Journal of Network Intelligence
Taiwan Ubiquitous Information
 ISSN 2414-8105 (Online)
Volume **7**, Number **3**, August **2022**

TMAPath: A Knowledge Inference Method Based on Transfer Learning and Multi-agent Deep Reinforcement Learning

Dairao He

College of Information Science and Technology
Dalian Maritime University
No. 1 Linghai Road, Dalian, Liaoning, China
dairaohe2022@163.com

Jingfeng Hu*

College of Navigation
Dalian Maritime University
No. 1 Linghai Road, Dalian, Liaoning, China
jingfenghu2022@163.com

*Corresponding author: Jingfeng Hu

Received May 30, 2022, revised July 2, 2022, accepted July 25, 2022.

Abstract. *Knowledge inference models aim to find the missing objects of user intent from knowledge graph, and are widely used in systems such as intelligent recommendation, Question Answering over knowledge graph, and assisted decision-making. However, the three existing knowledge inference models have some problems that have not been solved yet. Meanwhile, distributed knowledge inference models are unexplainable, and overlook the impacts of other key path information on the inference outcome and the correlations between entities in multi-hop paths, and hence do not have the ability to infer relations. Secondly, the traditional path-based knowledge inference models may suffer error accumulation in the face of large quantities of invalid paths, leading to a low success rate of path mining. Besides, these traditional models show poor performance when applied to paths of data sparsity, have high computation overheads. Finally, reinforcement learning-based inference models spend a lot of time on invalid or irrelevant paths, making the model inference less efficient. In the present work, TMAPath, a novel knowledge inference model based on transfer learning and multi-agent deep reinforcement learning, is proposed. The TMAPath model consists of a source task module and a target task module. First, during pretraining task in the source domain, single-step inference learning is employed to adjust the model parameters; then, these shared parameters are learnt through fine-tuned training in the target domain to realize multi-agent multi-step inference through transfer learning; last, the proposed approach is compared with other baseline models through tests on three public datasets, i.e., FB15K-237, NELL-995, and WN18RR, and the test results revealed that the proposed approach has better performance in knowledge inference than other baseline models.*
Keywords: Transfer learning, Knowledge inference, Deep reinforcement learning, Reward function, Multi-step inference.

1. Introduction.

Due to its weakness in the coverage of fact representations, the knowledge graph (KG), a technology that relies on network-based automatic extraction or manual processing of features, often suffers the problem of incompleteness, which makes it difficult to improve

the performance of downstream applications. The knowledge inference (KI) technique, which infers unknown information from existing facts, has become a hot topic and challenge in the field of KG research. In the KG, the new knowledge obtained through KI includes new entities and new relations, where the relation inference (link prediction) includes single-step inference and multi-step inference. The major methods of KI at present are distributed KI, path-based KI, and reinforcement learning-based KI. Within these approaches, there are three main challenges.

The mainstream models of distributed KI are the tensor factorization (TF) model and the embedding representation model. The TF model lowers the computation complexity by reducing the dimension of the high-dimensional KG. Moniruzzaman et al. [1] proposed a new TF model for fine-grained type inference of KGs, which improved the KI efficiency. Sedghi and Sabharwal [2] applied the idea of tensor factorization to knowledge completion for generics. The embedding representation model maps the triple of network semantic information onto a low-dimensional continuous vector space to learn the potential attributes of entities and relations; a smaller distance between the vectors indicates a higher similarity between two objects, and the score function of the triple is calculated to fulfill knowledge completion tasks like link prediction and fact prediction [3, 4, 5]. The first embedding representation model was proposed by Bordes et al. [6] in 2013, followed by improved variants like TransH, TransR, and TransD, which have later seen wide adoption in different fields and scenarios. Nie and Sun [7] proposed a new combined model called text-enhanced KGembedding (TKGE), which combined TransE and LSTM to learn KG embeddings from potential features of data to infer entities, relations, and texts. Zhang et al. [8] proposed a novel framework IterE, which used rule-based inference to improve the quality of sparse entity embeddings and the model's performance for link prediction. These distributed KI models boast such advantages as low complexity, strong usability, simplicity, and high efficiency, but the shortcomings remain. The first challenge is that distributed inference models do not adequately model relationships and are not interpretable. First, most distributed KI models parse the information like entities and relations in KGs merely from the numerical perspectives of vector and tensor, resulting in non-explainability of the inference results. Second, these models overlook the decisive impacts of other key path information on the inference outcome and the correlations between entities in multi-hop paths, thus unable to infer relations.

Path-based KI is a method that introduces the information hidden in the path rules between two entities into the KI process, in which the weights of relations between entities in the path are calculated to redesign the semantic information to complete the missing links between entities and improve the model's inference performance. The most classical path-based KI model adopts the path sorting algorithm [9], which explores the set of paths of two entities to predict possible relations. After that, more path-based KI models were developed. Vaigh et al. [10] interprets the indirect relations between entities by introducing property paths to the model and designed a novel entity correlation measurement method, which improved the performance of entity linking. However, the shortcomings of existing path-based approaches undermine the role of path information in the inference task. The second challenge is that the traditional path inference model treats each path equally, which is time-consuming and vulnerable to irrelevant paths, while it cannot adequately capture the sparse object features that occur less frequently and is less effective. First, these traditional methods may mine lots of invalid paths, which leads to error accumulation and a low success rate of path mining. Second, when applied to sparse paths, most of these models demonstrate poor performance, suffer a high computation cost, and cannot learn multi-step inferences. In these years, introducing the technique of reinforcement learning (RL) to path-based KI has drawn broad attention

from researchers. In 2017, Xiong et al. [11] proposed DeepPath, the first KI model that incorporated deep reinforcement learning (DFL) to KI, which used a policy-based agent to reason in the space environment of the KG, where the entities and relations in the KG were used as the state space and the action space. However, RL-based KI approaches represented by DeepPath suffer a low success rate in path search. The third challenge is that for spatially similar objects or when paths are encountered missing, reinforcement learning knowledge inference models need to learn the inference process repeatedly, consuming resources on top of invalid actions. First of all, most RL-based KI models use a single agent and are not efficient in the inference task; moreover, when the single agent fails to work, the model needs to start learning all over again, which results in poor robustness of the model. Second, there are many invalid actions in the action space of the agent, and if these invalid actions are selected in single-step inference or multi-step inference tasks, the model's performance will decline.

To address these problems, we propose TMAPath, a knowledge inference model that combines transfer learning and multi-agent deep reinforcement learning, which employs transfer learning to perform KG inference and introduces multiple agents to improve the model's efficiency and robustness. First, in the pretraining module in the source domain, single-step inference is achieved through a special reward mechanism; then, the inference result is transferred to fine-tuned training in the target domain where multiple agents perform multi-step inference; last, the proposed model is compared with other baseline models through tests on three public datasets, i.e., FB15K-237, NELL-995, and WN18RR, and the test results show that our model achieve a higher success rate of knowledge inference than other models.

The major contribution of our work is as follows:

1) A novel KI method that combines transfer learning and multi-agent deep reinforcement learning is proposed. In the source domain and the target domain, the inference task is converted into a sequential decision-making problem, which improves the agents' capacity for multi-step inference of relations in paths and the explainability of the inference.
2) In the source domain, a special reward mechanism is designed, and an extra special network is introduced to pretrain the triples. The reward mechanism allows the model to learn more fact triples, and the network like TransE that trains the triples can improve detection of the fact triples and representation of vectors. Ablation experiments show that pretraining in the source domain improves the success rate of single-step walk in the search of valid paths, and solves the problems of error accumulation and high computation overheads.
3) In the target domain, a multi-agent collaborative prediction algorithm is put forward so that the global agent collects the learning experience of other agents and updates the global network parameters; then, the updated parameters are replicated synchronously to other local agents to maximize the gains of the multi-agent system and minimize the overall loss, which will improve the model's robustness and efficiency, and reduce the chance of selecting invalid paths by the agent in multi-step inference.

