DAPath: Distance-Aware Knowledge Graph Reasoning Based on Deep Reinforcement Learning

Prayag Tiwari[a,1,*], Hongyin Zhu[b,1], Hari Mohan Pandey[c,*]

[a]*Department of Information Engineering, University of Padova, Padova, Italy*
[b]*Institute of Automation, Chinese Academy of Sciences,Beijing, China*
[c]*Department of Computer Science, Edge Hill University, Ormskirk, L39 4QP, United Kingdom.*

Abstract

Knowledge graph reasoning aims to find reasoning paths for relations over incomplete knowledge graphs (KG). Prior works may not take into account that the rewards for each position (vertex in the graph) may be different. We propose the distance-aware reward in the reinforcement learning framework to assign different rewards for different positions. We observe that KG embeddings are learned from independent triples and therefore cannot fully cover the information described in the local neighborhood. To this effect, we integrate a graph self-attention (GSA) mechanism to capture more comprehensive entity information from the neighboring entities and relations. To let the model remember the path, we incorporate the GSA mechanism with GRU to consider the memory of relations in the path. Our approach can train the agent in one-pass, thus eliminating the pre-training or fine-tuning process, which significantly reduces the problem complexity. Experimental results demonstrate the effectiveness of our method. We found that our model can mine more balanced paths for each relation.

Keywords: knowledge graph reasoning, reinforcement learning, graph self-attention, GRU

1. Introduction

Knowledge graph reasoning aims to complement missing triples in the knowledge base. For example, "The Eiffel Tower is located in Paris." can be represented as a machine-readable triple (Eiffel Tower, located, Paris) in a knowledge graph where the three elements are called the **h**ead entity (h), **r**elation (r) and **t**ail entity (t). This task aims to complete any triples (h,r,?) based on the existing KG. Our research is conducted in the context of multi-hop inference, which can learn explicit inference formulas given a large KG.

Xiong et al. [1] propose a reinforcement learning (RL) [2, 3, 4, 5, 6, 7, 8] method, DeepPath. It solves the problems of learning explicit inference formula based on the continuous state of an RL agent. However,

*Corresponding authors
Email addresses: prayag.tiwari@dei.unipd.it (Prayag Tiwari), zhuhongyin14@mails.ucas.ac.cn (Hongyin Zhu), pandeyh@edgehill.ac.uk (Hari Mohan Pandey)
[1]Prayag Tiwari and Hongyin Zhu contributed equally to this work.

this method only considers the overall reward and treats the reward of each position (vertex of the graph) equally, without taking into account that the rewards of different positions may be different. Inspired by the path-cost propagation [9], we simply assume that when the agent is at different positions, the rewards of taking action are different. We propose the distance-aware (DA) reward. When the agent approaches the target entity, choosing a correct action will generate a greater potential reward. However, when the agent is close to the source entity, choosing a correct action may not generate high reward, because there are still many subsequent challenges to complete this episode. Whether the agent performs well at the beginning of the episode and whether the agent performs well at the end contribute differently to the final result, so their rewards should be different. This is also similar to the law of our human behavior. When we are about to complete a task, we are at a critical moment, and continuing to choose the right behavior can easily ensure that the task is completed successfully. However, when we are just starting a task, our behavior is usually not directly related to the final result, because there is still a long way to go. Furthermore, we assume that usually shorter reasoning paths are more efficient. Our length factor enables the model to learn more from short inference paths. Our reward is simpler and does not use the reward of diversity. Although the reward of diversity can help find different paths, the randomness of the learning process will also increase and the stability will decrease.

The semantics of an entity usually includes many aspects, and its neighbors can provide more sufficient description of the attributes of the entity. Bansal et al. [10] propose an attention mechanism to learn query-dependent representations of entities. We integrate a graph self-attention (GSA) mechanism to incorporate more comprehensive neighborhood representations. The underlying assumption is that the information combination based on the similarity between the neighborhood entities and the central entity may not make full use of the potential of the neighborhood. Neighborhood entities may contain more diverse context information. Our method can adaptively weight neighbors based on their context.

