World Wide Web (2021) 24:1837–1856
https://doi.org/10.1007/s11280-021-00911-5

To hop or not, that is the question: Towards effective multi-hop reasoning over knowledge graphs

Jinzhi Liao[1,2] · **Xiang Zhao**[1,2] · **Jiuyang Tang**[1,2] · **Weixin Zeng**[1,2] · **Zhen Tan**[1,2]

Received: 21 October 2020 / Revised: 22 March 2021 / Accepted: 10 June 2021 /
Published online: 26 July 2021

Abstract
With the proliferation of large-scale knowledge graphs (KGs), multi-hop knowledge graph reasoning has been a capstone that enables machines to be able to handle intelligent tasks, especially where some explicit reasoning path is appreciated for decision making. To train a KG reasoner, supervised learning-based methods suffer from false-negative issues, i.e., unseen paths during training are not to be found in prediction; in contrast, reinforcement learning (RL)-based methods do not require labeled paths, and can explore to cover many appropriate reasoning paths. In this connection, efforts have been dedicated to investigating several RL formulations for multi-hop KG reasoning. Particularly, current RL-based methods generate rewards at the very end of the reasoning process, due to which short paths of hops less than a given threshold are likely to be overlooked, and the overall performance is impaired. To address the problem, we propose RL-MHR, a revised RL formulation of multi-hop KG reasoning that is characterized by two novel designs—the stop signal and the worth-trying signal. The stop signal instructs the agent of RL to stay at the entity after finding the answer, preventing from hopping further even if the threshold is not reached; meanwhile, the worth-trying signal encourages the agent to try to learn some partial patterns from the paths that fail to lead to the answer. To validate the design of our model RL-MHR, comprehensive experiments are carried out on three benchmark knowledge graphs, and the results and analysis suggest the superiority of RL-MHR over state-of-the-art methods.

Keywords Knowledge graph reasoning · Reinforcement learning · Reasoning path

1 Introduction

Recently, we have witnessed the rapid proliferation of large-scale knowledge graphs (KGs) (e.g., `DBPedia` [2], `YAGO` [33] and `Freebase` [20]). They serve as a well-organized store

This article belongs to the Topical Collection: *Special Issue on Explainability in the Web*
Guest Editors: Guandong Xu, Hongzhi Yin, Irwin King, and Lin Li

✉ Xiang Zhao
xiangzhao@nudt.edu.cn

Extended author information available on the last page of the article.

of factual knowledge that can be beneficial to a wide spectrum of downstream applications, including automatic question answering, knowledge-enhanced recommendation, etc.

There is a primitive task that is indispensable to a number of KG-oriented applications, which is termed as "multi-hop reasoning" on KGs. It can be abstracted as follows: given a *query triplet* (*head_entity, relation, ?*), to retrieve an answer *tail_entity* such that there is at least one explicit path from *head_entity* to *tail_entity* such that *head_entity, relation, tail_entity*) correctly states a fact.[1]

In particular, to perform multi-hop reasoning over KGs, one has to traverse the KG by *hopping* along with the relations, in order to relate the head entity to the tail entity. We use the following example to illustrate the task.

Example 1 Consider a partial KG of administrative divisions in Figure 1, and we would like to figure out the query

$$(\texttt{Asheville}, \texttt{Time\_zone}, ?),$$

where `Asheville` (resp. `Time_zone`) is referred as query entity (resp. relation). To reason against the query, we may necessitate a few triplets (e.g, (`Asheville, State, North Carolina`), (`North Carolina, Time_zone, Eastern Time Zone`)) to construct a reasoning path, and reach the answer (i.e., `Eastern Time Zone`).

To handle the task of multi-hop reasoning, an immediate solution is to leverage supervised learning. Studies in this branch pay more attention to the improvement of performance. They target to return an entity to answer the question, but ignore the reasoning process in the model. As the result lacks reasonable inference, it is hard for people to judge whether the selected entity is the golden ticket. The situation will impair the confidence of the model. On the top of that, supervised-learning based approaches rely on annotated paths to predict the answer (and path). Nevertheless, this can be prohibitively expensive, since there exist a large number of relations in KGs, and a reasoner may take an exponential number of paths before reaching the answer. When the annotated paths are scarce, i.e., only a small portion of the paths between head and tail entities, the trained models are unlikely to discover relations (and paths) that are less covered by training data. Subsequently, they tend to introduce *false negative* issues such that certain paths between head and tail entities are not to be found, and the correct answer is missed in prediction.

By approaching the problem from a different angle, the reasoning process recalls a class of sequence decision problems that can be solved by reinforcement learning (RL), which can retrieve explicit reasoning paths with answers and does not require supervision for each decision [40]. In order to teach an *agent* in RL about how to reason over KGs, current methods mainly take the *head_entity* as the starting node, then hop successively to some entities (each as an *action* in RL), and design some *reward* mechanism to evaluate every path reasoned by the agent up to some length T [8, 13, 21]. Particularly, the agent gets a positive reward if it arrives at the presumed *tail_entity*, or a negative reward (usually zero) otherwise. Among all the paths traversed by the agent, the highly ranked ones as per reward are expected to lead to the answer.

[1] Distinct from tasks like KG completion, this task requires the so-called path-based reasoning for deriving the answer, where a path is constituted of multiple relations in the KG (i.e., hops). The goal of the task is to improve the explainability of KG reasoning.

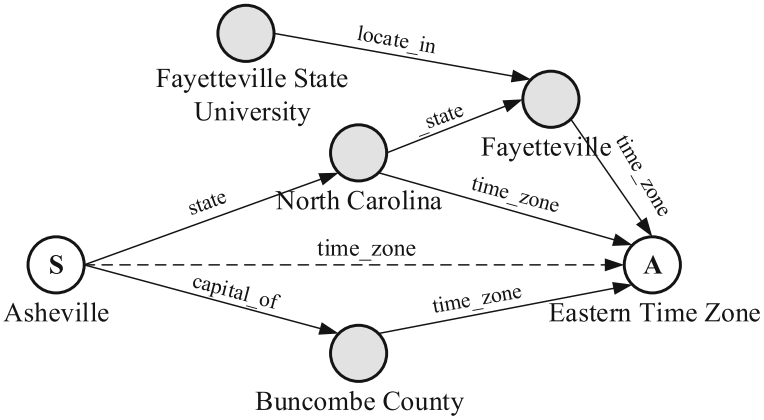

Figure 1 An incomplete KG for Example 1

Motivation It can be seen that such RL implementations necessitate a fixed number of hops T for the agent, and only after T hops are satisfied can the agent get some reward. Such formulation comes with not only the sparse reward issue, but also the implication that the agent should pursue reasoning paths of length T, in which case some correct but short paths may be discarded. As a remedy, an additional action of "NO_OP" (i.e., no option) is supplied to make short paths long enough [8, 21]. To some extent, this helps retain paths less than length T, unluckily, however, it does not teach the agent when to stop, or when to choose NO_OP. In other words, for each decision, NO_OP is considered equally as other actions, and thus, NO_OP could appear in any segment of a reasoning path, giving rise to no obvious pattern (cf. Section 4.6 for the detailed analysis). The situation also makes the inference in RL back to the supervised learning pursuing high performance, which further impairs the interpretability of results. In this light, we are motivated to teach the agent when to stop.

Moreover importantly, the aforementioned false negative issue is far from perfectly resolved. Due to the T-hop limit, paths longer than T will not be seen, paths short than T may be explored after injecting NO_OP's (or may be abandoned when NO_OP's are not perfectly incorporated), and only paths equal to length T are definitive to be discovered. Further, setting T to a practically large number does not help either. Out of expectation, however, according to our empirical results (cf. Section 4.7 for the detailed analysis), this alternative would downgrade the reasoning performance. Consequently, we are further motivated to revisit the formulation of RL for effective multi-hop reasoning.

