Learning to Walk with Dual Agents for Knowledge Graph Reasoning

Denghui Zhang[1†], Zixuan Yuan[1†], Hao Liu[2], Xiaodong Lin[1], Hui Xiong[1*]

[1]Rutgers University
[2]Hong Kong University of Science and Technology (HKUST)
{denghui.zhang, zy101, lin, hxiong}@rutgers.edu, liuh@ust.hk

Abstract

Graph walking based on reinforcement learning (RL) has shown great success in navigating an agent to automatically complete various reasoning tasks over an incomplete knowledge graph (KG) by exploring multi-hop relational paths. However, existing multi-hop reasoning approaches only work well on short reasoning paths and tend to miss the target entity with the increasing path length. This is undesirable for many reasoning tasks in real-world scenarios, where short paths connecting the source and target entities are not available in incomplete KGs, and thus the reasoning performances drop drastically unless the agent is able to seek out more clues from longer paths. To address the above challenge, in this paper, we propose a dual-agent reinforcement learning framework, which trains two agents (GIANT and DWARF) to walk over a KG *jointly* and search for the answer *collaboratively*. Our approach tackles the reasoning challenge in long paths by assigning one of the agents (GIANT) searching on cluster-level paths quickly and providing stage-wise hints for another agent (DWARF). Finally, experimental results on several KG reasoning benchmarks show that our approach can search answers more accurately and efficiently, and outperforms existing RL-based methods for long path queries by a large margin.

Introduction

Knowledge graphs (KGs) have become an essential building block of various knowledge-driven services, such as relation extraction (Mintz et al. 2009), question answering (Cui et al. 2019), and recommender systems (Zhang et al. 2016). A KG is usually defined as a directed graph $\mathcal{G} = (\mathcal{E}, \mathcal{R})$, where $\mathcal{E}$ is a collection of entity nodes, and $\mathcal{R}$ is a set of relation edges. Due to the highly incomplete nature, in practice, KGs often fail to include sufficient fact triples to satisfy the long-tail scenarios in various tasks. To this end, we focus our study in the context of automatic KG reasoning, also known as knowledge graph completion (KGC), i.e., constructing $f(e_s, r_q, ?|\mathcal{G})$ or $f(?, r_q, e_t|\mathcal{G})$ to infer missing facts by synthesizing information from multi-hop paths between the source and target nodes. One example is illustrated in Figure 1, where no direct link can be found on between the target node e_t = "U.S." and the source node e_s = "Boston". However, by leveraging existing indirect links and the query relation r_q = "LocatedIn", one is possible to infer the fact (BOSTON, LOCATEDIN, U.S.).

In the past several years, extensive research has been conducted on learning latent representations of entities ($e \in \mathcal{E}$) and relations ($r \in \mathcal{R}$) for knowledge graph reasoning by using tensor factorization or neural networks (Wang et al. 2014; Yang et al. 2014; Trouillon et al. 2016). Such embedding-based approaches mainly focus on preserving the structural information in the KG and are effective for single-hop reasoning. Also, recent works have considered exploiting reinforcement learning (RL) (Das et al. 2017; Xiong, Hoang, and Wang 2017; Shen et al. 2018) for KGC reasoning tasks, where a walking agent is leveraged over KG paths to compose single-hop triplets into multi-hop reasoning chains. For instance, MINERVA (Das et al. 2017) is an end-to-end model that adopts REINFORCE algorithm (Sutton et al. 1999) to train the RL agent to search over KGs starting from the source and arrive at the candidate answers.

However, a noteworthy issue of these walking-based models is that they rely heavily on short reasoning chains (e.g., maximum_path_length=3 in MINERVA), where the performance drops drastically if short indirect paths are also absent. Indeed, such single-agent approaches often get stuck when reasoning on a long path. The reasons are two-fold. First, KGs consist of massive entities and relations, the dimension of the discrete action space at each step is typically large (Das et al. 2017). As a result, the difficulty of reasoning (i.e., making right decisions constantly) increases drastically with the increasing number of reasoning steps. Without narrowing down the scope of representative entities and relations, the underlying agent may conduct unnecessary traverse among similar objects, and thus has low efficiency for path finding. Second, prior approaches train the agent with sparse rewards. Specifically, they only return a positive reward when the agent reaches the target entity by the end of a walking, and penalizes all actions within the path otherwise. This may result in false-negative rewards to the intermediate actions which are reasonable, and hinders the policy network from learning trustworthy long-term patterns.

To tackle the above problems, in this paper, we propose a **C**ollaborative D**u**al-agent **R**einforcement **L**earning framework, named CURL. Unlike existing walking-based RL models, which rely on one single agent to explore reasoning paths

*Hui Xiong is the corresponding author.
† Equal contribution. Author ordering determined by coin flip.

over entities and relations in KGs, we design two agents, GIANT agent and DWARF agent, to perform reasoning at different granularities and search for the answer collaboratively. Specifically, GIANT performs coarse-grained reasoning by walking rapidly over pre-defined abstractive KG clusters, while DWARF carefully traverses entities within each cluster to perform fine-grained reasoning. By leveraging a *Collaborative Policy Network*, two agents can share historical path information with each other to enhance their state representations. Moreover, we propose a *Mutual Reinforcement Reward System* to overcome the sparse reward issue. Instead of using a static final reward, we allow DWARF to borrow weighted reward from GIANT for its intermediate steps and vice versa. Intuitively, GIANT provides abstract and milestone-like hints to guide DWARF's behavior and reduce the search space. On the other hand, DWARF provides regularization over GIANT's behavior to help avoid less informative cluster-level reasoning chains. By training two agents jointly, our framework exploits the KG structure more thoroughly, i.e., from both global and local views, long and short reasoning paths, macro and micro trajectories, to improve the effectiveness of KG reasoning tasks. We conduct extensive experimental studies on three real-world KG datasets. The results demonstrate that our approach can search answers more accurately and efficiently than existing embedding-based approaches as well as traditional RL-based methods, and outperforms them on long path queries significantly.

Methodology

In this section, we first review the formal problem definition of knowledge graph reasoning and the single-agent reinforcement learning approach by (Das et al. 2017). Then we introduce our dual-agent based approach that explores path patterns at two granularity levels to tackle the long-path challenge and sparse reward problem.

Problem Definition

Formally, a knowledge graph $\mathcal{G}$ is represented as a directed graph: $\mathcal{G} = \{(e_s, r, e_o), e_s, e_o \in \mathcal{E}, r \in \mathcal{R}\}$, where $\mathcal{E}$ is the set of entities and $\mathcal{R}$ is the set of relations. Each directed link in the knowledge graph $l = (e_s, r, e_o) \in \mathcal{G}$ corresponds to a real-world fact tuple, e.g., (JOE BIDEN, PRESIDENT_OF, UNITED STATES). For the knowledge graph reasoning task, we follow the definition in prior graph walking literature, a.k.a, query answering task (Das et al. 2017). Specifically, given a query $(e_s, r_q, ?)$, where e_s is the source entity and r_q is the relation of interest, the goal of query answering is to perform efficient search over $\mathcal{G}$ and find the possible answers (i.e., correct target entities) $E_o = \{e_o\}$ where $(e_s, r_q, e_o) \notin \mathcal{G}$ due to incompleteness of KG.

Although being a promising way to discover new fact knowledge, this task is challenging as it requires an algorithm to be capable of sophisticated multi-hop reasoning over the incomplete KG. Note that we follow the *end-to-end walking for reasoning* paradigm as in MINERVA, to avoid undesirable requirements such as agent pre-training, path features precomputing, and all entities ranking in the graph.