2. **Related Works.**

2.1. **Development of deep reinforcement learning and multi-agent learning.**

There are different types of reinforcement learning (RL) methods. By the dependence on the environment model, these methods can be classified into model-based RL and model-free RL; by the policy updates, the methods can be classified into value-based

RL and policy-based RL. Value-based RL approaches like Q-learning [12] and deep Q network [13] optimize the action value functions; whereas policy-based approaches directly optimize the policy. Both methods can be combined to generate actor-critic algorithms or QT-Opt algorithms.

Deep reinforcement learning (DRL) is a technique that relies on the strong function approximation and feature representation capacity of deep neural networks (DNNs) [14, 15]to fit the functions of states, actions, policies, and values in reinforcement learning, which solves the curse of dimensionality and has been widely applied to autopilot, e-sports, machinery control, and recommender systems. Chen et al [16] combined reinforcement learning and imitation learning, and designed an auxiliary network in the RL framework to ensure stability of end-to-end autopilot. Ye et al. [17] designed a DL framework for multi-player online battle arena (MOBA), which learns successfully the control policies of complex actions in the game. Liu et al. [18] put forward a DL framework to model interactive recommender systems, which improved the quality of recommendation. Wang et al. [19, 20] proposed a trust reward and punishment method aims to achieve trust incentive consensus.

Multi-agent deep reinforcement learning (MADRL) is a technique that applies DL to multi-agent system control and implementation, and by the relation of the reward functions, it can be divided into three categories: cooperative models, competitive models, and mixed cooperative-competitive models. The cooperative models need collaborations between multiple agents to achieve the maximum reward. Competitive models, however, rely on the min-max theorem and entail games between agents to achieve the maximum reward. The mixed models basically stay in a static state where the agents remain independent from each other and there is no need to learn the actions of other agents.

Multi-agent reinforcement learning can be described by random games, the architecture of which is shown in Figure 1. The alliance state of multiple agents serves as the environment of the system, and the reward function is designed as per the field environment and the learning objective. Overall, MADRL can improve the efficiency of task completion through multiple agents working in parallel, and when some agents fail to function, other agents fill in, which improves the robustness of the model. Nonetheless, MADRL models are not free from problems or challenges: as the number of agents increases, the complexity of the system grows exponentially; besides, each agent is subject to multiple impacts from the environment and other agents, which makes it difficult to define the learning objective and converge to an optimal solution.

2.2. **Transfer learning in reinforcement learning.**

Transfer learning (TL) is a technique that obtains knowledge from the source domain to address the problem of insufficient training data in the target domain. By the tasks in the source domain and the target domain, TL methods can be divided into three types: inductive transfer learning, unsupervised transfer learning, and transductive transfer. Transfer learning in reinforcement learning tasks is called transfer reinforcement learning [10], whose premise is the similarity between the source domain and the target domain for knowledge transfer to avoid negative transfer.

Transfer reinforcement learning can accumulate rewards by enhancement of the three targets: learning speed enhancement, progressive enhancement, and quick launch enhancement. The learning speed is improved by transferring sufficient prior knowledge in the source domain and reducing the interactions between the target domain and the external environment. Progressive enhancement is to measure the improvement of the final performance. Quick launch enhancement is measured by the performance enhancement at the initial stage of learning.

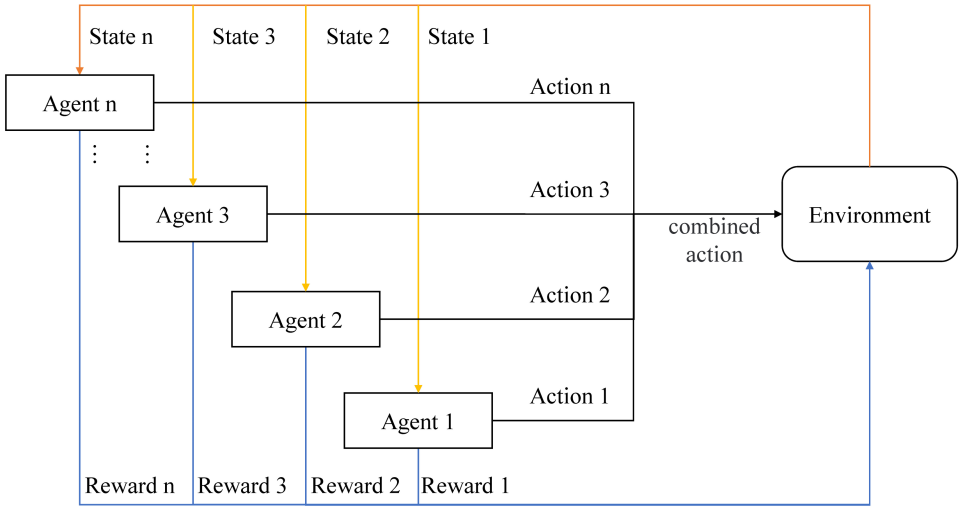

FIGURE 1. The basic framework for multi-agent deep reinforcement learning

2.3. **Development of reinforcement learning in knowledge inference.**

Reinforcement learning-based KI relies on the strong decision-making capacity of RL to model the search of relations in paths into sequential decision-making. Specifically, the whole KG, as the environment for the RL agents, models the entities and relations as the state space and the action space, and the agents walk on the KG and search the paths towards the target entity. For RL-based KI, the DeepPath model [11] is the first attempt that introduced DRL into KI. Das et al. [21] put forward the MINERVA model, which performs efficient search from the initial entity to the target entity and has strong performance in finding answers. The M-walk model proposed by Shen et al. [22] encodes the historical relation paths by an RNN, and combines the Monte Carlo tree search (MCTS) and the neural policy to forecast the target entity. In the Multi-Hop model, Lin et al. [23] employed the soft reward mechanism to achieve multi-hop reasoning. The DIVINE model proposed by Li et al. [24] used generative adversarial imitation learning (GAIL) to adaptively learn the reasoning policy and reward function. The collaborative policy learning (CPL) model put forward by Fu et al. [25] adopts a new collaborative policy framework, which extracts facts and searches paths from the corpus. Wang et al. [26] proposed the attention-based deep reinforcement learning (ADRL) model, which infers the weights of entities and relations using the self-attention network and uses the actor-critic method in reinforcement learning to improve the model's reasoning efficiency. Li et al. [27] put forward the model called MemoryPath, which introduces a memory component that combines the long and short-term memory network and the graph attention mechanism into the deep learning framework to solve the problem of over-dependence of reinforcement learning-based knowledge inference on pretraining. The DAPath model proposed by Tiwari et al. [28] addresses the memory of relations in the path by combining the graph self-attention mechanism and GRU, thus eliminating the pre-training or fine-tuning process. Instead of focusing on traditional reinforcement learning path query optimization, the work in this paper focuses on employing multiple intelligences and transfer learning to address the model's ability to process similar and missing objects quickly when inferring them, as well as to improve path relation inference by transforming the inference task into a sequential decision problem.