Our approach In this research, we propose another RL formulation for multi-hop reasoning, which is featured by two novel designs to not only address the challenges above, but also improve the explainability of the model. Firstly, we design a stop signal to remind the agent of staying at the entity that is considered to be the answer. In this way, the agent may either traverse up to T hops without reaching an answer, or stop at a certain point and do not proceed further. To implement the idea, we incorporate a special action "STOP" to let the agent stay static at the entity where it is now [8], indicating that an answer has been predicted at the entity. Thus, STOP can only be chosen once for each round. As a result, we punish the agent if it chooses STOP before it is believed to have reached an answer, and reward the agent that chooses STOP right after reaching the answer. Note that we assess whether the answer is correct at the entity if the agent chooses STOP there, or at the last

entity of T hops otherwise. Moreover, to favor short paths when there are multiple paths leading to the answer, we introduce a punishing reward that discourages the agent to look for longer reasoning paths when there is already a shorter one (i.e., with a stop signal). Secondly, in order to mitigate the sparse reward issue, we put forward a worth-trying signal for those failed reasoning paths that do not lead to the correct answer. As connection patterns in KGs are rather complex, we argue that giving a negative reward whenever the agent does not hit the correct answer is brutal; and reasonable endeavors may be encouraged, as part of the reasoning processes could be appropriate. Nonetheless, such a pattern is too difficult to capture, and hence we resort to selecting an entity from the failed path, and taking it as the "fake ending" of the reasoning. Different from the normal reward assessment, we propose to employ a vector-space similarity as the worth-trying reward between the query and the "fake" answer, which is expected to capture their semantic relevance.

Contribution This paper makes the following contributions:

- We identify the weakness of current RL formulations for multi-hop KG reasoning, and to mitigate the issues, we propose a revised model, namely, RL-MHR;
- There are two novel designs—the stop signal that teaches the agent about when to early-stop after hitting an answer, and the worth-trying signal that exploits partial experiences from failed reasoning paths to amplify rewards; and
- We conduct comprehensive experiments to verify the designs, and RL-MHR is shown to outperform state-of-the-art RL based methods on three benchmarks (FB15k-237, WN18RR, and NELL-995).

Organization Section 2 summarizes related work. RL-MHR and its components are elaborated in Section 3. Section 4 reports experimental evaluation results and detailed analysis, followed by conclusion in Section 5.

2 Related work

In this section, we discuss related work on model explainability and existing methods for KG reasoning as well as KG completion.

RL-based methods The most relevant work is the efforts dedicated to RL-based formulations of multi-hop KG reasoning. MINERVA [8] is among the first to introduce RL framework into KG reasoning. It regards relations in the KG as candidate actions, and designs a policy network to teach agents how to choose among the actions. After the agent chooses a relation, it interacts with the environment. The environment returns a reward, which hence changes the state. These signals are finally used to optimize the policy network, and the agent hops to the next entity following the chosen relation, which initiates another decision. As the research is seminal but preliminary, it fails to take other features of KG reasoning into consideration, e.g., the semantic meaning of entities and relations.

Sparse reward issue commonly exists in RL, which also applies to KG reasoning. Lin et al. proposed a knowledge-based reward shaping [21]. They observed that in the formulation of [8], some reasoning paths are illogical; that is, although these paths lead to the answer, they are not relevant to the query relation but reach the target by accident. Hence, they employed semantic features of entities and relations through representation learning in a low-dimensional space. If the agent does not arrive at the answer, in the end, they utilized

the similarity between the last triple with the query relation as the reward, instead of a negative reward. Besides, to force the agent to focus on the semantic information from KG and enhance the generalization ability of the model, they randomly dropped actions in each hop of agents during the training process.

Monte-Carlo tree search is employed to build a RL framework [32]. It assumes that the decisions made by agents should be relevant, and applies GRUs to encode the decision history to achieve the information passing. To address the sparse reward problem, a value-function is designed based on the Monte-Carlo search tree to exploit more features of KGs.

Recent effort finds it impossible for the agent to reach the answer based on many reasoning paths in KG [13], because after some hops the current entities become irrelevant to the query relation. In this connection, they proposed a NO_ANSWER node to teach the model to filter irrelevant paths before reasoning starts. They used depth-first search to identify which path between the query and answer entities is correct to train a supervised model. Then, the model is used to select paths from the KG, and these paths are then employed as the input of RL. With the combination of supervised learning and RL, the model judges which path is meaningful during the reasoning. The study is not a pure RL-based method, and the performance highly depends on that of the supervised learning component.

It is noted that most of the studies fix the path length, and reward the agent only if the correct answer is hit at the last hop. To include paths less than the fixed length into consideration, NO_OP is used to work around. As it can show up during any stage of the reasoning process, the agent is not advised in essence when to stop reasoning.

Relation-based methods Relation-based methods mainly explore various possible paths and rank to choose the most appropriate one. Path Ranking Algorithm (PRA) [18] firstly introduces a random walk with a restart mechanism to search potential reasoning paths in the KG. All the observed paths are then used as features for a per-target-relation binary classifier to generate final results. Lao et al. further set the constrain that selected paths must end at one of the target entities in the training set and are within a maximum length [19]. However, these methods do not generalize well to a large number of distinct paths in large-scale KGs, since they regard each unique path as a singleton and fail to capture the commonalities among paths. To alleviate the problem, Gardner et al. introduced vector-space similarity heuristics in random work by incorporating textual content [11], which also relieves the feature sparsity issue in PRA. Chain-of-Reasoning utilizes a neural attention mechanism to enable multiple reasons to represent logical composition across all relations, entities and text [9]. The selected reasoning paths in these methods are generated by information from KG or text. Even though some methods might consider the information from the query when searching paths, and they still take the path as a basic unit, failing to exploit features of every single decision.

Rule-based methods This category of methods mainly exploits the symbolic nature of knowledge. Previous studies focus on inductive Logic Programming (ILP) [23, 24, 27] to exploit purpose predicate rules from KGs based on some patterns designed by human. Statistical relational learning methods [17] along with probabilistic logic [38] are trying to combine machine learning and logic. However, with manual templates and only symbolic information, these methods can only perform well in specific KGs and do not have the generalization abilities.

Afterward, there are some methods trying to combine logical rules learning with embeddings. Neural Theorem Provers (NTP) [31] employs vectors instead of symbols to learn logical rules through an end-to-end framework. They construct the framework on Prolog's

backward chaining inference method, and then rank all paths by calculating a success score for each path. Different from NTP, Neural LP [44] selects TensorLog [7] as the operators of the learning system. They design a LSTM based controller with a novel memory component [34], and the scores are computed through attention. Probabilistic Logic Neural Networks (pLogicNet) [26] proposes probabilistic logic neural networks to combine the strengths of first-order logic rules and representation learning. It defines the joint distribution of a collection of triplets with a Markov Logic Network, which associates each logic rule with a weight and can be effectively trained with the variational EM algorithm. Although differentiable memory allows end-to-end training, these methods have to cover the entire memory during learning, which is computationally expensive. RL-based methods apply a hard selection of relation edges to hop over the KG [8], which is computationally attractive and also improves the performance.

KG completion methods KG completion addresses a similar task to KG reasoning, where an explicit reasoning path is not offered. The basic idea of embedding-based approaches is that the knowledge in KGs can be represented in low-dimension space, which helps models to capture the semantic information of entities and relations. TransE [5] assumes that in the triple (e_s, r, e_o) their embeddings satisfy $\mathbf{E}_{e_s} + \mathbf{E}_r = \mathbf{E}_{e_o}$. Thus, Bordes et al. design an energy function to model this equation and optimize it via calculations on vectors. As TransE cannot address the "one-to-many" problem, DisMult [43] introduces a diagonal matrix to store the complex connections among entities. It can decrease the model complexity and avoid overfitting. But only symmetric relations could be represented in the model.