Previous Single-Agent Reinforcement Walking

The process of walking (searching) on KG can be viewed as a Markov Decision Process (MDP) (Sutton and Barto 2018): given the query entity e_s and relation r, the agent departs from e_s, sequentially selects an outgoing edge l and traverses to a new entity until it arrives at a target answer or reaches maximum path length. The MDP can be expressed by the following essential components (we eliminate OBSERVATION part in MINERVA (Das et al. 2017) for simplicity, yet the formulation is equivalent).

State, Action, Transition, Reward Each state $s_t = (e_t, (e_s, r_q)) \in \mathcal{S}$ is a tuple where e_t is the entity visited at step t and (e_s, r_q) are the source entity and query relation. e_t can be viewed as state-dependent information while (e_s, r_q) are the global context shared by all states. The set of possible actions $A_t \in \mathcal{A}$ of at step t consists of the outgoing edges of e_t in $\mathcal{G}$. Concretely, $A_t = \{(r', e') | (e_t, r', e') \in \mathcal{G}\}$. To grant the agent an option to terminate a search, a self-loop edge is added to every A_t. Because search is unrolled for a fixed number of steps T, the self-loop acts similarly to a "stop" action. A transition function $\delta : \mathcal{S} \times \mathcal{A} \to \mathcal{S}$ is defined by $\delta(s_t, A_t) = \delta(e_t, (e_s, r_q), A_t)$. Action probability is predicted by a policy network π_θ, which takes as input the state information. Popular choices for π_θ includes simple models like MLP and sequence-aware models like RNN. In the default formulation, the agent receives a terminal reward of 1 if it arrives at a correct target entity at the end of search (i.e., within maximal steps) and 0 otherwise, i.e., $R_b(s_T) = \mathbb{1}\{(e_s, r_q, e_T) \in \mathcal{G}\}$.

Our Dual-Agent Reinforcement Walking

The above single-agent approach can effectively explore short paths on KG and discover short chains of reasoning. However, when the path length increases, the agent tends to miss the target entity and fails to catch meaningful long chains of reasoning. Contrarily, our approach launches two agents: GIANT AGENT and DWARF AGENT (short as GIANT and DWARF respectively), to collaboratively explore paths at different granularity levels and search for the answer. GIANT walks rapidly over inner clusters of the KG, DWARF slowly traverses by entities inside the clusters, while meantime, they share essential path and reward information to each other, taking advantage of a more comprehensive view (i.e., both cluster/global view and entity/local view) of KG to enhance reasoning. Figure 1 presents a concise illustration of our approach CURL.

Mapping KG to Clusters We first divide an original KG into N clusters of nodes using K-means (MacQueen et al. 1967) on the pre-trained entity embeddings[1]. Based on these clusters, we also build a cluster-wise connection graph $\mathcal{G}^c$, where two clusters will be connected if there is at least one entity-level edge between them. It can be viewed as a denser mapping of the original KG. GIANT aims to walk over $\mathcal{G}^c$ to reach a "fuzzy answer", i.e., the target cluster in which the end entity lies.

[1]We apply TransE (Bordes et al. 2013) for its efficiency in encoding the structural closeness information of KG.

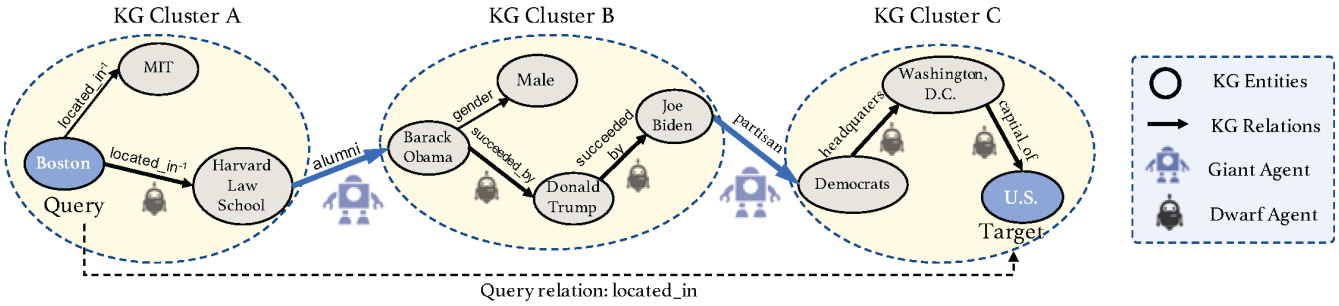


Figure 1: An illustrative diagram of the dual-agent reinforcement walking approach. Two agents work collaboratively to find the target answer (GIANT walks by clusters, DWARF walks by entities).

GIANT AGENT: Cluster-Level Exploration Following the RL paradigm, GIANT makes moves based on the state. Each state $s_t^c = (c_t, c_s) \in \mathcal{S}^c$ is a tuple where $c_t \in \mathcal{G}^c$ represents the cluster visited at step t and c_s is the start cluster which the source entity belongs to. Specifically, c_t contains state-dependent information while c_s are the global context shared by all states. At each step t, the possible actions for GIANT consists of neighbor clusters of c_t in $\mathcal{G}^c$. Concretely, $A_t^c = \{c' | (c_t, c') \in \mathcal{G}^c\}$. In other words, it traverses the KG in a fashion of cluster by cluster. Since cluster-level paths are normally shorter than entity-level paths (e.g., in Figure 1, $\text{length}_{cluster} = 3$, $\text{length}_{entity} = 7$), GIANT should be allowed to "stay" in some clusters during walking in order to *synchronize* with DWARF. To fulfill this purpose, a special action, i.e., STOP, is also added into every action space A_t^c. Each action $a_t^c \in A_t^c$ is made based on the prediction of the policy network of GIANT: π_θ^c. After walking multiple required steps, GIANT receives a terminal reward of 1 if it arrives at the cluster c_T where a correct entity answer lies in $e_T \in c_T$, and 0 otherwise.

DWARF AGENT: Entity-Level Exploration Similar to the single-agent approach, DWARF agent walks over the original KG $\mathcal{G}$ to reach an accurate answer, i.e., a target entity. Specifically, each state $s_t^e = (e_t, (e_s, r_q))$ is a tuple consisting of the current entity being visited e_t, and the query entity, relation, e_s and r_q. At each step t, to make an action, DWARF selects one from all the outgoing edges of the current entity, $A_t^e = \{(r', e') | (e_t, r', e') \in \mathcal{G}\}$. The selection is predicted by its own policy network π_θ^e. Within a maximum number of steps, if the agent arrives at a correct target entity, it will receive a final reward of 1 and 0 otherwise.

Collaborative Graph Walking: Jointly Train π_θ^e and π_θ^c Under the framework of dual-agent, to maximize the benefit of both sides, i.e., cluster-level and entity-level exploration, we propose two advances to tame the two distinct agents to walk in a mutually beneficial way, namely, Collaborative Policy Networks and Mutual Reinforcement Rewards.