3. **Model Design.**

3.1. **Architecture of the reinforcement learning-based model.** In the present work, a knowledge inference method that incorporates transfer learning and multi-agent deep reinforcement learning is proposed, the central idea of which is model-based cross-domain transfer learning. Figure 2 shows the architecture of the TMAPath model, which consists of two modules: the source task module and the target task module.

In the TMAPath model, the source task module and the target task module have different state-action spaces, but the data sources corresponding to the source domain environment and the target domain environment share high similarity. First, the model parameters are adjusted constantly in the single-step inference task in the source domain, and the shared parameters are learnt from the Markov decision-making process in the source domain; then, the shared parameters are used to initialize the Markov decision-making process in the target domain to optimize the learning capacity and efficiency of multi-step inference in the target-domain task.

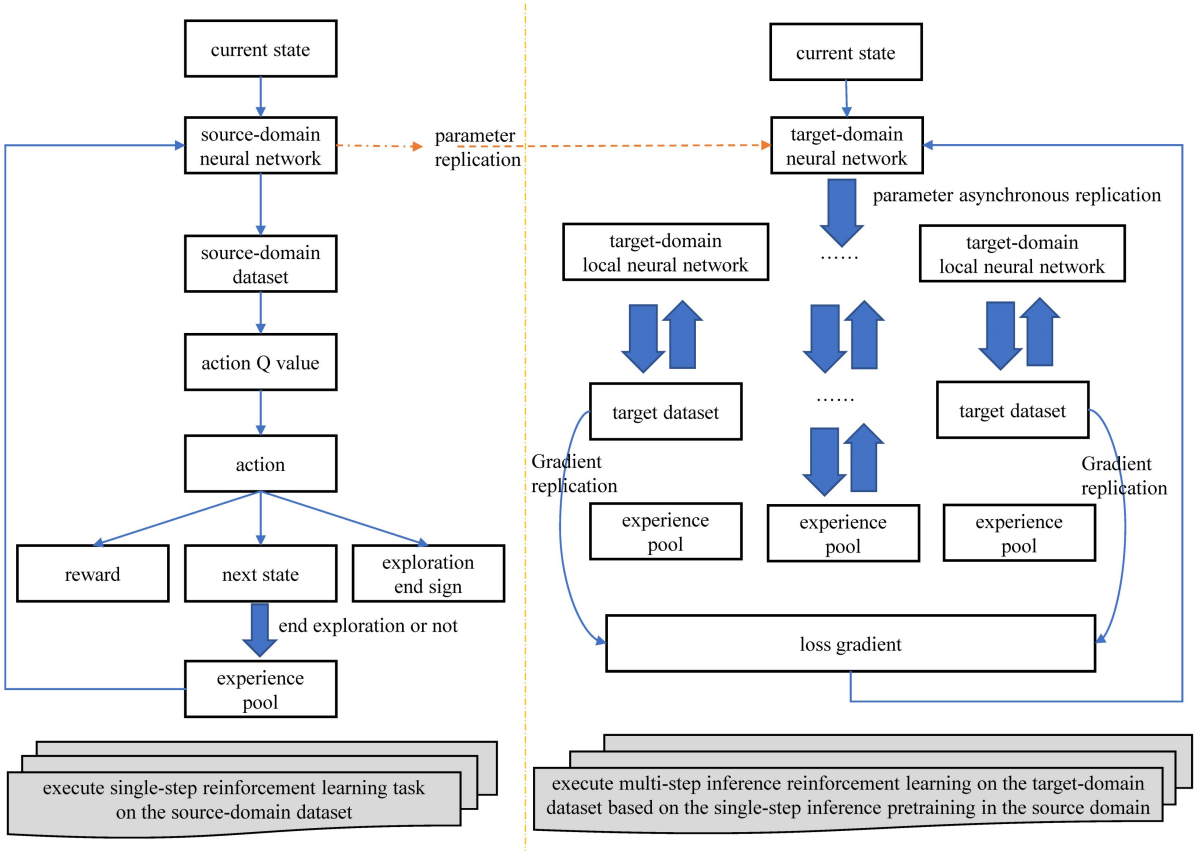

FIGURE 2. Architecture of TMAPath

3.1.1. *Modeling of the deep reinforcement learning environment.*

In deep reinforcement learning, the path search in KG inference is considered as a Markov decision-making process. This process forecasts the next state and the expected reward through the current state and action, and the next state does not necessarily remain unchanged. The state transfer and action selection of the RL agents are completed in the whole KG, so it is necessary to model the environment of DRL of theKG, and the environment remains consistent in the whole training process. For a given relation r_0, the environment set consisting of the RDF triples is defined as follows:

$$G_{r_0} = \{(h, r, t) | h, t \in E, r \in R, r \neq r_0, r \neq r_0^{-1}\}, \tag{1}$$

where G_{r_0} represents the agent environment excluding the relation r_0 and the inverse relation r_0^{-1}. In the present work, a reverse path search algorithm is designed to convert the reasoning process into a decision-making process:

$$D_{RL} =< S_M, A_M, P_M, R_M, \gamma >, \tag{2}$$

where S_M represents the state space of the agent, which may be a finite set space or a continuous infinite space; A_M represents the action space of the agent; P_M represents the reward function of the agent; R_M represents the state transfer policy of the agent; γ represents the discount factor; and D_{RL} represents the Markov decision-making process in deep learning.

Specifically, reinforcement learning-based path inference can be described as follows. The environment is modelled into a Markov decision-making process, where the action space in the environment is the set of edges of the KG, and the decision-making agent is set as a policy-based network. When the state space of the environment changes, the agent selects the actions according to the set policy network to transfer the initial state of the agent to a new state. At this time point, the reward function is used to judge whether the new state has reached the target state. During iterative training, the agent gradually improves its capacity to infer relations in the path. Through constant interactions between the external environment and the agent, reinforcement learning generates a path decision-making sequence from the source entity to the target entity, where the sequence contains the entity-relation information between two entities.

(1) State Space

The state space of reinforcement learning refers to any information of the external environment of the agent, including the primary perception data and the accumulated data processed based on the primary perception data. Usually, the more sufficient the valid state information is preserved, the stronger the agent's learning capacity is. The state information generally has Markov property, that is, at the time point t_0 and when the state information of the environment is known, the state distribution at the time point $t(t > t_0)$ is irrelevant of the state before the time point t_0. The state space of our model S_M is comprised of the entity set E in the KG, and the *TransE* model is used to represent the character-level embeddings of entities, which is defined as follows:

$$s(t) = TransE(e_t), \tag{3}$$

where e_t is the entity at the time point t, and $s(t)$ is the state vector representation at the time point t. In deep reinforcement learning models, the state information needs to capture the entity position of the agent in the KG, and after one action is selected, the state of the model changes, i.e., the path of the reliance relations between agents is transferred to another entity position. Therefore, the change of state refers to a process in which the agent moves from the position in the entity e_0 at the start time t_0 to the position in the target entity e_{target} at the end time point t_{target}, which is formalized into the following:

$$S_M = \{TransE(e_0), TransE(e_{\text{target}})\}. \tag{4}$$

(2) Action Space

An action refers to the feedback an agent provides after acquiring the state of the external environment, whose representations can be either discrete or continuous. For a KG inference task, an action means the agent selects an output path in the set of relations of KGs to move forward, and the number of action spaces relies on the number of neighboring entities of the current entity. In a certain relation, an entity pair $(e_{\text{source}}, e_{\text{target}})$ is known, and the reinforcement learning agent walks amid the entities to achieve state transfer and find the valid path of the optimal information quantity to link the pair of

entities, thereby fulfilling the inference task from the head entity to the target entity. In the present work, an entity and its output relation are encoded into a candidate action, and the correlation between the action of the current entity and the entity and relations of the entity in the previous state is not considered. From the head entity e_{source}, the agent selects the relation r by the reward to link with the next entity, and through selections of multiple actions, the agent reaches the target entity e_{target}. It has been proved in literature that introducing the inverse relations into the action space can allow the agent not only to master reverse reasoning and uncover hidden inferences, but to revoke wrong decisions automatically. Therefore, the reverse relations of all relations are introduced into the action space, which is represented as follows:

$$A_M = \{r_1, r_1^{-1}, r_2, r_2^{-1}, ..., r_{|R|}, r_{|R|}^{-1}\}, \tag{5}$$

where r_i^{-1} is the reverse relation of r_i, and R is the quantity of actions. In the framework of reinforcement learning, there are two different actions: valid actions and invalid actions. Valid actions refer to extending the paths of the current entity based on the existing or potential relations, whereas invalid actions mean the relations that are not present in the current entity links.