Lately, ComplEx [37] has proposed to resort to complex vector space. Based on the features of complex vector space, those asymmetrical relations can also be captured by the model. With the emerge of neural networks, ConvE [10] utilizes convolutional layers and full connect layers to achieve the interactions between $\mathbf{E}_{e_s}$ and $\mathbf{E}_t$, and then the information is applied to compute the similarity with $\mathbf{E}_{e_o}$. RotatE [35] proves that, in the complex vector space, r can be regarded as a rotation process between e_s and e_o. Thus, Sun et al. introduce new transition and loss functions to represent triples. Inspired by RotatE, QuatE [45] expands the two-dimension space to three-dimension space, and tackles the problems existing in the high-dimension space.

Among others, we are aware of relevant research that investigates reasoning over raw text document (e.g., that on WikiHop and MedHop [39]). While their input is similar to KG reasoning, they need to reason raw texts rather then KGs.

Explainability of machine learning models The goal of an explainable model is to make its behavior more understandable to humans by providing explanations about results. Generally, there are three categories about the model explainability: deep explanation, interpretable models and model induction [14]. The first branch tries to modify deep learning techniques to learn explainable features (e.g. axiomatic attribution from network [1]), and then employ them to perform downstream tasks. The second branch aims to exploit the model itself. Studies usually design some modules (e.g. expert knowledge [30], attention mechanism [42]) to leverage the original model to learn more structured, interpretable, causal models. Works from the last branch admit the difficulty of explaining a black box, and construct another explainable model (e.g. rules from LSTM [25], stochastic disturbance [28]) to infer the features from the target model. Our work can be classified into the second category, since we try to construct a more reasonable structure of RL to make promising reference in the multi-hop reasoning problem on KGs.

3 Methodology

In this section, we first formally define the task of multi-hop KG reasoning, and then introduce the outline of our proposed model.

3.1 Task definition

A KG is a directed graph consisting of an enormous amount of entities and their relations. We formally represent a KG as $\mathcal{G} = (\mathcal{E}, \mathcal{R})$, where $\mathcal{E}$ denotes the set of entities (nodes in the KG) and $\mathcal{R}$ denotes the set of relations (edges in the KG). Each fact in the KG can be represented in the form of a triplet $(e_s, r, e_o) \in \mathcal{G}$.

Given a query $(e_s, r_q, ?)$, multi-hop KG reasoning aims to output an entity e_o from $\mathcal{G}$, as well as explicit reasoning paths, so as to correctly answer the query. For instance, to answer *(Asheville, time_zone, ?)*, *(Asheville, state, North Carolina, time_zone, Eastern Time Zone)* can be regarded as a correct path. Since there are usually many reasoning paths with different lengths for each query, such as *(Asheville, capital_of, Buncombe County, time_zone, Eastern Time Zone)* and *(Asheville, state, North Carolina, _state, Fayetteville, time_zone, Eastern Time Zone)*, how to cover all potential reasoning paths without manual annotations becomes a challenging problem.

3.2 Proposed model

In multi-hop KG reasoning, the entity in the query is usually regarded as the initial node of each reasoning path. The reasoning process can be viewed as a sequential search over the KG to reach the answer entity based on the relations among entities. Generally in RL, the agent can interact with the environment, and then take an action to extend the current reasoning path. Next, the environment will update the state and provide a reward according to the action. The updated state and reward will then influence the agent in evaluating the current action and choosing the next action. The overall framework is shown in Figure 2, and, following previous studies [8], the main components can be illustrated as:

States At each timestep t, $S_t = (e_t, e_s, r_q) \in \mathcal{S}$ is a tuple where e_t is the current entity visited by the agent. e_s is the entity in the query and also the start point of the reasoning

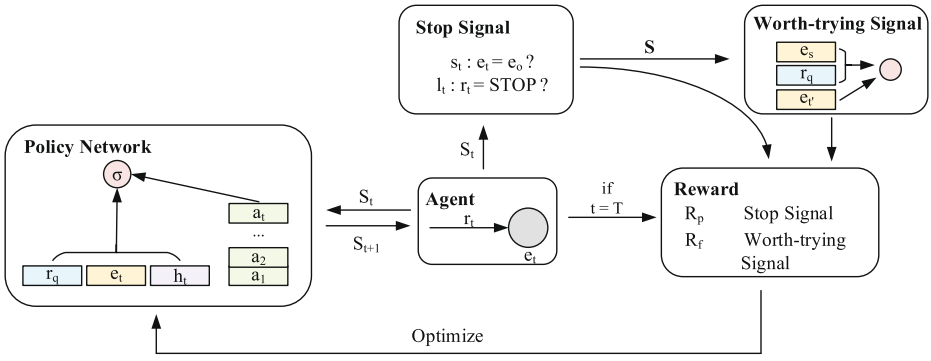

Figure 2 Framework about our method. See Section 3.2 for details about Policy Network and components in RL, Section 3.3 for details about Stop Signal, Section 3.4 for details about Worth-trying Signal, and Section 3.5 for details about Optimize

path. r_q is the query relation in the question. When the current state is put into the policy network for the next state, these entities and the relation will be represented in vector space. In other operations, they are stored as the triplet.

Actions At timestep t, the agent selects an action from the outgoing edges of e_t in $\mathcal{G}$. Formally, $A_t \in \mathcal{A}$ is made up of all the edges connected with e_t, i.e. $A_t = \{(r_{t+1}, e_{t+1})|(e_t, r_{t+1}, e_{t+1}) \in \mathcal{G}\}$. The A_t are candidates of hop directions for the agent when it arrives at the e_t. When it comes to computing the probability of each action, these entities and relations are represented as embeddings.

Policy network We parameterize the search policy using deep neural networks. The state information, query triplet and search history [8] are utilized as input, and the output is the probability distribution about candidate actions. The policy helps agents to determine which action is going to be selected at each timestep. In the policy network, all entities and relations in $\mathcal{G}$ are represented in vector embeddings $\mathbf{E}_e \in \mathbb{R}^d$, $\mathbf{E}_r \in \mathbb{R}^d$, where d is the dimension size of the embedding. At timestep t, the action $a_t \in A_t$ comprises of the entity and relation in the next time since the agent already takes the action. Formally, $\mathbf{a}_t = [\mathbf{E}_{r_{t+1}}; \mathbf{E}_{e_{t+1}}] \in \mathbb{R}^{2d}$ denotes the concatenation of the relation embedding and the entity embedding. The seach history h_t is applied to record previous paths before t. After that, we employ the LSTM [15] to encode the h_t and update its hidden state each timestep based on the selected action $\mathbf{a}_{t-1}$, as the following:

$$\mathbf{h}_t = LSTM(\mathbf{h}_{t-1}, \mathbf{a}_{t-1}). \tag{1}$$

With the history representation $\mathbf{h}_t$, the current entity representation $\mathbf{E}_{e_t}$ and the query relation representation $\mathbf{E}_{r_q}$, the policy π is defined as:

$$\pi_\theta(a_t|S_t) = softmax(\mathbf{A}_t f(\mathbf{h}_t, \mathbf{E}_{e_t}, \mathbf{E}_{r_q})), \tag{2}$$

where $softmax$ is the activation function, $\mathbf{A}_t \in \mathbb{R}^{|A_t| \times 2d}$ is the stack of all action embeddings in t, f is the feed forward network.

Transition The probability values of state transition are deterministic. Once the agent selects an action, the environment will respond. A transition function $\delta : \mathcal{S} \times \mathcal{A} \rightarrow \mathcal{S}$ is formally defined by $\delta(S_t, A_t) = \delta(e_t, e_s, r_q, A_t)$. Different from other tasks, where the next state is a probability distribution given the current state and action, the transition in KG reasoning totally relies on $\mathcal{G}$.

Rewards Rewards are the responses generated by the environment. In the traditional formulation, agents will receive a positive reward when they arrive at the answer entity. The goal of RL is to maximize the cumulative reward of each reasoning path. In conventional KG reasoning, a reasoning path will be rewarded only when the last hop reaches the e_o, and the agent can receive a positive terminal reward (usually 1). After that, each action in the path can be considered correct and also receives a positive reward. However, as mentioned above, this kind of reward mechanism fails to exploit potential paths.