Collaborative Policy Networks Agents make moves based on the output of policy network, we use two separate networks π_θ^e and π_θ^c respectively[2] to model the action selection of GIANT and DWARF. Specifically, every entity, relation in $\mathcal{G}$, as well as cluster in $\mathcal{G}^c$ is assigned a dense vector embedding $\mathbf{e} \in \mathbb{R}^d$, $\mathbf{r} \in \mathbb{R}^d$, $\mathbf{c} \in \mathbb{R}^{2d}$. We use these embeddings to represent RL actions and states of two agents. For DWARF, each action $a_t^e = (r_t, e_t) \in A_t^e$ consisting of the next outgoing relation and entity, is represented as the concatenation of the relation embedding and the end node embedding $\mathbf{a}_t^e = [\mathbf{r}_t; \mathbf{e}_t] \in \mathbb{R}^{2d}$. For GIANT, the action corresponds to the next outgoing cluster and we directly use the cluster embedding to represent it, i.e., $\mathbf{a}_t^c = \mathbf{c}_t \in \mathbb{R}^{2d}$. For both π_θ^e and π_θ^c, we first use two separate LSTMs to encode their search history $h_t^c = (c_1, \ldots, c_t) \in \mathcal{H}^c$, $h_t^e = (e_s, r_1, e_1, \ldots, r_t, e_t) \in \mathcal{H}^e$ according to the recurrent dynamics,

$$\mathbf{h}_0^c = \text{LSTM}_c(\mathbf{0}, \mathbf{c}_0),\ \mathbf{h}_0^e = \text{LSTM}_e(\mathbf{0}, [\mathbf{r}_0; \mathbf{e}_s]), \quad (1)$$

$$\mathbf{h}_t^c = \text{LSTM}_c(\mathbf{W}^c[\mathbf{h}_{t-1}^c; \mathbf{h}_{t-1}^e],\ \mathbf{a}_{t-1}^c),\ t > 0, \quad (2)$$

$$\mathbf{h}_t^e = \text{LSTM}_e(\mathbf{W}^e[\mathbf{h}_{t-1}^e; \mathbf{h}_{t-1}^c],\ \mathbf{a}_{t-1}^e),\ t > 0, \quad (3)$$

where we modify their internal structures to allow *state sharing* between LSTM_c of GIANT and LSTM_e of DWARF. Specifically, at each step $t > 0$, we calculate the concatenation of the two raw states $\mathbf{h}_t^c, \mathbf{h}_t^e \in \mathbb{R}^{2d}$ as the new states $[\mathbf{h}_t^c; \mathbf{h}_t^e], [\mathbf{h}_t^e; \mathbf{h}_t^c] \in \mathbb{R}^{4d}$. In addition, we apply two transforming matrices $\mathbf{W}^c, \mathbf{W}^e \in \mathbb{R}^{2d \times 4d}$ to reduce the dimension of the new states, otherwise, their dimension will increase exponentially w.r.t the number of steps. With the modification, each hidden state $\mathbf{h}_t^c$ ($\mathbf{h}_t^e$) of an agent is conditioned on: its own previous state $\mathbf{h}_{t-1}^c$ ($\mathbf{h}_{t-1}^e$), the other agent's previous state $\mathbf{h}_{t-1}^e$ ($\mathbf{h}_{t-1}^c$), its previous action $\mathbf{a}_{t-1}^c$ ($\mathbf{a}_{t-1}^e$). This ensures sharing essential path information between GIANT and DWARF, as to some extent, cluster-level are complementary to entity-level paths, it can consequently enable more robust action selection for both of them.

To predict the next action (i.e., the next cluster for GIANT and next relation-entity edge for DWARF), we further apply a two-layer feedforward network with ReLU nonlinearity on the concatenation of their last LSTM states and current RL state embeddings,

$$\pi_\theta^c(a_t^c | s_t^c) = \sigma(\mathbf{A}_t^c \times \mathbf{W}_2^c\ \text{ReLU}(\mathbf{W}_1^c[\mathbf{c}_t; \mathbf{h}_t^c])), \quad (4)$$

$$\pi_\theta^e(a_t^e | s_t^e) = \sigma(\mathbf{A}_t^e \times \mathbf{W}_2^e\ \text{ReLU}(\mathbf{W}_1^e[\mathbf{e}_t; \mathbf{r}_q; \mathbf{h}_t^e])), \quad (5)$$

where $\mathbf{W}_1^c, \mathbf{W}_2^c \in \mathbb{R}^{4d \times 4d}$, $\mathbf{W}_1^e, \mathbf{W}_2^e \in \mathbb{R}^{6d \times 6d}$ are the matrices of learnable weights, $\mathbf{A}_t^c \in \mathbb{R}^{|A_t^c| \times 4d}$, $\mathbf{A}_t^e \in \mathbb{R}^{|A_t^e| \times 6d}$ represent the embeddings of all next possible actions for GIANT and DWARF. σ denotes the softmax operator.

[2]The superscript c, e stands for cluster/entity.

Mutual Reinforcement Rewards The default rewards for GIANT and DWARF only consider whether their own agent arrives at the target cluster or entity, leaving two problems limiting the dual agent to collaborate effectively: (i) As GIANT walks over the cluster graph which is more densely connected, it leads to more diverse trajectories and hard to be consistent with entity-level paths. (ii) DWARF does not learn any stage-wise knowledge from GIANT explicitly. To alleviate the issue, we provide a new mutual reinforcement reward system, which can (1) constrain GIANT to generate cluster trajectories consistent with entity-level paths; (2) allow GIANT to provide stage-wise hints for DWARF to follow. Specifically, both agents' final reward consists of two parts, i.e., their own default reward and an auxiliary weighted reward borrowed from their partner,

$$R_c(s_t^c) = \underbrace{r_c(s_T^c)}_{\textit{default reward}} + \underbrace{\Phi(s_t^c, s_t^e) \cdot r_e(s_T^e)}_{\textit{partner reward}}, \ t \in [1, T], \quad (6)$$

$$R_e(s_t^e) = \underbrace{r_e(s_T^e)}_{\textit{default reward}} + \underbrace{\Phi(s_t^e, s_t^c) \cdot r_c(s_T^c)}_{\textit{partner reward}}, \ t \in [1, T], \quad (7)$$

where $r_c(s_T^c)$ and $r_e(s_T^e)$ denote the default final rewards for GIANT and DWARFT, defined as $r_c(s_T^c) = \mathbb{1}\{\exists e_a \in c_T, (e_s, r_q, e_a) \in \mathcal{G}\}, r_e(s_T^e) = \mathbb{1}\{(e_s, r_q, e_T) \in \mathcal{G}\}$. Moreover, $\Phi(s_t^c, s_t^e)$ is an evaluation function measuring the consistency of each action made by two agents. In practice, it is calculated as the cosine similarity between the pre-trained embeddings[3] of the current traversed cluster and entity, i.e., $\Phi(s_t^c, s_t^e) = \frac{\mathbf{c}_t^\top \mathbf{e}_t}{||\mathbf{c}_t||_2 ||\mathbf{e}_t||_2}$. For GIANT, its partner reward will be valid only if DWARF reaches the target entity by the end, and meantime, the current cluster visited must be close to the corresponding entity at step t so that the weight is sufficient. Similarly, for DWARF, the partner reward will be valid only if GIANT reaches the target cluster by the end and the consistency weight is sufficient. This ensures that both agents learn knowledge from partner only at the right time, i.e., when their partner succeeds to reach the correct target. The measuring coefficient $\Phi(s_t^c, s_t^e)$ controls the strength of partner reward based on the path consistency, i.e., the overlap between the cluster-level path and entity-level path.

The detailed training procedure of CURL is described in Algorithm 1. During inference, we use the same procedure of lines 3-7 below to calculate the action probabilities at each step. To find the target answer, we do a beam search with a beam width of 50 on DWARF agent and rank entities by the probabilities of the trajectory that DWARF took to reach the entity, all other unreachable entities get a rank of ∞.

