang. Process.*, Beijing, China, 2015, pp. 687–696.

[10] Y. Lin, Z. Liu, M. Sun, Y. Liu, and X. Zhu, "Learning entity and relation embeddings for knowledge graph completion," in *Proc. 29th Conf. on Artif. Intell. (AAAI)*, Austin TX, USA, 2015, pp. 2181–2187.

[11] Z. Wang, J. Zhang, J. Feng, and Z. Chen, "Knowledge graph embedding by translating on hyperplanes," in *Proc. 28th Conf. Artif. Intell. (AAAI)*, Québec, Canada, 2014, pp. 1112–1119.

[12] M. Nickel, V. Tresp, and H. P. Kriegel, "A three-way model for collective learning on multi-relational data," in *Proc. Int. Conf. Mach. Learn.*, Bellevue Washington, USA, 2011, pp. 809–816.

[13] B. Yang, W. Yih, X. He, J. Gao, and L. Deng, "Embedding entities and relations for learning and inference in knowledge bases," in *Proc. 2015 Int. Conf. Learn. Represent.*, 2015, doi: 10.48550/arXiv.1412.6575.

[14] T. Trouillon, J. Welbl, S. Riedel, E. Gaussier, and G. Bouchard, "Complex embeddings for simple link prediction," in *Proc. Int Conf. Mach. Learn.*, New York, NY, USA, 2016, pp. 2071–2080.

[15] H. Liu, Y. Wu, and Y. Yang, "Analogical inference for multi-relational embeddings," in *Proc. Int. Conf. Mach. Learn.*, Sydney, NSW, Australia, 2017, pp. 2168–2178.

[16] R. Das *et al.*, "Go for a walk and arrive at the answer: reasoning over paths in knowledge bases using reinforcement learning," in *Proc. 6th Int. Conf. Learn. Represent.*, 2018.

[17] H. Wang, S. Li, R. Pan, and M. Mao, "Incorporating graph attention mechanism into knowledge graph reasoning based on deep reinforcement learning," in *Proc. 2019 Conf. Empir. Methods Nat. Lang. Process. 9th Int. Conf. Nat. Lang. Process. (EMNLP-IJCNLP)*, Hong Kong, China, 2019, pp. 2623–2631.
[18] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural Comput.*, vol. 9, no. 8, pp. 1735–1780, 1997.
[19] P. Veličković, G. Cucurull, A. Casanova, A. Romero, P. Liò and Y. Bengio, "Graph attention networks," arXiv preprint arXiv:1710.10903, 2017.
[20] N. LAO and W. COHEN, "Relational retrieval using a combination of path-constrained random walks," *Mach. Learn.*, vol. 81, no. 1, pp. 53–67, 2010. doi: 10.1007/s10994-010-5205-8.
[21] Q. Wang, J. Liu, Y. Luo, B. Wang, and C. Lin, "Knowledge base completion via coupled path ranking," in *Proc. 54th Annu. Meet. Assoc. Comput. Linguist.*, Berlin, Germany, 2016, pp. 1308–1318.
[22] S. E. Li, Deep reinforcement learning,"in *Reinforcement Learning for Sequential Decision and Optimal Control*. Singapore: Springer Nature Singapore, 2023, pp. 365–402,
[23] Z. Zhu, K. Lin, A. K. Jain, and J. Zhou, "Transfer learning in deep reinforcement learning: A survey," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 45, no. 11, pp. 13344–13362, 2023. doi: 10.1109/TPAMI.2023.3292075.
[24] Y. Keneshloo, N. Ramakrishnan, and C. K. Reddy, "Deep transfer reinforcement learning for text summarization," in *Proc. 2019 SIAM Int. Conf. Data Min., Soc. Ind. Appl. Math.*, pp. 675–683, 2019. doi: 10.1137/1.9781611975673.
[25] P. Ammanabrolu and M. O. Riedl, "Transfer in deep reinforcement learning using knowledge graphs," arXiv preprint arXiv:1908.06556, 2019.
[26] Y. Liu, Y. Hu, Y. Gao, Y. Chen, and C. Fan, "Value function transfer for deep multi-agent reinforcement learning based on N-Step returns," in *Inter. Joint Conf. Artif. Intell. (IJCAI)*, 2019, pp. 457–463. doi: 10.24963/ijcai.2019/65.
[27] D. P. Kingma and J. Ba, "Adam: a method for stochastic optimization," arXiv preprint arXiv:1412.6980, 2014.
[28] R. J. Williams, "Simple statistical gradient-following algorithms for connectionist reinforcement learning," *J. Mach. Learn.*, vol. 8, no. 3–4, pp. 229–256, 1992. doi: 10.1007/BF00992696.
[29] "OpenKE: An open toolkit for knowledge embedding," in *Proc. EMNLP*, Brussels, Belgium. Accessed: Oct. 31–Nov. 4, 2018. [Online]. Available: https://github.com/thunlp/OpenKE
[30] W. Chen, W. Xiong, X. Yan, and W. Y. Wang, "Variational knowledge graph reasoning," arXiv preprint arXiv:1803.06581, 2018.
[31] R. Li and X. Cheng, "DIVINE: A generative adversarial imitation learning framework for knowledge graph reasoning," in *Proc. 2019 Conf. Empir. Methods Nat. Lang. Process. 9th Inter. Joint Conf. Nat. Lang. Process. (EMNLP-IJCNLP)*, Hong Kong, China, 2019, pp. 2642–2651.