3.3 The stop signal

In the traditional setting of RL-based methods, agents only receive the feedback at the end of reasoning. If the output is the correct answer, it would receive a positive reward. This, however, would lead to the fixed path length, where the majority of reasoning paths are abandoned and there are only a few positive instances for learning the policy network.

In KG reasoning, apart from the sparsity of reward, instances sometimes might be wrongly regarded as "negative". More specifically, since the length of each reasoning path is always fixed, the agent would have to reach the very end to generate the outcome. However, this might lead to the overlook of the situation where the answer entity appears in the middle of the path, e.g., some of the queries are very easy and can be answered in a small number of hops. Given the aforementioned settings, these short but correct reasoning paths would be ignored and the agent would receive a negative reward, which guides it to avoid these paths. We term this as a false-negative problem.

To address this issue, we design a signal vector $\mathbf{S} = (s_1, s_2, ..., s_n) \in \mathbb{R}^n$, where n denotes the length of the path, to record whether the agent arrives at the answer entity during the reasoning process. Concretely, we judge whether the agent reaches the answer entity at the end of each hop and mark the action with 0/1, as the following:

$$s_t = \begin{cases} 1 & e_t = e_o, \\ 0 & e_t ! = e_o, \end{cases} \tag{3}$$

where t denotes the timestep, e_t is the selected entity at t, e_o is the answer entity. Then, we regard the reasoning path with at least one value 1 in the signal vector $\mathbf{S}$ as the positive instance and give the agent a positive reward. In the policy learning strategy, there needs to be a fixed number of hops for each agent in every try. During the training process, agents may not quickly and accurately reach the answer. When they arrive at the answer earlier than the prescribed number of hops, unnecessary and wrong actions might be taken by the agents, resulting in the miss of the answer. In this connection, there should be at least one value 1 to encourage the agents to select STOP when the answer appears in the hops.

Since there is no labeled data about the length of the reasoning path in RL, we cannot train a supervised component to predict the path length. Consequently, a viable solution is to let the agent learn to repeatedly hit the current entity once it reaches the answer. In this light, we add a self-loop action "STOP" to every entity to encourage the agent to hop at the same place. Besides, for each correct reasoning path, all actions will receive the positive reward to encourage agents to follow the reasoning patterns in KG reasoning.

As the STOP action is an external relation and does not interact with other entities in the KG, it should not show up before the agent arrives at the answer entity. To this end, we design a vector $\mathbf{L} = (l_1, l_2, ..., l_n) \in \mathbb{R}^n$ to record the appearance of STOP, where 1 represents the agent select STOP and 0 represents it does not (similar to (3)). Besides, we propose a special path reward for the reasoning paths containing the STOP. Assuming the last STOP appears at timestep t', $R_p(S_t)$ is initialized with the positive reward r_p and then can be modified as the following:

$$R_p(S_t) = \begin{cases} r_e & s_t = 1,\ l_t = 1, \\ r_n & s_t = 0,\ l_t = 1, \\ r_n & t > t', \end{cases} \tag{4}$$

where t denotes the timestep, S_t is the state of the agent at t, r_e denotes the encourage reward, r_n denotes the negative reward. The path reward is then passed to the policy network as a part of the cumulative reward.

Due to the complex connections between entities and relations in KGs, there might be more than one correct reasoning path for the same query. When the agent does not loop in the shorter path, the hops after the answer get negative rewards, which makes the longer path cumulate more positive rewards. This impairs the process of the agent to exploit short paths. Therefore, we add a punitive reward r_u for each hop in the path before the agent finds

the answer, mathematically:

$$r_u(S_t) = \begin{cases} 0 & U = 1, \\ -\gamma & U = 0, \end{cases} \tag{5}$$

where γ is a hyper-parameter to balance the rewards of paths with different length. $U \in [0, 1]$ is a numerical variable, and is initialized with 0. When timestep t' makes $s'_t = 1$, U is changed to 1.

3.4 The worth-trying signal reward

As mentioned above, the sparse reward problem also makes it hard for the model to converge. For those questions where the facts are not in $\mathcal{G}$, the agent has to hop over a large search space and only gets the feedback when it reaches the answer. This might slow down the learning process. On the top of that, due to the complex connections in KGs, some reasoning paths might be very close to reaching the answer. Further to Example 1, The agent might first select the correct reasoning path $(Asheville, state, NorthCarolina, \_state, Fayetteville)$, but then hops to the wrong answer entity *"Fayetteville State University"*. Although the overall reasoning path, $(Asheville, state, NorthCarolina, \_state, Fayetteville, \_locate\_in, FayettevilleState\ University)$, is erroneous, the agent should get some rewards for the correct hops in the beginning. This in turn would encourage the agent to learn more useful patterns of the inference, and also alleviate the sparse reward problem.

Therefore, a worth-trying signal is introduced for those reasoning paths without the stop signal. Specifically, we select an entity $e_{t''}$ from the reasoning path and regard it as the answer. Since the signal is expected to guide the agent when it faces the dilemma, such as hopping to *(time_zone, Eastern Time Zone)* or *(_locate_in, Fayetteville State University)*, we design the reward based on the semantic relevance among entities and relations. Inspired by [21], we employ the pre-trained model to calculate the relevance score between the query (e_s, r_q) and $e_{t''}$, formally:

$$R_f(S_t) = F(\mathbf{E}_{e_s}, \mathbf{E}_{r_q}, \mathbf{E}_{e_{t''}}), \tag{6}$$

where F is a score function from the pre-trained models, the representations $\mathbf{E}$ of entities and relations are also pre-trained embeddings. The relevance score is applied as the reward for the current negative reasoning path and then transferred to compute the cumulative reward for negative paths.

With these proposed rewards, we define the reward function as:

$$R'(S_t) = \mathbb{1}_s * R_p(S_t) + (1 - \mathbb{1}_s) * R_f(S_t), \tag{7}$$

where $\mathbb{1}_s$denotes whether the current reasoning path contains STOP, and 1 denotes there is. This is then employed in the objective function to optimize the policy network.

3.5 Training

To teach agents to exploit more correct reasoning paths, we use the REINFORCE algorithm [40] to optimize the policy network π by maximizing the expected reward:

$$J(\theta) = \mathbb{E}_{(e_s, r_q, e_o) \in \mathcal{G}} \mathbb{E}_{a_1, \ldots, a_T \sim \pi_\theta} [R'(S_T | e_s, r_q)], \tag{8}$$

where T is the target timestep, and θ is the parameters needing to be trained in the policy network. The first expectation is calculated through empirical average over the data distribution in KGs. For the second expectation, we apply the current policy to roll out multiple

trajectories for each query to estimate a stochastic gradient, and updates the policy through stochastic gradient ascent.

4 Experiments

This section reports the experiments with in-depth analysis.

4.1 Experiment setting

To evaluate the performance of RL-MHR, we compare our method with other baselines in different datasets. We first introduce the benchmarks used in the experiment.

Datasets We employ three popular benchmark datasets[2] in multi-hop KG reasoning and follow the datasets split in [8]. The statistics of datasets are shown in Table 1.

- **FB15K-237** is constructed by Toutanova et al. [36]. There are many simply reverse relations in FB15K [5], which makes the task less challenging and realistic. Thus, they exclude redundant relations and direct training links for held-out triplets.
- **WN18RR** is constructed by Dettmers et al. [4]. They find that WN18 [5] also suffers from the problem of inverse relations, whereby a simple reversal rule-based model can achieve good performance. Thus, they remove sources appearing in both training data and testing data to address the test leakage problem.
- **NELL-995** is constructed by Xiong et al. [41]. It comes from the 995th iteration of NELL system [6] with some processing. Xiong et al. separate the KG for each query relation to build various sub-graphs, where every graph for a relation can have triplets from the test set of another query relation.