Algorithm 1: CURL Training Algorithm

1: **Input:** KG $\mathcal{G}$; Initial entity and cluster nodes e_s, c_s; Entity-level query r_q; Target entity and cluster nodes e_o, c_o; Maximum path length T; Episode size P, Rollout size L
2: **Output:**Well-trained policy networks $\pi_\theta^c, \pi_\theta^e$
3: **for** episode p in $\{1, \ldots, P\}$ **do**
4: Set current entity and cluster nodes $e_0 = e_s, c_0 = c_s$
5: **for** $t = 0, \ldots, T-1$ **do**
6: Predict the next cluster c_{t+1} for GIANT and next relation-entity edge (r_{t+1}, e_{t+1}) for DWARF based on Eq. (4) - (5)
7: **end for**
8: Set default cluster-level reward $r_c = 1$ if the end of the path $c_T = c_o$ otherwise $r_c = 0$
9: Set default entity-level reward $r_e = 1$ if the end of the path $e_T = e_o$ otherwise $r_e = 0$
10: Repeat lines 5 - 9 for running L rollouts (see the expectation in Eq. (8) - (9)) to update the cluster-level and entity-level policies
11: Compute the mutual reinforcement rewards $R_e(s_t^e)$, $R_c(s_t^c)$ based on Eq. (6) - (7)
12: Update the model parameters with REINFORCE:

$$\theta^c \leftarrow \theta^c + \alpha \cdot \nabla_{\theta^c} \mathbb{E}_{A_1^c, \ldots, A_T^c \sim \pi_\theta^c} \sum_{t=0}^{T-1} [R_c(s_t^c) | s_0^c] \quad (8)$$

$$\theta^e \leftarrow \theta^e + \alpha \cdot \nabla_{\theta^e} \mathbb{E}_{A_1^c, \ldots, A_T^c \sim \pi_\theta^e} \sum_{t=0}^{T-1} [R_e(s_t^e) | s_0^e] \quad (9)$$

13: **end for**
14: **return** $\pi_\theta^c, \pi_\theta^e$

Experiments

Experiment Setup

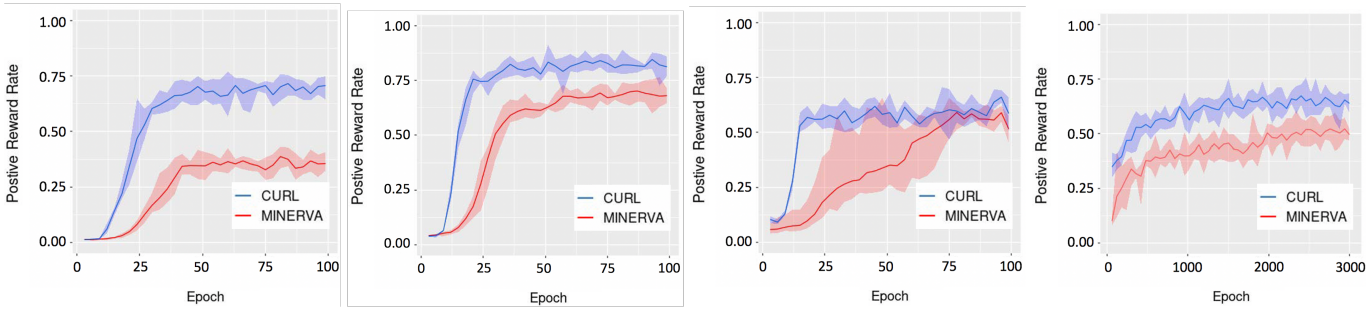
We evaluate the effectiveness of CURL[4] by performing two fundamental tasks using three real-world KG datasets, i.e., FB15K-237, WN18RR, and NELL-995. The WN18RR (Dettmers et al. 2018) and FB15K-237 (Toutanova et al. 2015) datasets are separately created from the original WN18 and FB15K datasets by removing various sources of test leakage, making the datasets more realistic and challenging. The NELL-995 dataset released by (Xiong, Hoang, and Wang 2017) contains separate graphs for each query relation. We compare against two classes of state-of-the-art KG reasoning baselines: KG representation learning methods (TransE (Wang et al. 2014), TransR (Lin et al. 2015b), DistMult (Yang et al. 2014), CompIEx (Trouillon et al. 2016)) and multi-hop neural approaches (NeuralLP (Yang, Yang, and Cohen 2017), PRA (Lao, Mitchell, and Cohen 2011), DeepPath (Xiong, Hoang, and Wang 2017), MINERVA (Das et al. 2017), M-Walk (Shen et al. 2018)). For reproducing all baseline results, we used the implementation released by authors on the best hyperparameter settings reported by them [5]. CURL and all baselines are implemented with Pytorch framework (Paszke et al. 2019) and run on a single 2080 Ti GPU. More experimental details and hyperparameters of our model are illustrated in Appendix A. According to (Shen et al. 2018), KG reasoning can be divided into the following tasks:

Link Prediction (Query Answering) Given an incomplete KG, the link prediction task aims to predict the missing entities in the unknown links. In our settings, for a query $(e_1, r, ?)$ or $(?, r, e_2)$, we run multiple rollouts to search for answer node based on query relation and source entity, and

[3]Each cluster embedding is obtained by averaging all entity embeddings within it.

[4]Source code: https://github.com/RutgersDM/DKGR/tree/master

[5]We obtain and report the best results using the code and hyperparameters released by the authors of the baseline models.



(a) TeamPlaysSport. (b) AthletePlaysInLeague. (c) PersonBornInLocation. (d) NELL-995.

Figure 2: The positive reward rate on three NELL-995 relation tasks (a - c) and all tasks (d), and our agent is consistently better than MINERVA.

Model	FB15K-237				WN18RR				NELL-995			
	@1	@3	@10	MRR	@1	@3	@10	MRR	@1	@3	@10	MRR
TransE	24.8	40.1	45.0	36.1	28.9	46.4	53.4	35.9	51.4	67.8	75.1	45.6
DistMult	27.5	41.7	56.8	37.0	41.0	44.1	47.5	43.3	61.0	73.3	79.5	68.0
ComplEx	**30.3**	**43.4**	**57.2**	**39.4**	38.2	43.3	48.0	41.5	61.2	76.1	82.1	68.4
NeuralLP	16.6	24.8	34.8	22.7	37.6	46.8	**65.7**	45.9	–	–	–	–
MINERVA	19.2	30.7	42.6	27.1	41.3	45.6	51.3	44.8	58.8	74.6	81.3	67.5
M-Walk	16.8	24.5	40.3	23.4	41.5	44.7	54.3	43.7	63.2	75.7	81.9	70.7
CURL	22.4	34.1	47.0	30.6	**42.9**	**47.1**	52.3	**46.0**	**66.7**	**78.6**	**84.3**	**73.8**

Table 1: Query answering performance compared to state-of-the-art embedding based approaches (top part) and multi-hop reasoning approaches (bottom part). The Hits@1, 3, 10 and MRR metrics were multiplied by 100. We highlight the best approach in each category.

Task	TransR	TransE	PRA	DeepPath	MINERVA	M-Walk	CURL
AthletePlaysInLeague	91.2	77.3	84.1	96.0	94.0	96.1	**97.1**
AthletePlaysForTeam	67.3	62.7	54.7	75.0	80.0	**84.7**	82.9
AthleteHomeStadium	72.2	71.8	85.9	89.0	89.8	91.9	**94.3**
TeamPlaysSports	81.4	76.1	79.1	73.8	88.0	88.4	**88.7**
AthletePlaysSport	96.3	87.6	47.4	95.7	98.0	98.3	**98.4**
OrganizationHiredPerson	73.7	71.9	59.9	74.2	85.6	**88.8**	87.6
PersonBornInLocation	81.2	71.2	66.8	75.7	78.0	81.2	**82.1**
WorksFor	69.2	67.7	68.1	71.1	81.0	**83.2**	82.1
OrgHeadquarteredInCity	65.7	62.0	81.1	79.0	94.0	94.3	**94.8**
PersonLeadsOrganization	77.2	75.1	70.0	79.5	87.7	88.3	**88.9**

Table 2: Performance on fact prediction, MAP scores for different relation tasks in NELL-995 dataset.

then do a beam search with a beam width of 50 to rank the entities by the probability of their trajectories reaching the correct entity. Here, we use Hits@1, 3, 10 and Mean Reciprocal Ranking (MRR) as standard ranking metrics (Xiong, Hoang, and Wang 2017; Sutton and Barto 2018).