(3) Reward Function

In reinforcement learning, the reward is the given to the agent when it perceives the state change and responds with actions, and the reward mechanism is set as per the actual scenario. The reward functions include the positive activation function and the negative penalty function. In KG inference, once an agent succeeds or fails in completing a task, the environment provides different rewards to the agent; the agent then updates its policy as per the provided reward to maximize the reward. Our model comprises of pretraining in the source domain and fine-tuned training in the target domain, and different reward functions are adopted for these two tasks.

(4) State Transfer

State transfer refers to the probability distribution that an agent takes actions in response to the current state and transfers to the next state. In the KG environment, the agent stays at the position of the current entity e_t, and selects the relation linked to the current entity e_t as the next action to transfer the agent to the position of the next entity e_{t+1}. The state transfer function is defined as follows:

$$f : p\{s_{t+1}|(s_t, a), s \in S_M, a \in A_M\}, \tag{6}$$

where s_t is the current state; s_{t+1} is the new state of the next entity; and a refers to an action the agent takes. A special stop action a_{stop} is set to indicate that the agent no longer selects any new state and stops the state transfer.

3.1.2. *Modelling Principle.*

(1) Double Deep-Q Network

Be it for the single agent in the source domain or for the multiple agents in the target domain, the TMAPath model adopts the double deep Q network for Q value generation. When the nonlinear function approximator is used to represent the Q function, the traditional Q-learning algorithm shows unstable performance. The deep-Q network (DQN) with an experience replay pool provides a solution to this problem. The agent stores the samples with learning experience to the experience replay pool, and updates Q-learning through small-scale samples obtained by uniform sampling. Meanwhile, the DQN updates the parameters using the online network $Q(s, a; \theta_i)$ and the target network $Q(s, a; \theta_i^{-1})$. These two neural networks have the same structure, and the target Q value is calculated

by the following equation:

$$y_t^{DQN} = r_t + \gamma \max_{a'} Q^-(s_{t+1}, a', \theta^-). \tag{7}$$

The loss function at the position i of each iteration is minimized to train the Q network, and the loss function is defined as follows:

$$L_i^{DQN}(\theta_i) = E_{s,a\sim\rho(\cdot)}\left[(y_t^{DQN} - Q(s,a;\theta_i))^2\right]. \tag{8}$$

However, the traditional DQN suffers the problem of overestimation of the action value in knowledge inference, which leads to poor performance of Q-learning. Therefore, the double deep Q network (DDQN) is used in the present work so that two different networks are used in selection and evaluation stages. In the calculation of the Q value function, the parameter θ of the current Q network is based on to select actions, and then the parameter θ^{-1} of the target Q network is used to assess the action. The target Q value can be calculated by the following equation:

$$y_t^{DDQN} = r_t + \gamma Q(s_{t+1}, \arg\max_a Q(s_{t+1}, a'; \theta); \theta'). \tag{9}$$

The loss function is calculated by the following equation:

$$L_i^{DDQN}(\theta_i) = E_{s,a\sim\rho(\cdot)}\left[(y_t^{DDQN} - Q(s,a;\theta_i))^2\right]. \tag{10}$$

(2) Multi-Agent Deep Reinforcement Learning Module

This module performs multi-step inference training in the target domain, and adopts a multi-agent collaborative prediction algorithm. Figure 3 shows the architecture of the MADRL module.

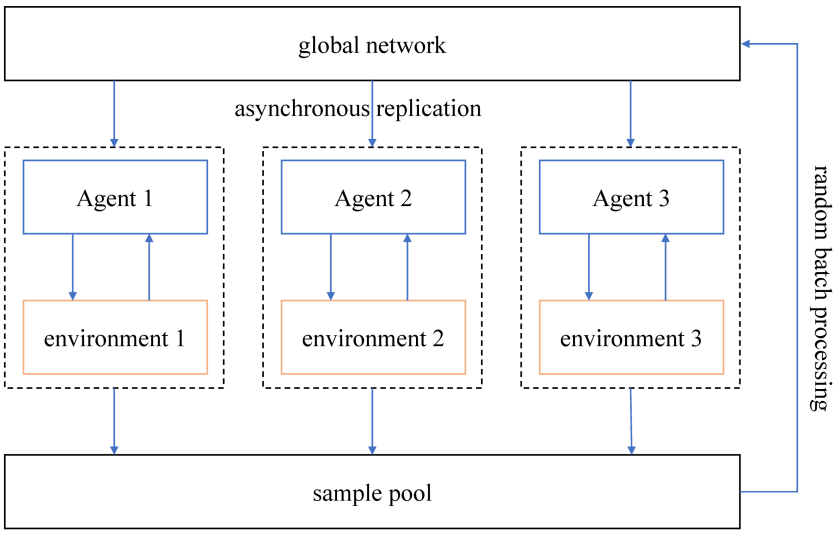

FIGURE 3. Architecture of the MADRL module

The multi-agent collaborative prediction algorithm has a structure similar to that of the "asynchronous advantage actor critic" algorithm, consisting a global agent and multiple local agents. The agents in the global network and the local network are in the same environment and parameters, but there are transfers between the local network and the global network, and the triples collected by each agent differ. Specifically, multiple local agents learn the model parameters in a distributed environment, and each local agent experiences cyclic pauses in learning; the gradient information the agents learn from parallel training is summarized to their own gradient and update the global network parameters.

Then, the global network parameters are copied to other local networks through asynchronous updating to generate a "global-local" collaborative forecast reinforcement learning algorithm.

The multi-agent collaborative forecast algorithm shows considerable advantages in fine-tuned training in the target domain. First, it has strong stability. The instability of reinforcement learning models, to a great extent, results from the correlations between the series of data obtained through RL. The multi-agent collaborative forecast algorithm, however, breaks the correlations between data through asynchronous replication: the multiple agents randomly select an action in the exploration stage to learn experience, and then rely on the experience to learn the optimal action; as the exploration and learning processes differ, the sample correlation between agents is reduced. After that, different agents adopt different parameters in the exploration stage, which increases the diversity of global exploration, improves the model's robustness and reasoning efficiency, and reduces the selection of invalid paths in multi-step inference.

In collaborative forecast, the global agent serves as the test subject. Though the network of the global agents and that of the local agents are the same, any agent, in theory, can serve as the forecast subject; nonetheless, the parameters of the global agents are updated more frequently than those of the local agents, and part of the gradient information is from the aggregation of the individual agents, so the global agent performs better than the local agents.