Baselines We compare RL-MHR with all existing RL-based methods. We discuss more details about these methods in Section 2.

- **MINERVA** is proposed by Das et al. [8]. They firstly convert KG reasoning into RL based on the Monte-Carlo algorithm. It judges correct reasoning paths via final prediction.
- **Multi-Hop Model** is proposed by Lin et al. [21]. They utilize the similarity score within a triplet to replace the negative reward. Then they equip it with an action drop part to improve the generalization ability of RL.
- **No-Answer Model** is proposed by Godin et al. [13]. They add a NO_ANSWER node to help the model filter the paths irrelevant to the answer. To improve the performance, a supervised component is then designed to label those paths.

4.2 Implementation details

We employ the pre-trained KG embeddings from ConvE [10] and ComplEX [36] to represent the embeddings of entities and relations and set the embedding size to 200. Inspired by [8], an entropy regularization is added to the objective and the weight parameter β is tuned within [0, 0.1]. Regarding the LSTM used to encode the action history, we set the

[2] https://github.com/shehzaadzd/MINERVA/tree/master/datasets

Table 1 Statistics of different KGs used in the experiment

Dataset	#Entities	#Relations	#Triplets	#Degree	
				Average	Median
WN18RR	40,945	11	86,835	2.19	2
NELL-995	75,492	200	154,213	4.07	1
FB15K-237	14,505	237	272,115	19.74	14

number of layers to 3, and the hidden size of it is also 200. Xavier [12] is utilized as initialization for layers in the neural networks. We adopt dropout in the KG embeddings and feed-forward layers of all models and search the rates within [0, 0.5]. For stop signal, the encourage reward r_e is tuned within (1, 2] and the punish reward r_u is tuned within (0, 0.5]. For the REINFORCE algorithm, the discount factor η is tuned within [0.9, 1]. We apply the training strategy used in [21] and tune the dropout rate within [0, 1]. ADAM [16] optimizer is employed to optimize training parameters and search the learning rate within [0.001, 0.003]. In the experiment, we set the last entity from each negative reasoning path to compute the worth-trying reward. We used an identical representation for all STOPs. Specifically, in the action space, every relation connects to a specific (tail) entity. After an action (i.e., relation) is selected by the agent, the connecting entity is also determined. For STOP, which can be deemed as a special relation, its connecting entity is the same as the one in the last hop (i.e., the head entity in this hop). In testing, the entity before the first appearance of the STOP action is taken as the predicted answer, as now the agents are expected to be able to determine when to stop after training.[3]

4.3 Results on RL-based methods

As can be observed from Table 2, similar to the distributions of results from ConvE and ComplEx in Section 4.5, RL-MHR (ConvE) beats RL-MHR (ComplEx) in FB15K-237 and NELL-995, and gets comparable results in WN18RR. This indicates the effects of pre-trained KG embeddings and also demonstrates that RL-MHR works seamlessly with embedding-based methods. For clarity, we use the results from RL-MHR (ConvE) for in-depth analysis in the sequel and omit the "(ConvE)". Besides, RL-MHR achieves superior results than other methods on all metrics across all datasets. This can be attributed to the fact that these baselines do not learn when to stop during the reasoning process. As they set a fixed length of the reasoning path, the agent tries to meet the path length and ignores those short but correct paths. Therefore, they have to tune the path length in accordance with the data distribution in different datasets. In contrast, the agent in RL-MHR understands when to stop the inference and hits the target repeatedly until reaching the path length. In this connection, RL-MHR can cover more correct reasoning paths regardless of the length. We further elaborate on the analysis of path length in Section 4.7.

There are roughly two classes of relations—*one-to-many* and *many-to-one*. The former denotes that the majority of the queries containing the relation r_q have multiple answer entities, while the latter represents that the majority of the queries containing the relation

[3]Only results in FB15K-237 are available in No-Answer Model. The source codes are not available. We tried our best to re-implement the model but the results are much worse than those reported in the paper. Hence, we omit the results for a fair comparison

Table 2 Overall performance compared with all RL-based methods

Method	FB15K-237			WN18RR			NELL-995		
	Hits@1	Hits@10	MRR	Hits@1	Hits@10	MRR	Hits@1	Hits@10	MRR
MINERVA	21.7	45.6	29.3	41.3	51.3	44.8	66.3	83.1	72.5
Multi-Hop Model	32.2	56.0	40.3	43.7	54.2	47.2	65.6	84.4	72.7
No-Answer Model	27.4	44.1	32.9	–	–	–	–	–	–
RL-MHR (ComplEx)	32.4	54.6	39.6	43.2	**56.7**	48.2	66.5	83.8	73.1
RL-MHR (ConvE)	**33.3**	**57.2**	**41.4**	**44.2**	55.5	**48.8**	**67.9**	**85.8**	**74.2**

The Hits@1, Hits@10 and MRR are represented in percentages. As Multi-Hop Model performs differently with different pre-trained embeddings, we show the best results in the table. "-" denotes the results are not available. The best results in each dataset are represented in bold. See Section 4.3 about details for result analysis

r_q merely have one correct answer. In NELL-995, most of the queries are many-to-one (87.1%). In contrast, WN18RR is a relatively balanced dataset, where the percentage of one-to-many queries is slightly higher (5.6%) than that of many-to-one queries [21]. Since MINERVA considers the entity in the last hop as the output, the agent tends to ignore the short but correct paths, which leads to the overlook of many useful patterns. Without any other useful information, it fails to address both one-to-many problem. In Multi-Hop Model, although the reward shaping component and action dropout strategy provide the agent with more semantic information, the fixed path length still impairs its ability to handle the many-to-one problem. Many potential patterns are still ignored. In No-Answer Model, it devises a supervised component to detect the reasoning path that cannot help the agent to reach the answer with the fixed path length. This mitigates the issue to an extent, but the results are largely dependent on the annotated path data. Different from these methods, RL-MHR overcomes the problems by designing the stop signal and worth-trying signal to help the agent to learn when to stop by itself and the results verify its effectiveness.

We notice that the gaps among all methods are relatively large on FB15K-237, and RL-MHR obtains the best results. The reason might be that there are many (76.6%) one-to-many queries in FB15K-237 [21], which requires the agent to take into account the semantic information of triplets. Since there are many candidates for each query, it is easy for the agent to make a false judgment merely based on the structured information. Besides, many wrong reasoning paths containing correct patterns. If these paths are abandoned directly, it will increase the difficulty for the agent to locate the final answer in these cases. As mentioned above, MINERVA does not take any semantic information into account, which leads to the worst performance on FB15K-237. No-Answer Model also ignores the semantic information, but the supervised component helps it to filter out some irrelevant paths and improve the result. Multi-Hop Model and RL-MHR consider the features of relations and entities in the vector space, which can capture the semantic relevance of triplets in the KG. Nevertheless, Multi-Hop Model fails to exploit many short but correct reasoning paths. Adding the stop signal component, our proposal outperforms all other methods.

4.4 Ablation study

In this section, an ablation study is performed to evaluate the significance of different components. We remove the stop signal component and worth-trying signal component,

respectively, and compare the performance of the rest frameworks with the original one on the very same datasets. It can be observed from the results in Table 3 that, both two components contribute to the performance of RL-MHR, except on WN18RR, where removing the worth-trying signal slightly improves the result. The detailed clarification is in the next paragraph.

Notably, the performance drop of removing the signal component differs across different datasets. This might be ascribed to the different data distributions. The main query class in FB15K-237 is one-to-many, while in NELL-995, the majority is many-to-one. As discussed before, one-to-many queries require more semantic information than structure information to instruct the agent to select the correct answer. Many-to-one queries, on the other hand, depend more on the structure of KGs, and the relations in queries can already make the agent accurately find the answer. Therefore, the semantic relevance introduced by the worth-trying signal can largely improve the performance on FB15K-237, while it merely brings a minor increase of the result on NELL-995. When it comes to WN18RR, the distribution of query classes is relatively balanced, which theoretically should rely on both semantic and structure features. However, there are only 11 relations in the dataset, which increases the difficulty for the agent to handle the one-to-many problem. As all the relations in queries are similar, the representation of the same relation might have semantic overlap with a huge number of entities. This impairs the effectiveness of the worth-trying signal component.