Fact Prediction Subtly different from link prediction, fact prediction task targets at inferring whether an unknown fact (triple) holds or not. According to (Xiong, Hoang, and Wang 2017), the true test triples are ranked with some generated false triples. In the experiments, we first remove all links of groundtruth relations in the raw KG. Then the dual agents try to infer and walk through the KG to reach the target entity. Since we share a similar query-answering mechanism as MINERVA (Das et al. 2017), CURL can directly locate the correct entity node for a given query, and eliminate the need to evaluate negative samples any particular relation. Note that if CURL fails to reach any of the entities in the set of correct and negative entities, it then falls back to a random ordering of the entities. Here, we report Mean Average Precision (MAP) scores for various relation tasks of NELL-995 (corresponding to different subsets).

Overall Performance

As demonstrated in Table 1, we first report the performances of CURL and all baselines on the link prediction task. The results of NeuralLP are not included on NELL-995 because it can not scale to the size larger dataset. We observe that on FB15K-237, the embedding based methods dominate over several neural multi-hop reasoning approaches. With deeper investigation, we discover that the structural characteristics of FB15K-237 differ significantly from WN18RR and NELL-995, since it contains much larger number of 1-to-M than the M-to-1 relation instances (Wan et al. 2020). This indicates that the search process of multi-hop reasoning methods is prone to be stuck in the local nodes with high-degree centrality, renders it hard to reach the correct entity. Note that on WN18RR, neural symbolic methods (MINERVA, NeuralLP, M-Walk) generally beat embedding based methods, with CURL achieving the highest score on Hits@1, 3 and MRR metrics. On NELL-995, our approach delivers comparable performance to embedding based methods, further outperforms MINERVA by a clear margin on all metrics. After averaging results on three datasets, we find that our dual-agent based approach leads to overall improvements relative to the single-agent approach with similar settings (MINERVA) by $3.1\%, 2.1\%, 1.7\%, 2.3\%$, in terms of Hits@1, 3, 10 and MRR.

Table 2 reports the performance of fact prediction on 10 relation tasks of NELL-995. Our approach produces an encouraging result in most tasks, contributing an average gain of 7.8% relative to the multi-hop neural methods (PRA, DeepPath, MINERVA, and M-walk) and 14.8% gain compared to the embedding-based approaches (TransR and TransE).

Analysis of CURL and Case Studies

Based on the above results, we conducted the analysis to discuss the superiority of our dual-agent framework on KG

(i) **Can learn shorter path:** Oklahoma Thunder $\xrightarrow{\text{TeamPlaysInLeague}}$?

Oklahoma Thunder $\xrightarrow{\text{SubPartOfOrganization}}$ NBA

Oklahoma Thunder $\xrightarrow{\text{SportGameTeam}^{-1}}$ SportGames $\xrightarrow{\text{SportGameTeam}}$ Boston Celtics $\xrightarrow{\text{SubPartOfOrganization}}$ NBA

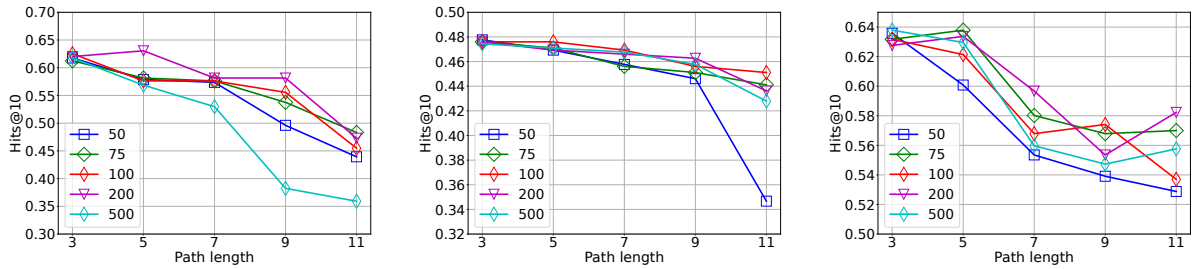
(ii) **Can learn longer path:** Prof. Stephanie Strom $\xrightarrow{\text{WorkFor}}$?

Prof. Stephanie Strom $\xrightarrow{\text{WritesForPublication}}$ Times Newspaper $\xrightarrow{\text{WritesForPublication}^{-1}}$ Journalist David Johnston $\xrightarrow{\text{WorksFor}}$ NY Times Website $\xrightarrow{\text{WritesForPublication}^{-1}}$ Journalist Adam Liptak $\xrightarrow{\text{WorksFor}}$ Times MusicArtist

(iii) **Can recover from mistakes in longer path:** US Government $\xrightarrow{\text{OrgHiredPerson}}$?

US Government $\xrightarrow{\text{AgentParticipatedIn}}$ Crime Charge $\xrightarrow{\text{AgentParticipatedIn}^{-1}}$ Adolescent MusicArtists $\xrightarrow{\text{AgentParticipatedIn}}$ **Event Outcome** $\xrightarrow{\text{AgentParticipatedIn}^{-1}}$ "Women of MORE Magazine" $\xrightarrow{\text{AgentParticipatedIn}}$ **Event Outcome** $\xrightarrow{\text{AgentParticipatedIn}^{-1}}$ Government Law $\xrightarrow{\text{AgentParticipatedIn}^{-1}}$ Ryan Whitney

Table 3: A few examples of paths found by CURL on NELL-995.



(a) AthletePlaysForTeam. (b) AthletePlaysSport. (c) OrgHiredPerson.

Figure 3: The effect of different cluster numbers used by GIANT. We present the link prediction performance on three relation tasks on NELL-995. Lines in different colors indicate results by different cluster numbers.

reasoning. First, the above quantitative results show that our dual-agent design contributes significant gains relative to the DWARF-solely method (i.e., MINERVA) on all datasets, confirming the effectiveness of our high-level motivation that using cluster-level reasoning to guide entity-level reasoning can alleviate the long path challenge and sparse reward issue. To further examine this, in Figure 2, we show the positive reward rate (i.e., the percentage of trajectories with positive reward during training) on the NELL-995 tasks. Compared to MINERVA under the same training and testing procedure, CURL is capable of generating trajectories with more positive rewards, and this continues to improve as training progresses. Additionally, in Figure 5 (in Appendix B.1), we show the Hits@1 performance change w.r.t the increasing path length, where we find that CURL maintains a relatively stable result or slower ratio of performance degradation. These two evidences prove that introducing GIANT agent by our Collaborative Walking and Mutual Reinforcement Reward can indeed benefit the entity-level KG reasoning. Finally, we present some illustrative examples of paths found by CURL in Table 3. Example (i) and (ii) illustrate that CURL is capable of capturing both short and long reasoning chains for diverse tasks. Example (iii) displays the ability to correct a previously taken decision even in the long paths, where our model took an incorrect decision at the first step but was able to revert the decision because of the presence of inverted edges. This property is similar to MINERVA, which however cannot well handle the long-path reasoning compared to us.