3.2. **Training of the transfer learning-based model.**

3.2.1. *Pretraining in the source domain.*

During pretraining in the source domain, the agent learns single-step inference to select valid paths. In the generated training dataset, all triples (h, r, t) in the state space are decomposed into a pair of two-tuples (h, r) and (t, r^{-1}) to support forward path search and backward path search. If an agent selects a valid path at the state $s(t)$, a forward activation reward 1 is given to the agent; otherwise, no reward is provided.

3.2.2. *Pretraining in the source domain.*

In fine-tuned training, the shared parameters learnt by the agent in the source domain are transferred to the multi-step reasoning task in the target domain. Different from the single-step reasoning task, multi-step reasoning starts from the head entity e_{source} and searches the path towards the target entity e_{target} through selection of multiple paths. The reward function is set as follows:

$$P_M = \begin{cases} 1, \text{ifsuccese} \\ 0, \text{otherwise} \end{cases} . \tag{11}$$

If the path is the correct target path, the forward activiation reward is set at 1; otherwise, the reward is 0.

4. **Experiment and Analyses.**

4.1. **Datasets.**

In the present work, three public datasets used for knowledge inference, i.e., FB15K-237, NELL-995, and WN18RR, are used in experiments to verify the performance of the TMAPath model. The FB15K-237 dataset is a subset of the open-source link database Freebase; obtained by deletion of redundant relations from the FB15K dataset, the FB15K-237 dataset consists of 237 relations, 14,500 entities, and 310,000 fact triples. The NELL-995 dataset is developed by Carnegie Mellon University after 995 iterations of the structured comprehensive KGNELL, consisting of 200 relations, 7,500 entities, and

154,000 fact triples. The WN18RR dataset is a subset of the large lexical database of English WordNet, which is obtained after deletion of redundant relations from the WN18 dataset, consisting of 11 relations, 4,000 entities, and 9,300 fact triples. Table 1 shows the statistical specifications of these datasets.

TABLE 1. Statistical specifications of the three datasets

Dataset	Relations	Entities	Fact triples	Avg. degree	Median degree
FB15K-237	237	14505	310116	19.74	14
NELL-995	200	75492	154213	4.07	1
WN18RR	11	40493	93003	2.19	2

4.2. **Parameter settings.**

All experiments in the present work are performed in the same environment and on the same datasets to verify the performance of our model. The experiments were performed based on the multi-agent deep reinforcement learning framework, and on the Ubuntu18.04LTS operating system; the experiment software is Pycharm Community2021.1, the GPU is NVIDIA Tesla K80, the computer memory is 8 GB, and the Python model is Python 3.6.8. The training in the experiment was realized through transfer learning, and during pretraining in the source domain, the learning rate was set at 0.01, the embedding dimension of the KG was set at 128, and the batchsize was 64, and 10 epochs of training was performed. During fine-tuned training in the target domain, the learning rate was set at 0.001, the embedding dimension of the KG was 128, the batchsize was 64, and 200 epochs of training was performed.

In the present work, each of the three datasets was divided into two sets: a training set and a test set, and repetitive optimization experiments were performed to achieve the optimal result. In pretraining in the source domain, 80% of the dataset was used for training, and the rest 20% was used for testing; in the fine-tuned training test in the target domain, 70% of the dataset was used for training, and the reset 30% was used for testing.

4.3. **Baseline models.**

Our model was compared with three types of baseline methods, i.e., distributed KI models, path-based KI models, and RL-based KI models, to verify its effectiveness. To be specific, the distributed KI models include TransE, TransH, TransR, and TransD; the path-based KI models include the path-ranking algorithm (PRA); and the RL-based KI models include DeepPath_TransE, DeepPath_TransD, DeepPath_nop, DIVINE (DeepPath), AttnPath, and AttnPathForce.

TransE: a typical representation learning method which maps the entities and relations in the KG onto a vector space according to translation invariance to store the internal structural information of the KG.

TransH: to address the complicated many-to-one and many-to-many relations that TransE cannot process, TransH represents the different relations of the same entity by different representations.

TransR: it constructs entities and relations in the entity space and multiple relation spaces, and realizes conversion in the corresponding relation space.

TransD: in the TransD model, each entity and relation are represented by two vectors, among which one vector represents the semantic meaning of the entity or the relation, and the other mapping vector is used to construct a mapping matrix.

PRA: Random walk is used to perform limited search of specific paths in the KG and inquire the probability of correlations between two entities, which solves the problem that the types of the edges cannot be differentiated in other models.

DeepPath_TransE: it is a reinforcement learning model used for inference of multi-hop relations in paths, which has a policy-based reward function (based on such indicators as precision, diversity, and efficiency). As the primary DeepPath model, DeepPath_TransE has a state space initialized by TransE.

DeepPath_TransD: Different from the primary DeepPath_TransE, this model has a state space initialized by TransD.

DIVINE: it is a GAIL reinforcement learning model, which allows the existing reinforcement learning methods to self-learn the reasoning policies and reward functions.

AttnPath: LSTM and the graph attention mechanism are used as the memory component to alleviate the model's reliance on pretraining, and fine-tuning of the model is performed under the framework of reinforcement learning.

4.4. **Comparative Experiments.**

Three comparative experiments were performed and analyzed in the present work: path search experiments, fact prediction experiments, and link prediction experiments.

4.4.1. *Path search experiment.*

Comparative experiments for path search were performed in the present work to verify our model's performance in path learning. The TMAPath model was compared with other reinforcement learning models including the DeepPath model and the AttnPath model. The sum success rate of path search is taken as the evaluation indicator, which is defined as follows:

$$TSR = \frac{suc_{edge}}{sum_{edge}}, \tag{12}$$

where suc_{edge} represents the number of correctly predicted edges in the path search; sum_{edge} is the sum of edges in the path search; TSR is the total success rate of path search, and a larger TRS indicates a better performance of the model. Table 2 shows the results of the path search experiments.

Table 2. Total success rate of path search experiments (%)

Model	FB15K-237	NELL-995	WN18RR
DeepPath	15.301	30.09	27.89
DeepPathNoPre	6.103	22.73	24.462
AttnPath	17.899	30.11	28.935
AttnPathForce	30.330	48.089	29.340
TMAPath	47.126	48.008	45.00

In Table 2, DeepPathNoPre is a DeepPath variant with the pretraining process removed; AttnPathForce is an AttnPath variant with a forced walk mechanism introduced. The result of path search experiments is analyzed as follows.

(1) Pretraining has improved the model's path search performance. In the three datasets, the DeepPath model achieved a far higher TSR than the DeepPathNoPre variant in path search, which proved that learning the parameters and experience in the pretraining module could improve the agent's capacity in path inference.

(2) The forced walk mechanism has improved the agent's efficiency in selecting valid paths. AttnPathForce achieved a higher success rate in path search than the AttnPath model on all the three datasets, which indicates that introducing a special

walk mechanism could force the agent to avoid selection of invalid paths every step forward and prevent the agent from being blocked at a node.

(3) Compared with other reinforcement learning models, the TMAPath model proposed in the present work has considerably improved the success rate of path search, especially on the FB15K-237 and WN18RR datasets where it achieved a TSR 16.79% and 15.66% higher than the AttnPathForce model. The major cause is that during pretraining in the source domain, the method for detection of the fact triples is introduced, which increased the success rate of the agent to mine effective path by single-step walks and hence improved the capacity for multi-step inference in the target domain.