4.5 Results of other methods

In this section, we compare RL-MHR with representative rule-based methods and embedding-based methods. As the main goal of relation-based methods is to address the relation classification, we do not list them in the experiment. The results can be found in Table 4, and RL-MHR performs comparably to methods from other categories.

We observe that embedding-based approaches dominate in FB15K-237 (i.e., the best performance in terms of all metrics) and NELL-995 (i.e., the best performance in terms of two out of the three metrics). The reason might be that they represent every link from KG into the same high dimensional space, which could implicitly encode the connectivity of the whole graph. However, RL-MHR applies the discrete representations, which leaves out a significant proportion of the combinatorial path space by selection.

However, as shown in Table 1, there are only 11 relations in WN18RR, which decreases the importance of relations. Thus, embedding-based methods achieve low performance in the dataset. Rule-based methods are symbolic, which merely utilize the structure information of triplets. For each query, there are many reasoning paths matching the logic rules, leading to a relatively large number of candidates returned by the models. Therefore, the Hits@1 score of these methods is low, while the Hits@10 score is the best among all

Table 3 Comparison of results Hits@1 of RL-MHR and models without stop signal (SS) and worth-trying signal (WT)

Model	FB15K-237	WN18RR	NELL-995
RL-MHR	**33.3**	43.6	**67.9**
w/o SS	29.4 (-11.7%)	41.1 (-5.8%)	63.7 (-6.2%)
w/o WT	29.2 (-12.3%)	**44.2** (+1.3%)	66.8 (-1.6%)

% is computed based on the proposed model. See Section 4.4 about details for ablation analysis

Boldface indicates the best results

Table 4 Performance with other methods in different datasets

Method	FB15K-237			WN18RR			NELL-995		
	Hits@1	Hits@10	MRR	Hits@1	Hits@10	MRR	Hits@1	Hits@10	MRR
MLN [29]	6.7	16	9.8	19.1	36.1	25.9	–	–	–
NeuralLP [44]	16.6	34.8	22.7	37.6	**65.7**	46.3	–	–	–
pLogicNet [26]	–	–	–	39.8	53.7	44.1	–	–	–
AnyBURL [22]	26.9	52.0	–	42.9	53.7	–	–	–	–
ComplEx [37]	32.8	61.6	42.5	41.8	51.0	43.7	64.3	86.0	72.6
ConvE [10]	**34.1**	**62.2**	**43.5**	40.3	54.0	44.9	67.2	**86.4**	**74.7**
HypER [3]	–	–	–	43.6	52.2	46.5	–	–	–
DistMult [43]	32.4	60.0	41.7	43.1	52.4	46.2	55.2	78.3	64.1
RL-MHR	33.3	57.2	41.4	**44.2**	55.5	**48.8**	**67.9**	85.8	74.2

The Hits@1, Hits@10 and MRR are multiplied by 100. "-" denotes the results are not available. Boldface indicates the best results, while the second-best performance is underlined. See Section 4.5 about details for other methods analysis

approaches. Different from rule-based methods, embedding-based methods try to exploit the semantic features of KG. They take the semantic relevance of triplets into consideration, and use a low-dimensional vector space to exploit latent patterns of entities and relations. Therefore, they can answer the queries without providing the reasoning paths. Our RL-based model guides the agent to take actions based on both structure and semantic information, which can mitigate the impact of the noise induced by rules and make sequential decisions.

4.6 Case study

In this section, we conduct a case study to compare the reasoning paths selected by different models. To make the comparison clearer, we employ the complete reasoning paths from RL-MHR, while the entity appearing before the first STOP is used for evaluation.

As Figure 3 shows, both MINERVA and RL-MHR can correctly answer the *"Profession"* of *"Guy Ritchie"* in different path length. Other methods introduce a self-loop action

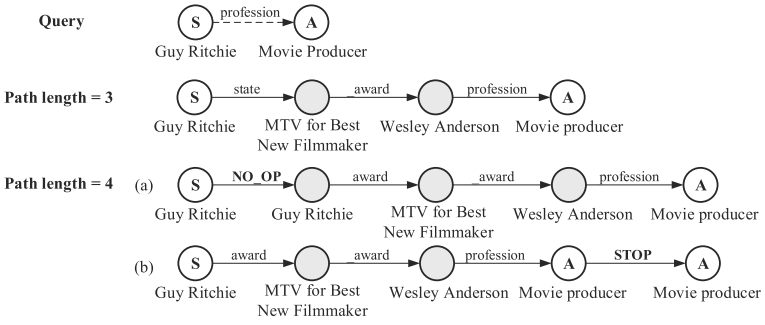

Figure 3 Sketch about the query correctly answered in different path length in FB15K-237. Path (a) is from MINERVA, and path (b) is from RL-MHR. "_" in the relation denotes reverse direction. Note that the relation "Profession" between "Guy Ritchie" and "Movie producer" is not in $\mathcal{G}$. See Section 4.6 for details

NO_OP. Although the agent reaches the correct answer, in the end, MINERVA only takes the action to meet the requirement about the number of hops. In path (a), the first selected action is NO_OP, which only helps to expand the length of the path. In this way, MINERVA can technically convert the long path into a short one. In RL-MHR, the agent firstly performs similar to the short path and stays at the same entity once it reaches *"Movie producer"*. This represents that the agent regards STOP as a signal and only takes it when meeting the possible answer.

We present several representative negative instances in Figure 4. MINERVA and Multi-Hop Model treat NO_OP as a usual action candidate for the agent, and all actions in the reasoning path are given a positive reward, once the agent arrives at the answer in the end. As NO_OP can be selected in each hop without any constraints, it might be hard for the agent to judge whether and when to choose NO_OP. This makes the agent randomly select NO_OP and leads to the situation as the path (a). The double hits about *"Winston-Salem"* contribute nothing to the reasoning path. In contrast, the increase of the path length leads to the higher difficulty of the reasoning and makes the agent fail to answer the relatively easy queries. Besides, without the stop signal, the agent might even miss the answer (path(b)). The target entity *"Eastern Time Zone"* appears in the middle of the path, but the agent passes it and hops to another entity to meet the length of the path. The NO_OP action does not work in the situation, since there is no reason for the agent to determine whether to stay. Therefore, in other RL-based methods, the agent tends to ignore the short but correct reasoning paths. In addition, without worth-trying rewards, the reward space comprises a large number of '0's and few '1's. This leads to the situation where the agent can hardly acquire positive feedback at each round, and further fails to learn correct patterns of the inference. Thus, it tends to reach the wrong entity (e.g., "*Fayetteville*" in the path (d)) and STOP there. The observation also proves the superiority of worth-trying rewards.

Different from them, in RL-MHR, the agent can detect the answer *"Eastern Time Zone* during the reasoning. Then it selects the STOP to repeatedly hit the answer entity to gain

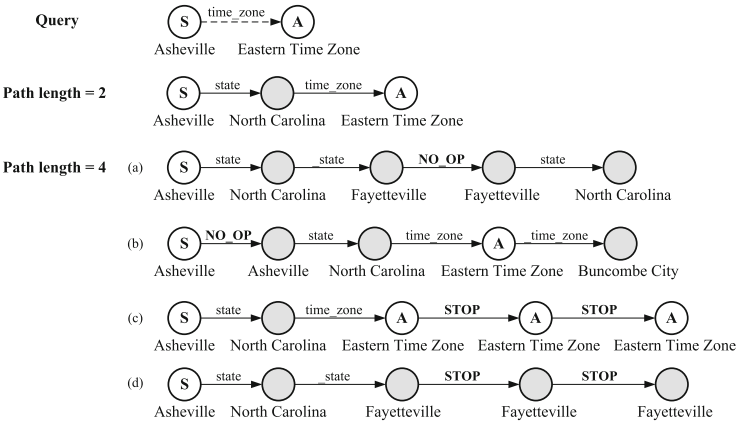

Figure 4 Sketch about the correctness of the output changes with the path length in FB15K-237. Path (a) is from MINERVA, path (b) from Multi-Hop Model and path (c) is from RL-MHR. "_" in the relation denotes reverse direction. Note that the relation "Time_zone" between "Asheville" and "Eastern Time Zone" is not in $\mathcal{G}$. See Section 4.6 for details

the encourage reward r_e. In other words, the long length provides the agent with the opportunity to cumulate more rewards, which helps RL to exploit potential patterns of the correct reasoning paths.