Long Path Reasoning Performance

While short chains (length$\leq$ 3) can partially support KG reasoning and navigate the agent to find target answers (Das et al. 2017), we consider a more practical and rigorous scenario in real-world KGs, where short paths are mostly absent from incomplete KGs. For effective evaluation, we compare our model with MINERVA in NELL-995 dataset, where we remove the most frequently-visited short paths found by the bi-directional search (BiS) (Xiong, Hoang, and Wang 2017) inside KG. In particular, given a triplet (e_s, r, e_o) in the training or testing set, we sample an intermediate entity node e_i and use the breadth-first search algorithm (Bundy and Wallen 1984) to verify the traversability from e_s to e_i and from e_i to e_o inside KG. After conducting the BiS on each task 50 times, we choose the walkable paths with a length smaller than 3 (self-included), and eliminate their traversed edges inside original KGs. Due to the space limit, we plot the MAP scores on varying path lengths in various tasks in Figure 4 in Appendix B.2. As can be seen, CURL outperforms MINERVA across all three tasks with different path lengths, and our gains are much more prominent when path length is 5. This explains that the cluster-level exploration is essential to lead entity-level agent rather than doing purely random walks

in the neighborhood of the source entity. Overall, CURL is much more robust to queries where a longer reasoning path is required, showing minimal degradation in performance for even the longest path setting. Additional experiments on FB15K-237, WN18RR, and NELL-995 can be found in Appendix B.2 of the supplementary material.

Sensitivity Test on Cluster Size

Figure 3 shows the Hits@10 scores of CURL under different cluster number N during training on three NELL-995 tasks. We observe a performance improvement when we increase N from 50 to 75 and performance degrade when we further increase N from 75 to 500. These results illustrate 75 cluster number is powerful enough to capture high-level semantic information. In other words, parametrized by a suitable N, our approach is generally stable when the path lengths increase, indicating that proper clustering of KG entities does refine a series of meaningful high-level semantics to facilitate the low-level path searching.

Related Work

Knowledge Graph Representation Learning Representation learning has shown great success in a wide range of fields (Zhang et al. 2019, 2017; Li et al. 2021; Yuan et al. 2021) and KG reasoning is no exception. Recent advances in this area have proposed a variety of embedding-based methods that project the entities and relations into low-dimensional continuous vector space by exploiting entity types (Guo et al. 2015; Ouyang et al. 2017), relation paths (Lin et al. 2015a; Toutanova et al. 2016; Li et al. 2018a; Zhang et al. 2018), textual descriptions (Zhong et al. 2015; Xiao et al. 2017), and logical rules (Omran, Wang, and Wang 2018; Hamilton et al. 2018). For instance, TransE (Wang et al. 2014) first encoded the entities and relations into latent vectors by following translational principle in point-wise Euclidean space. Besides, ComplEx (Trouillon et al. 2016) introduced complex vector space to capture both symmetric and anti-symmetric relations. However, such models ignored the symbolic compositionality of KG relations, making them unable to discover complex reasoning paths with one-hop distance-based measure (Wang et al. 2017; Ji et al. 2021). Furthermore, it is hard to interpret the traversal paths, and these models can be computationally expensive to access the entire graph in memory.

Deep Reinforcement Learning for KG Reasoning The emergence of deep reinforcement learning (RL) enables many path-based approaches to learn symbolic inference rules from relational paths inside KG (Ji et al. 2021). By formulating KG reasoning as a sequential decision problem and taking multi-hop random walks, existing studies improve the empirical performance of various tasks (Xiong, Hoang, and Wang 2017; Shen et al. 2018; Lin, Socher, and Xiong 2018; Das et al. 2017; Wan et al. 2020; Hildebrandt et al. 2020) including KG completion, fact prediction, and query answering. Specifically, DeepPath (Xiong, Hoang, and Wang 2017) first introduced RL to search for diversified representative paths between entity pairs. M-Walk (Shen et al. 2018) further proposed to solve the reward sparsity problem in MCTS-based query answering in an off-policy manner. Reward shaping based approach (Lin, Socher, and Xiong 2018) leveraged pre-trained embeddings to estimate the likelihood of unseen facts, with the purpose of reducing the impact of false-negative supervision as well as facilitating the path inference. Our work aligns with the RL formulation of MINERVA (Das et al. 2017), i.e., learning to walk and search for answer entities of a particular KG query in an end-to-end fashion. Note that MINERVA solely exploited the entity-level KG information to update the policy network. In comparison, we trained the entity-level policy of the DWARF AGENT using the cluster-level semantics and trajectories generated by the GIANT AGENT, such that the dual agents can collaborative to reach the optimal answers given queries.

Regarding the dual-agent structure, our work is closely related to the hierarchical RL (Barto and Mahadevan 2003; Wang, Li, and He 2018; Li et al. 2018b; Wan et al. 2020; Wang et al. 2020; Wen et al. 2020) in the sense of leveraging both low-level and high-level policies to solve long-horizon problems with sparse rewards. However, instead of requiring high-level policy to specify subgoals for low-level skills, CURL enables both policies to achieve the subgoals possessed by their counterparts. That is, with the faster convergence in a reduced search space, GIANT AGENT can provide high-level stepwise guidance for DWARF AGENT to follow, while DWARF AGENT walks along the real-existing paths, thus checking the correctness of abstractive cluster-level paths found by GIANT AGENT. Such a learning scheme can avoid extra design efforts for complex subgoal space, which is not always trustworthy and tractable (Zhang, Yao, and Chen 2019).

Conclusion

We proposed a dual-agent framework (**CURL**) that learns to walk over a KG for searching desired answer nodes given query relation and source entity. Specifically, we first leveraged LSTM to project trajectory history into latent vectors of different granularities and semantics. To facilitate the entity-level exploration, CURL launched two agents: GIANT and DWARF to collaboratively explore paths at different granularity levels and search for the answer. GIANT walks rapidly over inner clusters of the KG, which can guide DWARF to smoothly traverse through the entities inside the clusters. We later developed the collaborative policy network for sharing historical path information between two agents, and established the mutual reinforcement reward system for handling sparse reward issue. Experimental results on several knowledge graph reasoning benchmarks show that our approach can search for answers more accurately and efficiently. Furthermore, we compared with the DWARF-solely MINERVA in the long-path experiment. We found that our method is more accurate in long path reasoning, which can be explained by that the stage-wise signals provided by GIANT do play a critical role in leading the DWARF towards the target node.

Dataset	#entities	#relations	#facts	#queries	#degree mean	#degree median
WN18RR	40,945	11	86,835	3,134	2.19	2
NELL-995	75,492	200	154,213	3,992	4.07	1
FB15K-237	14,505	237	272,115	20,466	19.74	14

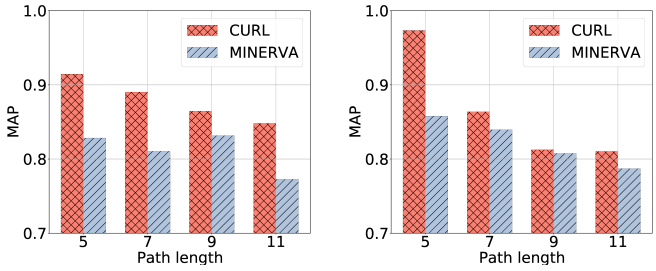
Table 4: Statistics of three datasets used in our experiments.

Appendix

A. Experiment Details

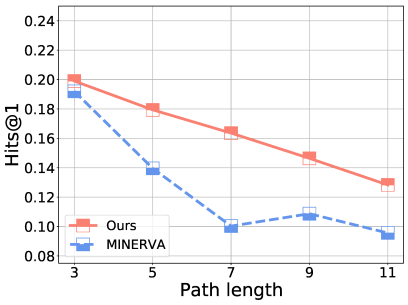
A.1. Statistics of Datasets The NELL-995 knowledge dataset contains $75,492$ unique entities and 200 relations. WN18RR contains $93,003$ triples with $40,943$ entities and 11 relations. And FB15K-237, a subset of FB15K where inverse relations are removed, contains $14,541$ entities and 237 relations. The detailed statistics are shown in Table 4.