4.4.2. *Fact prediction experiment.*

As a downstream task of knowledge inference, fact prediction is to judge whether the fact is true or false. Specifically, fact prediction aims to predict whether the given triple (e_h, r, e_t) is correct, that is, to identify whether there is a relation r between the head entity and the tail entity. The fact prediction task does not rank the target entities, but ranks all the positive and negative samples of specific relations. In the experiments, three datasets FB15K-237, NELL-995, and WN18RR have already provided negative samples of triples, and the ratio of positive triples to negative triples is 1:10. Scores are given to the triples as per whether they are consistent with the valid paths, and they are ranked by the score in the test set; the mean average precision (MAP) is taken as the evaluation indicator, and a higher MPA indicates a higher probability that the triple is a positive sample.

In the present work, a new scoring mechanism based on transfer reinforcement learning is proposed, which gives scores by the judgement results of both the edge and the tail entity rather than by the mere judgement of the edge as other methods do. This new scoring mechanism that focuses merely on the judgement of the triples is efficient, as it allows the loss function to play a stronger role and make the model learn correct triples. Table 3 shows the result of fact prediction experiments, where the numbers like 1 and 10 mean the rounds of tests in the experiment.

As Figure 4 shows, the TMAPath model outperformed other baseline models on all the three datasets in fact prediction. As the rounds of tests increased, the performance of fact prediction improved, reaching the optimum after 50 rounds of tests, achieving a MAP of 0.467 on FB15K-237, 0.702 on NELL-995, and 0.609 on WN18RR. To conclude, the TMAPath model improved the robustness and precision of the reinforcement learning framework, which would enhance the performance of knowledge inference.

4.4.3. *Link prediction experiments.*

As an important task in knowledge inference, link prediction is to predict the target entity. In link prediction, a head entity-relation two-tuple (e_h, r) is provided to predict the tail entity e_t , and the candidate tail entities are ranked by the scores they are given. In the link prediction experiment, one two-tuple generates one fact entity and 10 false entities, the dataset is divided into a training set and a test set, and the inferred path is taken as a binary feature; a classifier is obtained by pretraining in the training set, and is then used to give scores to the tail entities in the test set. Hits@10 is used another indicator for performance assessment. If there is only one that hits the tail entity every ten actions, then Hits@10 is considered correct. Table 4 shows the result of link prediction experiments.

As Figure 5 and Figure 6 shows, our model achieved the best Hits@10 on the NELL-995 dataset among all models, and it also performed well on other datasets. In sum, our model performed better than other models in knowledge inference.

TABLE 3. Result of fact prediction experiments (measured by MAP)

	Fact prediction result (MAP)		
	FB15K-237	NELL-995	WN18RR
TransE	0.276	0.383	0.293
TransH	0.298	0.385	0.320
TransR	0.301	0.406	0.288
TransD	0.301	0.413	0.310
PRA	0.215	0.275	0.198
DIVINE (DeepPath)	0.338	0.493	0.361
DeepPath_TransE	0.310	0.493	0.373
DeepPath_TransD	0.313	0.533	0.414
DeepPath_nop	0.310	0.446	0.379
AttnPath	0.315	0599	0.451
AttnPathForce	0.381	0.692	0.512
TMAPath-1	0.356	0.478	0.385
TMAPath-10	0.391	0.512	0.468
TMAPath-20	0.408	0.524	0.594
TMAPath-50	0.467	0.702	0.609

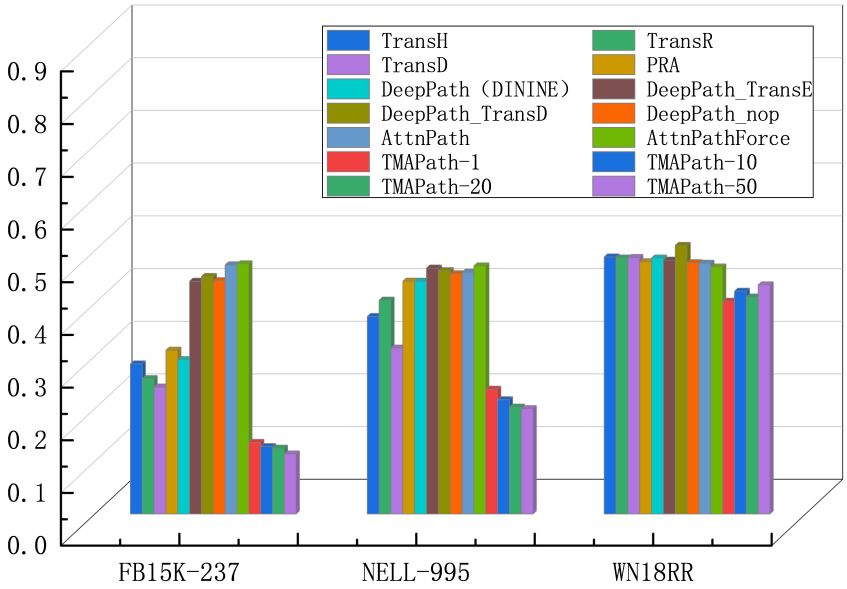

FIGURE 4. Result of fact prediction experiments (measured by MAP)

4.5. **Ablation experiment.**

Ablation experiments were performed to verify the effect of different modules of the model on the final result. Specifically, modules were removed from the model to retrain the model.

(1) Removing the pretraining module in the source domain

To explore the impact of pretraining in the source-domain on the model's performance, we performed reinforcement learning training directly in the target domain to generate a TMAPath-Target model, and path search experiments and single-step walk experiments were then performed on the trained model.

TABLE 4. Result of link prediction experiments

Models	Results					
	FB15K-237		NELL-995		WN18RR	
	MRR	Hits@10	MRR	Hits@10	MRR	Hits@10
TransE	0.240	0.471	0.371	0.671	0.488	0.892
TransH	0.285	0.585	0.375	0.713	0.488	0.867
TransR	0.257	0.655	0.406	0.772	0.486	0.917
TransD	0.241	0.742	0.315	0.801	0.487	0.925
PRA	0.311	0.792	0.442	0.844	0.479	0.911
DeepPath (DININE)	0.293	0.747	0.442	0.834	0.486	0.923
DeepPath_TransE	0.442	0.809	0.467	0.881	0.482	0.910
DeepPath_TransD	0.451	0.824	0.462	0.871	0.510	0.963
DeepPath_nop	0.443	0.811	0.456	0.860	0.477	0.901
AttnPath	0.473	0.865	0.459	0.873	0.476	0.899
AttnPathForce	0.475	0.878	0.471	0.895	0.469	0.891
TMAPath-1	0.136	0.794	0.237	0.833	**0.404**	0.871
TMAPath-10	0.128	0.839	0.217	0.867	0.423	0.893
TMAPath-20	0.125	0.844	0.203	0.899	0.412	0.922
TMAPath-50	**0.114**	**0.864**	**0.200**	**0.911**	0.435	**0.925**