4.7 Further experiment

In this section, we compare our model with some baselines in different path lengths, and examine the change of their performance. As shown in Figure 5, with the length increasing, the performance of other methods decreases while that of RL-MHR rises and then stabilizes. Since the baseline methods all apply fixed path length to train the model, they can perform relatively better, especially, when most queries can be answered within a certain number of hops. With the length increasing, queries that need more hops to be solved can be addressed by these baselines. Since the path arriving at the answer at the last hop is regarded as the positive instance, many short but correct paths will be ignored. Consequently, they fail to answer some queries which can be resolved in a shorter path length, and their performance also decreases.

The reasons for the slight decrease of the performance of RL-MHR can be attributed to two parts, and both of them lead to the case where the model is hard to converge. The first one is the longer path increases the search space of the agent. A large amount of the possible reasoning path enhances the difficulty to detect the correct paths, and impair the learning process of the special action STOP. Since the longer path means more possible combinations of decisions, the probability of the positive reasoning path decreases. The agent gets fewer positive instances about repeatedly hitting the answer, which makes the system fail to acquire the ability to judge when to stop. The second one is the sparse reward problem. Although the agent might find the answer based on the stop signal, the requirement of more hops influences the cumulative reward. During the learning process, after the agent reaches the answer, it might not select STOP in the rest of the hops. The punish and negative rewards for the rest are then added to the cumulative reward. When the path length is relatively short, the final reward of this correct reasoning path is positive. As the length becomes longer, the times of making wrong decisions of the agent also increase. The final reward might turn to a negative one. The situation makes it hard for the agent to learn positive patterns.

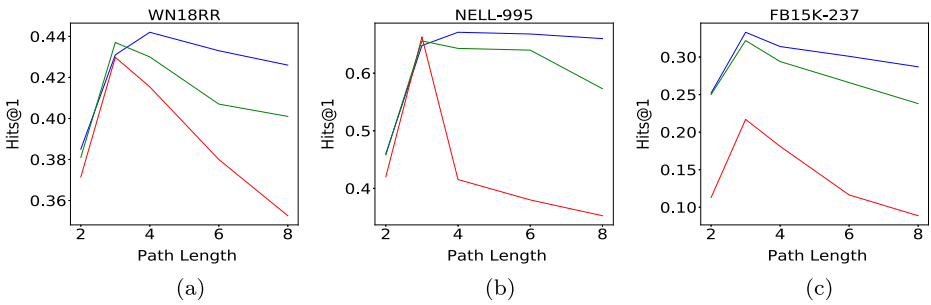

Figure 5 Performance of different RL-based methods with the growth of path. Blue curve denotes RL-MHR, green curve denotes Multi-Hop Model, and red curve denotes MINERVA. For the clear clarification, we apply Hits@1 as metric and select the model with the best performance of Multi-Hop Model. See Section 4.7 for details

5 Conclusion

In this paper, we propose a RL-based multi-hop KG reasoner, namely RL-MHR, which aims to make up the deficiency of existing methods. To address the false-negative and sparse reward problems to improve the explainability of the model, we introduce two components: stop signal and worth-trying signals. For those paths containing the answer, RL-MHR can judge when to stop and stay at the entity. For those negative paths, worth-trying rewards are applied to guarantee that RL-MHR can get semantic relevance from them. The outcome proves the efficiency of our model, and detailed analysis about different parts and the change of the path length show more advantages of RL-MHR.

In future work, we aim to research the following directions: (1) the semantic meaning of reasoning paths. In the experiment, we observe that RL-MHR sometimes retrieves a large number of paths meeting the requirement of the tail entity and target relation. However, in some cases, paths are not reasonable and readable. The reason might be that recent RL methods still fail to capture the semantic information containing by reasoning paths. Thus, we will explore how to model semantic information into RL to alleviate the problem. (2) the application of fancy frameworks in RL. With the development of RL, there are many new frameworks trying to handle the weakness of RL (e.g low usage of samples and static input). These problems also appear in KG reasoning and impair the performance of reasoning. It might be interesting to transfer these ideas into multi-hop KG reasoning.

Acknowledgments This work was partially supported by NSFC under grants Nos. 61872446, 61902417 and 71971212, NSF of Hunan Province under grant No. 2019JJ20024.

References

1. Arras, L., Montavon, G., Müller, K., Samek, W.: Explaining recurrent neural network predictions in sentiment analysis. In: WASSA@EMNLP 2017, September 8 2017, pp. 159–168, Copenhagen, Denmark (2017)
2. Auer, S., Bizer, C., Kobilarov, G., Lehmann, J., Cyganiak, R., Ives, Z.G.: Dbpedia: A nucleus for a web of open data. In: ISWC 2007 + ASWC 2007, November 11-15, 2007, pp. 722–735, Busan, Korea (2007)
3. Balazevic, I., Allen, C., Hospedales, T.M.: Hypernetwork knowledge graph embeddings. In: ICANN 2019, September 17-19, 2019, pp. 553–565, Munich, Germany (2019)
4. Bordes, A., Glorot, X., Weston, J., Bengio, Y.: A semantic matching energy function for learning with multi-relational data - application to word-sense disambiguation. Mach. Learn. **94**(2), 233–259 (2014)
5. Bordes, A., Usunier, N., García-Durán, A., Weston, J., Yakhnenko, O.: Translating embeddings for modeling multi-relational data. In: NeurIPS 2013, December 5-8, 2013, pp. 2787–2795, Lake Tahoe, Nevada, United States (2013)
6. Carlson, A., Betteridge, J., Kisiel, B., Settles, B., Hruschka, Jr.E., Mitchell, T.M.: Toward an architecture for never-ending language learning. In: AAAI 2010, July 11–15, 2010, Atlanta, Georgia, USA (2010)
7. Cohen, W.W.: Tensorlog: A differentiable deductive database. arXiv:1605.06523 (2016)