A.2. Optimal Hyperparameters To reproduce the results of our model in Table 1 and Table 2, we report the empirically optimal hyperparameters. Specifically, we set the entity embedding dimension to 50 and relation embedding dimension to 50. We use the K-means algorithm (MacQueen et al. 1967) and the pre-trained entity embeddings to initialize 75 clusters for WN18RR and NELL-995, and 50 clusters for FB15K-237, respectively. After the maximum step T has been reached, CURL evaluates the action sequence and assigns the mutual rewards according to lines 10, 11 in Algorithm 1. The action embedding is the concatenation of the entity embedding vector and the relation embedding vector. We use two single-layer LSTMs with hidden state size of 200 for GIANT and DWARF, respectively. On all datasets, the quantities of path rollouts in training and testing are 20 and 100, separately. We use the Adam optimization (Kingma and Ba 2014) in REINFORCE for model training with learning rate as 0.001, and the best mini-batch size is 128. For the rest parameters, i.e., maximum path length T, the entropy regularization constant β, and the moving average constant λ, the best combination of them are $\{3, 0.2, 0.2\}$ for FB15K-237, $\{3, 0.06, 0.00\}$ for WN18RR, $\{3, 0.07, 0.07\}$ for NELL-995.

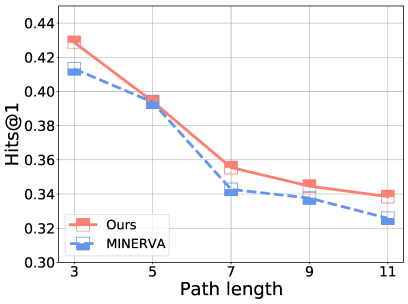


(a) OrgHeadqInCity. (b) AthletePlaysInLeague.

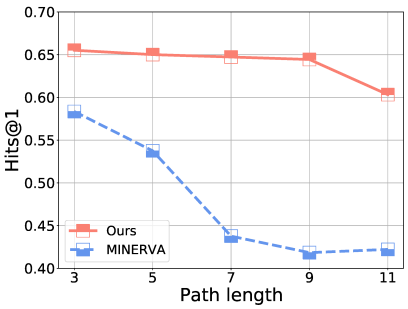
Figure 4: The long-path performance: CURL significantly outperforms MINERVA on NELL-995 tasks.



(a) FB15K-237.



(b) WNRR18.



(c) NELL-995.

Figure 5: The long-path experiments: we significantly outperform MINERVA in three large datasets.

B. Additional Experiments

B.1. Parameter Sensitivity on Beam Search Size

We also check the influence of different beam search size during testing, as reported in Table 5. As can be seen, except when beam size is 20, the performances of CURL are much higher than MINERVA, and our gain is maximized at beam search size 50. Moreover, the test accuracy doesn't change substantially with larger beam search sizes.

B.2. Long Path Recovery

We conduct further long-path experiments on three datasets, including FB15K-237, WN18RR, and NELL-995. Here, we investigate the ability of recovering from mistakes in the long-path traversal. Unlike Section "Long Path Reasoning Performance", we retain their original KGs and set the walkable path length to be relatively larger (> 3), since short

Size		MINERVA	CURL
20	Hits@1	72.3	**74.7**
	Hits@10	**81.9**	81.1
	MRR	75.8	**77.2**
50	Hits@1	70.7	**75.9**
	Hits@10	81.1	**82.3**
	MRR	74.9	**78.5**
100	Hits@1	59.4	**76.3**
	Hits@10	76.3	**82.5**
	MRR	66.0	**78.9**
200	Hits@1	69.9	**75.1**
	Hits@10	79.9	**82.7**
	MRR	73.6	**78.2**
500	Hits@1	68.7	**73.1**
	Hits@10	78.3	**83.9**
	MRR	72.0	**77.1**

Table 5: Hits@1, 10 and MRR test accuracy (%) in task of OrganizationHeadquarteredInCity, where "Size" denotes the width of beam search.

chains in these datasets can usually produce good empirical results (Das et al. 2017; Wan et al. 2020), and the longer ones are more likely to walk through unnecessary or incorrect links. As reported in Figure 5, CURL outperforms the other DWARF-only method (i.e., MINERVA) in all datasets. This observation demonstrates that dual-agent design is able to stabilize the entity-level searching in long paths by utilizing the cluster-level reasoning information. One possible reason is that GIANT converges much faster in a reduced search space, consistently providing high-level stage-wise guidance for DWARF to follow.

Acknowledgements

This work was partially supported by the National Science Foundation through award 2006387, 1814510, 2040799.

References

Barto, A. G.; and Mahadevan, S. 2003. Recent Advances in Hierarchical Reinforcement Learning. *Discrete event dynamic systems*, 13(1): 41–77.

Bordes, A.; Usunier, N.; Garcia-Duran, A.; Weston, J.; and Yakhnenko, O. 2013. Translating Embeddings for Modeling Multi-Relational Data. In *Neural Information Processing Systems (NIPS)*, 1–9.

Bundy, A.; and Wallen, L. 1984. Breadth-First Search. In *Catalogue of artificial intelligence tools*, 13–13. Springer.

Cui, W.; Xiao, Y.; Wang, H.; Song, Y.; Hwang, S.-w.; and Wang, W. 2019. KBQA: Learning Question Answering over QA Corpora and Knowledge Bases. *arXiv preprint arXiv:1903.02419*.

Das, R.; Dhuliawala, S.; Zaheer, M.; Vilnis, L.; Durugkar, I.; Krishnamurthy, A.; Smola, A.; and McCallum, A. 2017. Go For A Walk And Arrive At The Answer: Reasoning Over Paths In Knowledge Bases Using Reinforcement Learning. *arXiv preprint arXiv:1711.05851*.

Dettmers, T.; Minervini, P.; Stenetorp, P.; and Riedel, S. 2018. Convolutional 2D Knowledge Graph Embeddings. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 32.

Guo, S.; Wang, Q.; Wang, B.; Wang, L.; and Guo, L. 2015. Semantically Smooth Knowledge Graph Embedding. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, 84–94.

Hamilton, W. L.; Bajaj, P.; Zitnik, M.; Jurafsky, D.; and Leskovec, J. 2018. Embedding logical queries on knowledge graphs. *arXiv preprint arXiv:1806.01445*.

Hildebrandt, M.; Serna, J. A. Q.; Ma, Y.; Ringsquandl, M.; Joblin, M.; and Tresp, V. 2020. Reasoning on Knowledge Graphs with Debate Dynamics. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, 4123–4131.

Ji, S.; Pan, S.; Cambria, E.; Marttinen, P.; and Philip, S. Y. 2021. A Survey on Knowledge Graphs: Representation, Acquisition, and Applications. *IEEE Transactions on Neural Networks and Learning Systems*.

Kingma, D. P.; and Ba, J. 2014. Adam: A Method for Stochastic Optimization. *arXiv preprint arXiv:1412.6980*.

Lao, N.; Mitchell, T.; and Cohen, W. 2011. Random Walk Inference and Learning in A Large Scale Knowledge Base. In *Proceedings of the 2011 conference on empirical methods in natural language processing*, 529–539.