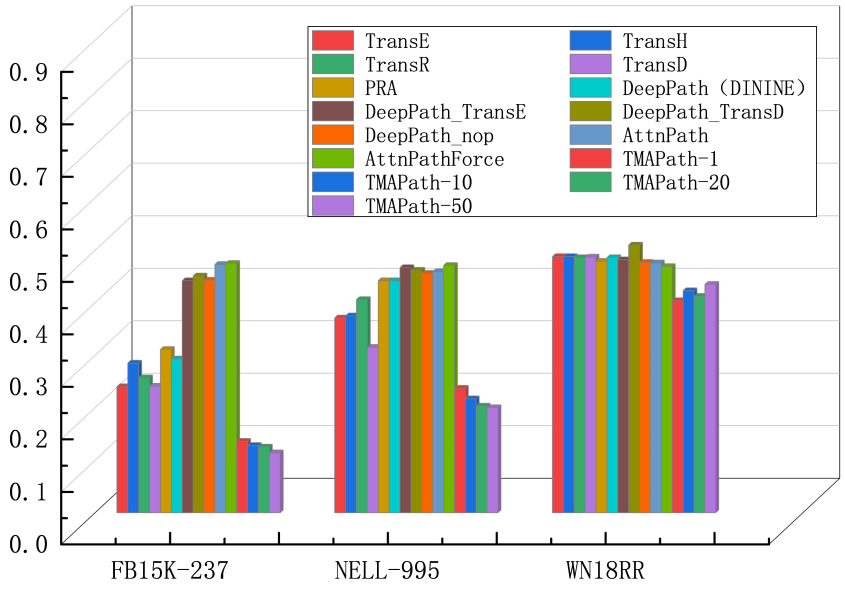

FIGURE 5. Result of link prediction experiments (measured by MRR)

(2) Removing the fine-tuned training module in the target domain

To explore the impact of the fine-tuned training in the target domain on the model's performance, we pretrained the model directly in the source domain to generate the TMAPath-Pre model, and then path search experiments and single-step walk experiments were performed on the model.

The two mutilated models (TMAPath-Target and TMPath-Pre) were compared with the original TMAPath model in path search and fact prediction experiments on the NELL-995 and FB15K-237 datasets. Table 5 shows the results of ablation experiments. In the

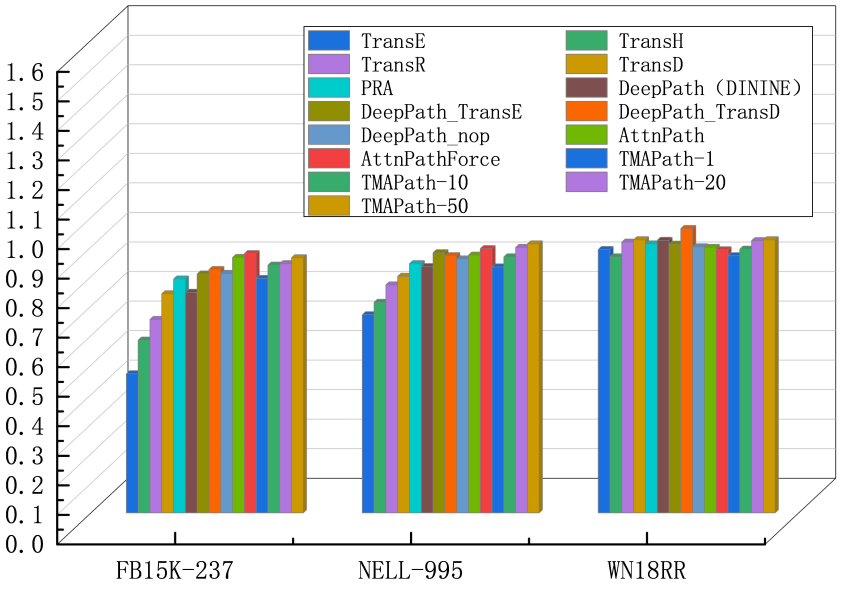

FIGURE 6. Result of link prediction experiments (measured by Hits@10)

path search experiment, the total success rate was taken as the evaluation indicator; in the fact prediction experiment, the round of tests was set at 1.

TABLE 5. Result of ablation experiments of transfer learning

Models	Source Task	Fine-tunning Task	NELL-995		FB15K-237	
			TSR	MAP	TSR	MAP
TMAPath-Pre	✓	✗	0.231	0.301	0.102	0.119
TMAPath-Target	✗	✓	0.310	0.371	0.253	0.213
TMAPath	✓	✓	0.480	0.478	0.471	0.356

As Table 5 shows, the TMAPath-Pre model demonstrated poor performance in all experiments. The major cause is that pretraining in the source domain only provides single-step inference training, but no training for multi-step inference, which leads the model to poor performance in KI. That is to say, models relying merely on single-step inference cannot have high efficiency in path search, and multi-step inference training is necessary to improve the efficiency. Compared with the TMAPath-Target model, the original TMAPath model achieved a TSR 17.0% and 21.8% higher on the two datasets in the path search experiment, and an mPA 10.7% and 14.3% higher on the two datasets in the fact prediction experiment. It indicates that the pretraining task for transfer learning could improve the multi-step inference capacity of the reinforcement learning architecture and hence enhance the model's knowledge inference abilities.

5. **Conclusions.**

In the present work, TMAPath, a novel knowledge inference method based on transfer learning and multi-agent deep reinforcement learning, is proposed, which converts the inference tasks in the source domain and the target domain into a sequential decision-making problem. First, a source task module that comprises of a triple pretraining network and reward mechanism is introduced to equip the agent with the capacity to learn single-step inference and adjust the model parameters; then, transfer learning is employed to train

the model to learn the shared parameters through fine-tuned training in the target domain; last, the proposed model is compared with other baseline models on three public datasets through experiments of path search, fact prediction, and link prediction, and ablation experiments of transfer learning are also performed. The experiment result proved the good performance of our model in knowledge inference.

REFERENCES

[1] A. B. M. Moniruzzaman, R. Nayak, M. Tang, and B. Thirunavukarasu, "Fine-grained type inference in knowledge graphs via probabilistic and tensor factorization methods," in *WWW'19: Proceedings of the World Wide Web Conference*. ACM, 2019, pp. 3093–3100. [Online]. Available: https://doi.org/10.1145/3308558.3313597

[2] H. Sedghi and A. Sabharwal, "Knowledge completion for generics using guided tensor factorization," *Transactions of the Association for Computational Linguistics*, vol. 6, pp. 197–210, 2018. [Online]. Available: https://doi.org/10.1162/tacl_a_00015

[3] F. Zhang, T.-Y. Wu, Y. Wang, R. Xiong, G. Ding, P. Mei, and L. Liu, "Application of quantum genetic optimization of lvq neural network in smart city traffic network prediction," *IEEE Access*, vol. 8, pp. 104 555–104 564, 2020. [Online]. Available: https://doi.org/10.1109/ACCESS.2020.2999608

[4] J. M.-T. Wu, Z. Li, N. Herencsar, B. Vo, and J. C.-W. Lin, "A graph-based cnn-lstm stock price prediction algorithm with leading indicators," *Multimedia Systems*, pp. 1–20, 2021. [Online]. Available: https://doi.org/10.1007/s00530-021-00758-w

[5] J. M. Wu, L. Sun, G. Srivastava, and J. C. Lin, "A long short-term memory network stock price prediction with leading indicators," *Big Data*, vol. 9, no. 5, pp. 343–357, 2021. [Online]. Available: https://doi.org/10.1089/big.2020.0391

[6] A. Bordes, N. Usunier, A. García-Durán, J. Weston, and O. Yakhnenko, "Translating embeddings for modeling multi-relational data," in *NIPS'13: Proceedings of the International Conference on Neural Information Processing Systems*, 2013, pp. 2787–2795. [Online]. Available: https://proceedings.neurips.cc/paper/2013/hash/1cecc7a77928ca8133fa24680a88d2f9-Abstract.html

[7] B. Nie and S. Sun, "Knowledge graph embedding via reasoning over entities, relations, and text," *Future Generation Computer Systems*, vol. 91, pp. 426–433, 2019. [Online]. Available: https://doi.org/10.1016/j.future.2018.09.040