8. Das, R., Dhuliawala, S., Zaheer, M., Vilnis, L., Durugkar, I., Krishnamurthy, A., Smola, A., McCallum, A.: Go for a walk and arrive at the answer: Reasoning over paths in knowledge bases using reinforcement learning. In: ICLR 2018, April 30 - May 3, 2018 Conference Track Proceedings, Vancouver, BC, Canada (2018)
9. Das, R., Neelakantan, A., Belanger, D., McCallum, A.: Chains of reasoning over entities, relations, and text using recurrent neural networks. In: EACL 2017, April 3-7, 2017, pp. 132–141, Valencia, Spain (2017)
10. Dettmers, T., Minervini, P., Stenetorp, P., Riedel, S.: Convolutional 2d knowledge graph embeddings. In: AAAI-18, February 2-7, 2018, pp. 1811–1818, New Orleans, Louisiana, USA (2018)
11. Gardner, M., Talukdar, P.P., Krishnamurthy, J., Mitchell, T.M.: Incorporating vector space similarity in random walk inference over knowledge bases. In: EMNLP 2014, October 25-29, 2014, pp. 397–406, Doha, Qatar (2014)
12. Glorot, X., Bengio, Y.: Understanding the difficulty of training deep feedforward neural networks. In: AISTATS 2010, May, 13-15, 2010, pp. 249–256, Chia Laguna Resort, Sardinia, Italy (2010)
13. Godin, F., Kumar, A., Mittal, A.: Learning when not to answer: a ternary reward structure for reinforcement learning based question answering. In: NAACL-HLT 2019, June, 2-7, 2019, pp. 122–129, Minneapolis, MN, USA (2019)
14. Gunning, D., Stefik, M., Choi, J., Miller, T., Stumpf, S., Yang, G.: XAI - explainable artificial intelligence. Sci. Robot. vol 4(37) (2019)
15. Hochreiter, S., Schmidhuber, J.: Long short-term memory. Neural Comput. **9**(8), 1735–1780 (1997)
16. Kingma, D.P., Ba, J.: Adam: A method for stochastic optimization. In: ICLR 2015, May 7-9, 2015 Conference Track Proceedings, San Diego, CA, USA (2015)
17. Kok, S., Domingos, P.M.: Statistical predicate invention. In: ICML 2007, June, 20-24, 2007, pp. 433–440, Corvallis, Oregon, USA (2007)
18. Lao, N., Cohen, W.W.: Relational retrieval using a combination of path-constrained random walks. Mach. Learn. **81**(1), 53–67 (2010)
19. Lao, N., Mitchell, T.M., Cohen, W.W.: Random walk inference and learning in A large scale knowledge base. In: EMNLP 2011, 27-31 July 2011, John McIntyre Conference Centre, Edinburgh, UK, pp. 529–539 (2011)
20. Lehmann, J., Isele, R., Jakob, M., Jentzsch, A., Kontokostas, D., Mendes, P.N., Hellmann, S., Morsey, M., van Kleef, P., Auer, S., Bizer, C.: Dbpedia - A large-scale, multilingual knowledge base extracted from wikipedia. Semant. Web **6**(2), 167–195 (2015)
21. Lin, X.V., Socher, R., Xiong, C.: Multi-hop knowledge graph reasoning with reward shaping. In: EMNLP 2018, October 31 - November 4, 2018, pp. 3243–3253, Brussels, Belgium (2018)
22. Meilicke, C., Chekol, M.W., Ruffinelli, D., Stuckenschmidt, H.: Anytime bottom-up rule learning for knowledge graph completion. In: Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence, IJCAI 2019, August 10-16, 2019, pp. 3137–3143, Macao, China (2019)
23. Muggleton, S.: Developments in inductive logic programming, panel position paper. In: FGCS 1992, June 1-5, pp. 1071–1073, Tokyo, Japan (1992)
24. Muggleton, S.: Inverse entailment and progol. New Gener. Comput. **13**(3&4), 245–286 (1995)
25. Murdoch, W.J., Szlam, A.: Automatic rule extraction from long short term memory networks. In: ICLR 2017, Toulon, France, April 24-26, 2017 Conference Track Proceedings (2017)
26. Qu, M., Tang, J.: Probabilistic logic neural networks for reasoning. In: NeurIPS 2019, 8-14 December 2019, pp. 7710–7720, Vancouver, Canada (2019)
27. Quinlan, J.R.: Learning logical definitions from relations. Mach. Learn. **5**, 239–266 (1990)
28. Ribeiro, M.T., Singh, S., Guestrin, C.: "why should I trust you?": Explaining the predictions of any classifier. In: SIGKDD 2016, August, 13-17, 2016, pp. 1135–1144, San Francisco, CA, USA (2016)
29. Richardson, M., Domingos, P.M.: Markov logic networks. Mach. Learn. **62**(1-2), 107–136 (2006)
30. Rieger, L., Singh, C., Murdoch, W.J., Yu, B.: Interpretations are useful: penalizing explanations to align neural networks with prior knowledge. arXiv:1909.13584 (2019)
31. Rocktäschel, T., Riedel, S.: End-to-end differentiable proving. In: NeurIPS 2017, 4-9 December 2017, pp. 3788–3800, Long Beach, CA, USA (2017)
32. Shen, Y., Chen, J., Huang, P., Guo, Y., Gao, J.: M-walk: Learning to walk over graphs using monte carlo tree search. In: NeurIPS 2018, 3-8 December 2018, pp. 6787–6798, Montréal, Canada (2018)
33. Suchanek, F.M., Kasneci, G., Weikum, G.: Yago: a core of semantic knowledge. In: WWW 2007, May 8-12, 2007, pp. 697–706, Banff, Alberta, Canada (2007)
34. Sukhbaatar, S., Szlam, A., Weston, J., Fergus, R.: End-to-end memory networks. In: NeurIPS 2015, December 7-12, 2015, pp. 2440–2448, Montreal, Quebec, Canada (2015)

35. Sun, Z., Deng, Z., Nie, J., Tang, J.: Rotate: Knowledge graph embedding by relational rotation in complex space. In: ICLR 2019, May 6–9, 2019, New Orleans, LA, USA (2019)
36. Toutanova, K., Chen, D., Pantel, P., Poon, H., Choudhury, P., Gamon, M.: Representing text for joint embedding of text and knowledge bases. In: EMNLP 2015, September 17-21, 2015, pp. 1499–1509, Lisbon, Portugal (2015)
37. Trouillon, T., Welbl, J., Riedel, S., Gaussier, É., Bouchard, G.: Complex embeddings for simple link prediction. In: ICML 2016, June 19-24, 2016, pp. 2071–2080, New York City, NY, USA (2016)
38. Wang, W.Y., Mazaitis, K., Cohen, W.W.: Programming with personalized pagerank: a locally groundable first-order probabilistic logic. In: CIKM 2013, October 27 - November 1, 2013, pp. 2129–2138, San Francisco, CA, USA (2013)
39. Welbl, J., Stenetorp, P., Riedel, S.: Constructing datasets for multi-hop reading comprehension across documents. Trans. Assoc. Comput. Linguist. **6**, 287–302 (2018)
40. Williams, R.J.: Simple statistical gradient-following algorithms for connectionist reinforcement learning. Mach. Learn. **8**, 229–256 (1992)
41. Xiong, W., Hoang, T., Wang, W.Y.: Deeppath: A reinforcement learning method for knowledge graph reasoning. In: EMNLP 2017, September 9-11, 2017, pp. 564–573, Copenhagen, Denmark (2017)
42. Xu, K., Ba, J., Kiros, R., Cho, K., Courville, A.C., Salakhutdinov, R., Zemel, R.S., Bengio, Y.: Show, attend and tell: Neural image caption generation with visual attention. In: ICML 2015, July 6-11, 2015, pp. 2048–2057, Lille France (2015)
43. Yang, B., Yih, W., He, X., Gao, J., Deng, L.: Embedding entities and relations for learning and inference in knowledge bases. In: ICLR 2015, May 7-9, 2015 Conference Track Proceedings, San Diego, CA, USA (2015)
44. Yang, F., Yang, Z., Cohen, W.W.: Differentiable learning of logical rules for knowledge base reasoning. In: NeurIPS 2017, December 4-9, 2017, pp. 2319–2328, Long Beach, CA, USA (2017)
45. Zhang, S., Tay, Y., Yao, L., Liu, Q.: Quaternion knowledge graph embeddings. In: NeurIPS 2019, December 8-14, 2019, pp. 2731–2741, Vancouver, BC, Canada (2019)

Affiliations

Jinzhi Liao[1,2] · Xiang Zhao[1,2] · Jiuyang Tang[1,2] · Weixin Zeng[1,2] · Zhen Tan[1,2]

Jinzhi Liao
liaojinzhi12@nudt.edu.cn

Jiuyang Tang
jiuyang_tang@nudt.edu.cn

Weixin Zeng
zengweixin13@nudt.edu.cn

Zhen Tan
tanzhen08a@nudt.edu.cn

1 National University of Defense Technology, 109 Deya Road, Changsha, Hunan China

2 Science and Technology on Information Systems Engineering Laboratory, 109 Deya Road, Changsha, China