Li, M.; Zhang, D.; Jia, Y.; none Wang, Y.; and Cheng, X. 2018a. Link Prediction in Knowledge Graphs: A Hierarchy-Constrained Approach. *IEEE Transactions on Big Data*.

Li, Y.; Chen, Z.; Zha, D.; Du, M.; Zhang, D.; Chen, H.; and Hu, X. 2021. Interpretable Time-series Representation Learning With Multi-Level Disentanglement. *arXiv preprint arXiv:2105.08179*.

Li, Z.; Jin, X.; Guan, S.; Wang, Y.; and Cheng, X. 2018b. Path reasoning over knowledge graph: A multi-agent and reinforcement learning based method. In *2018 IEEE International Conference on Data Mining Workshops (ICDMW)*, 929–936. IEEE.

Lin, X. V.; Socher, R.; and Xiong, C. 2018. Multi-Hop Knowledge Graph Reasoning with Reward Shaping. *arXiv preprint arXiv:1808.10568*.

Lin, Y.; Liu, Z.; Luan, H.; Sun, M.; Rao, S.; and Liu, S. 2015a. Modeling Relation Paths for Representation Learning of Knowledge Bases. *arXiv preprint arXiv:1506.00379*.

Lin, Y.; Liu, Z.; Sun, M.; Liu, Y.; and Zhu, X. 2015b. Learning Entity and Relation Embeddings for Knowledge Graph Completion. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 29.

MacQueen, J.; et al. 1967. Some Methods for Classification and Analysis of Multivariate Observations. In *Proceedings of the fifth Berkeley symposium on mathematical statistics and probability*, volume 1, 281–297. Oakland, CA, USA.

Mintz, M.; Bills, S.; Snow, R.; and Jurafsky, D. 2009. Distant supervision for relation extraction without labeled data. In *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP*, 1003–1011.

Omran, P. G.; Wang, K.; and Wang, Z. 2018. Scalable Rule Learning via Learning Representation. In *IJCAI*, 2149–2155.

Ouyang, X.; Yang, Y.; He, L.; Chen, Q.; and Zhang, J. 2017. Representation Learning with Entity Topics for Knowledge Graphs. In *International Conference on Knowledge Science, Engineering and Management*, 534–542. Springer.

Paszke, A.; Gross, S.; Massa, F.; Lerer, A.; Bradbury, J.; Chanan, G.; Killeen, T.; Lin, Z.; Gimelshein, N.; Antiga, L.; et al. 2019. Pytorch: An imperative style, high-performance deep learning library. *arXiv preprint arXiv:1912.01703*.

Shen, Y.; Chen, J.; Huang, P.-S.; Guo, Y.; and Gao, J. 2018. M-Walk: Learning to Walk over Graphs using Monte Carlo Tree Search. *arXiv preprint arXiv:1802.04394*.

Sutton, R. S.; and Barto, A. G. 2018. *Reinforcement learning: An introduction*. MIT press.

Sutton, R. S.; McAllester, D. A.; Singh, S. P.; Mansour, Y.; et al. 1999. Policy Gradient Methods for Reinforcement Learning with Function Approximation. In *NIPs*, volume 99, 1057–1063. Citeseer.

Toutanova, K.; Chen, D.; Pantel, P.; Poon, H.; Choudhury, P.; and Gamon, M. 2015. Representing Text for Joint Embedding of Text and Knowledge Bases. In *Proceedings of the 2015 conference on empirical methods in natural language processing*, 1499–1509.

Toutanova, K.; Lin, X. V.; Yih, W.-t.; Poon, H.; and Quirk, C. 2016. Compositional Learning of Embeddings for Relation Paths in Knowledge Base and Text. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, 1434–1444.

Trouillon, T.; Welbl, J.; Riedel, S.; Gaussier, É.; and Bouchard, G. 2016. Complex Embeddings for Simple Link Prediction. In *International Conference on Machine Learning*, 2071–2080. PMLR.

Wan, G.; Pan, S.; Gong, C.; Zhou, C.; and Haffari, G. 2020. Reasoning Like Human: Hierarchical Reinforcement Learning for Knowledge Graph Reasoning. In *International Joint Conference on Artificial Intelligence 2020*, 1926–1932. Association for the Advancement of Artificial Intelligence (AAAI).

Wang, Q.; Mao, Z.; Wang, B.; and Guo, L. 2017. Knowledge Graph Embedding: A Survey of Approaches and Applications. *IEEE Transactions on Knowledge and Data Engineering*, 29(12): 2724–2743.

Wang, S.; Wei, X.; dos Santos, C.; Wang, Z.; Nallapati, R.; Arnold, A.; Xiang, B.; and Philip, S. Y. 2020. H2KGAT: Hierarchical Hyperbolic Knowledge Graph Attention Network. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 4952–4962.

Wang, W. Y.; Li, J.; and He, X. 2018. Deep Reinforcement Learning for NLP. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics: Tutorial Abstracts*, 19–21.

Wang, Z.; Zhang, J.; Feng, J.; and Chen, Z. 2014. Knowledge Graph Embedding by Translating on Hyperplanes. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 28.

Wen, Z.; Precup, D.; Ibrahimi, M.; Barreto, A.; Van Roy, B.; and Singh, S. 2020. On Efficiency in Hierarchical Reinforcement Learning. *Advances in Neural Information Processing Systems*, 33.

Xiao, H.; Huang, M.; Meng, L.; and Zhu, X. 2017. SSP: Semantic Space Projection for Knowledge Graph Embedding with Text Descriptions. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 31.

Xiong, W.; Hoang, T.; and Wang, W. Y. 2017. DeepPath: A Reinforcement Learning Method for Knowledge Graph Reasoning. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, 564–573.

Yang, B.; Yih, W.-t.; He, X.; Gao, J.; and Deng, L. 2014. Embedding Entities and Relations for Learning and Inference in Knowledge Bases. *arXiv preprint arXiv:1412.6575*.

Yang, F.; Yang, Z.; and Cohen, W. W. 2017. Differentiable Learning of Logical Rules for Knowledge Base Reasoning. *arXiv preprint arXiv:1702.08367*.

Yuan, Z.; Liu, H.; Hu, R.; Zhang, D.; and Xiong, H. 2021. Self-Supervised Prototype Representation Learning for Event-Based Corporate Profiling.

Zhang, D.; Li, M.; Jia, Y.; Wang, Y.; and Cheng, X. 2017. Efficient parallel translating embedding for knowledge graphs. In *Proceedings of the International Conference on Web Intelligence*, 460–468.

Zhang, D.; Liu, J.; Zhu, H.; Liu, Y.; Wang, L.; Wang, P.; and Xiong, H. 2019. Job2Vec: Job title benchmarking with collective multi-view representation learning. In *Proceedings of the 28th ACM International Conference on Information and Knowledge Management*, 2763–2771.

Zhang, F.; Yuan, N. J.; Lian, D.; Xie, X.; and Ma, W.-Y. 2016. Collaborative knowledge base embedding for recommender systems. In *Proceedings of the 22nd ACM SIGKDD international conference on knowledge discovery and data mining*, 353–362.

Zhang, M.; Wang, Q.; Xu, W.; Li, W.; and Sun, S. 2018. Discriminative Path-Based Knowledge Graph Embedding for Precise Link Prediction. In *European Conference on Information Retrieval*, 276–288. Springer.

Zhang, Y.; Yao, Q.; and Chen, L. 2019. Interstellar: Searching Recurrent Architecture for Knowledge Graph Embedding. *arXiv preprint arXiv:1911.07132*.

Zhong, H.; Zhang, J.; Wang, Z.; Wan, H.; and Chen, Z. 2015. Aligning Knowledge and Text Embeddings by Entity Descriptions. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, 267–272.