[8] W. Zhang, B. Paudel, L. Wang, J. Chen, H. Zhu, W. Zhang, A. Bernstein, and H. Chen, "Iteratively learning embeddings and rules for knowledge graph reasoning," in *WWW'19: Proceedings of the World Wide Web Conference*. ACM, 2019, pp. 2366–2377. [Online]. Available: https://doi.org/10.1145/3308558.3313612

[9] N. Lao and W. W. Cohen, "Relational retrieval using a combination of path-constrained random walks," *Machine Learning*, vol. 81, no. 1, pp. 53–67, 2010. [Online]. Available: https://doi.org/10.1007/s10994-010-5205-8

[10] L. Cai, P. Luo, G. Zhou, T. Xu, and Z. Chen, "Multiperspective light field reconstruction method via transfer reinforcement learning," *Computational Intelligence and Neuroscience*, vol. 2020, pp. 8 989 752:1–8 989 752:14, 2020. [Online]. Available: https://doi.org/10.1155/2020/8989752

[11] W. Xiong, T. Hoang, and W. Y. Wang, "Deeppath: A reinforcement learning method for knowledge graph reasoning," in *EMNLP'17: Proceedings of the Conference on Empirical Methods in Natural Language Processing*, M. Palmer, R. Hwa, and S. Riedel, Eds. Association for Computational Linguistics, 2017, pp. 564–573. [Online]. Available: https://doi.org/10.18653/v1/d17-1060

[12] C. J. Watkins and P. Dayan, "Q-learning," *Machine Learning*, vol. 8, no. 3, pp. 279–292, 1992.

[13] V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, A. Graves, M. Riedmiller, A. K. Fidjeland, G. Ostrovski *et al.*, "Human-level control through deep reinforcement learning," *Nature*, vol. 518, no. 7540, pp. 529–533, 2015. [Online]. Available: https://doi.org/10.1038/nature14236

[14] N. Feng, T. Wu, and Y. Liang, "A deep dynamic neural network model and its application for ECG classification," *Journal of Intelligent & Fuzzy Systems*, vol. 43, no. 2, pp. 2147–2154, 2022. [Online]. Available: https://doi.org/10.3233/JIFS-219314

[15] K.-K. Tseng, R. Zhang, C.-M. Chen, and M. M. Hassan, "Dnetunet: a semi-supervised cnn of medical image segmentation for super-computing ai service," *The Journal of Supercomputing*, vol. 77, no. 4, pp. 3594–3615, 2021. [Online]. Available: https://doi.org/10.1007/s11227-020-03407-7

[16] S. Chen, M. Wang, W. Song, Y. Yang, Y. Li, and M. Fu, "Stabilization approaches for reinforcement learning-based end-to-end autonomous driving," *IEEE Transactions on Vehicular Technology*, vol. 69, no. 5, pp. 4740–4750, 2020. [Online]. Available: https://doi.org/10.1109/TVT.2020.2979493

[17] D. Ye, Z. Liu, M. Sun, B. Shi, P. Zhao, H. Wu, H. Yu, S. Yang, X. Wu, Q. Guo, Q. Chen, Y. Yin, H. Zhang, T. Shi, L. Wang, Q. Fu, W. Yang, and L. Huang, "Mastering complex control in MOBA games with deep reinforcement learning," in *AAAI'20: Proceedings of the AAAI Conference on Artificial Intelligence*, 2020, pp. 6672–6679. [Online]. Available: https://ojs.aaai.org/index.php/AAAI/article/view/6144

[18] F. Liu, R. Tang, X. Li, W. Zhang, Y. Ye, H. Chen, H. Guo, Y. Zhang, and X. He, "State representation modeling for deep reinforcement learning based recommendation," *Knowledge-Based Systems*, vol. 205, p. 106170, 2020. [Online]. Available: https://doi.org/10.1016/j.knosys.2020.106170

[19] K. Wang, S. P. Xu, C.-M. Chen, S. H. Islam, M. M. Hassan, C. Savaglio, P. Pace, and G. Aloi, "A trusted consensus scheme for collaborative learning in the edge ai computing domain," *IEEE Network*, vol. 35, no. 1, pp. 204–210, 2021. [Online]. Available: https://doi.org/10.1109/MNET.011.2000249

[20] K. Wang, C.-M. Chen, Z. Liang, M. M. Hassan, G. M. Sarne, L. Fotia, and G. Fortino, "A trusted consensus fusion scheme for decentralized collaborated learning in massive iot domain," *Information Fusion*, vol. 72, pp. 100–109, 2021. [Online]. Available: https://doi.org/10.1016/j.inffus.2021.02.011

[21] R. Das, S. Dhuliawala, M. Zaheer, L. Vilnis, I. Durugkar, A. Krishnamurthy, A. Smola, and A. McCallum, "Go for a walk and arrive at the answer: Reasoning over paths in knowledge bases using reinforcement learning," in *ICLR'18: Proceedings of the International Conference on Learning Representations*, 2018. [Online]. Available: https://openreview.net/forum?id=Syg-YfWCW

[22] Y. Shen, J. Chen, P. Huang, Y. Guo, and J. Gao, "M-walk: Learning to walk over graphs using monte carlo tree search," in *NIPS'18: Proceedings of the International Conference on Neural Information Processing Systems*, 2018, pp. 6787–6798. [Online]. Available: https://proceedings.neurips.cc/paper/2018/hash/c6f798b844366ccd65d99bc7f31e0e02-Abstract.html

[23] X. V. Lin, R. Socher, and C. Xiong, "Multi-hop knowledge graph reasoning with reward shaping," in *EMNLP'18: Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, E. Riloff, D. Chiang, J. Hockenmaier, and J. Tsujii, Eds. Association for Computational Linguistics, 2018, pp. 3243–3253. [Online]. Available: https://doi.org/10.18653/v1/d18-1362

[24] R. Li and X. Cheng, "Divine: a generative adversarial imitation learning framework for knowledge graph reasoning," in *EMNLP-IJCNLP'19: Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing*, 2019, pp. 2642–2651. [Online]. Available: https://doi.org/10.18653/v1/D19-1266

[25] C. Fu, T. Chen, M. Qu, W. Jin, and X. Ren, "Collaborative policy learning for open knowledge graph reasoning," in *MNLP-IJCNLP'19: Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing*, K. Inui, J. Jiang, V. Ng, and X. Wan, Eds. Association for Computational Linguistics, 2019, pp. 2672–2681. [Online]. Available: https://doi.org/10.18653/v1/D19-1269

[26] Q. Wang, Y. Hao, and J. Cao, "Adrl: An attention-based deep reinforcement learning framework for knowledge graph reasoning," *Knowledge-Based Systems*, vol. 197, p. 105910, 2020. [Online]. Available: https://doi.org/10.1016/j.knosys.2020.105910

[27] S. Li, H. Wang, R. Pan, and M. Mao, "Memorypath: A deep reinforcement learning framework for incorporating memory component into knowledge graph reasoning," *Neurocomputing*, vol. 419, pp. 273–286, 2021. [Online]. Available: https://doi.org/10.1016/j.neucom.2020.08.032

[28] P. Tiwari, H. Zhu, and H. M. Pandey, "Dapath: Distance-aware knowledge graph reasoning based on deep reinforcement learning," *Neural Networks*, vol. 135, pp. 1–12, 2021. [Online]. Available: https://doi.org/10.1016/j.neunet.2020.11.012