The relations in an inference path are ordered. Most KG completion methods only consider the direct relation between entities, rarely considering the entire relation path. If taking action based only on its current position, the agent may miss some historic action experience. For example, when the agent reaches a specific position from different paths, the agent will choose the next action with the same probability every time, so the remaining paths will easily overlap. Zhu et al. [11] project the head and tail entities into different spaces to model the relation order. Wang et al. [12] use LSTM [13] to consider the memory of relations in the path. We adopt GRU-RNN [14] to model path information because GRU uses fewer training parameters and can speed up training and inference. We incorporate GSA mechanism with GRU (GSA-GRU) to model relation order and keep the historical memory in the path. Different from the previous work, we only use the hidden representation of GRU to predict actions. The advantage is that the model is simpler and we can observe the effectiveness of GRU more clearly.

1.1. Contribution

We found that the proposed model can mine more balanced reasoning paths for each relation. By controlling the distance-aware factor, we can make the model mine more reliable paths. Our method can also find some commonsense reasoning paths and use common-sense knowledge for reasoning, such as teammates belonging to the same organization. Our model eliminates the pre-training process, but finds more valuable paths. The distance-aware factor can be potentially used in other reinforcement learning scenarios, such as robot path planning.

We conduct experiments on two well-known datasets, NELL-995 and FB15K-237. Experimental results demonstrate the effectiveness of the method. The major contributions are shown below:

(i) We propose a distance-aware reward. This allows the agent to get different rewards when they are at different positions.

(ii) We incorporate the GSA mechanism with GRU in the policy network to model the sequence information of the path.

(iii) Our RL framework is simple and can be trained in one-pass without the pre-training or fine-tuning issues.

1.2. Organization

The paper is organised as follows. Section 1 provide the introduction. Section 2 describe the related works about rule-based methods, embedding-based methods, path-based methods, etc. Section 3 provide the detailed approach for this paper. Section 4 explain about the experiment results followed by experimental setting in subsection 4.1, results of link prediction in subsection 4.2, results of fact prediction in subsection 4.3, ablation study in subsection 4.4, and case study in subsection 4.5. Finally conclusion is given in Section 5.

2. Related Work

For the rule-based methods, Wang et al. [15] propose to learn the first-order logic embeddings for probabilistic inference. Yang et al. [16] propose the Neural Logic Programming that can learn the first-order logical rules in an end-to-end differentiable fashion. For the Embedding-Based methods, prior works, such as TransE [17], map the head and tail entities and relations into a low-dimensional continuous vector space and then compare the distance between $||t-h||$ and $||r||$ [18]. There are challenges when dealing with one-to-many and many-to-one relations. To resolve this problem, TransH [19], TransR [20], TransD [21], TansSparse [22], TransG [23], TransA [24], KB2E [25], etc. are proposed. This type of methods are more suitable for the single-hop reasoning. Tensor decomposition models [26], such as DistMult [27], ComplEx [28], Analogy [29],

SimplE [30], HolE [31], TuckER [32], and deep learning models, such as ConvE [33], ConvKB [34], ConvR [35], CapsE [36], RSN [37], are also used to represent entities and relations.

For the path-based method, Lao et al. [38, 39] propose the path-ranking algorithm to resolve the link prediction task. Neelakantan et al. [40] propose to use Path-RNN to input entities in the path between two entities into RNN to infer the relation. Das et al. [41] propose Single-Model that not only considers the entities in the path, but also considers the relations in the path, thereby enhancing the expressive capacity of the neural network. Xiong et al. [1] propose to use RL framework to mine the reasoning paths in KG. They propose the diversity metric to enable the agent to find more different paths. Besides, this approach needs to pretrain the model based on supervised learning. Das et al. [42] propose an RL model that can learn to navigate the graph conditioned on the input query to find predictive paths without the need for precomputed paths. Lin et al. [43] enhance the pathfinding process by using action dropout and reward shaping. Chen et al. [44] propose a variational inference framework that integrates the path-finding and path-reasoning processes in a probabilistic framework. Shen et al. [45] tackle the problem of sparse rewards by iteratively refining the policy using a Monte Carlo Tree Search (MCTS) method.

Many neighborhood or graph based models are proposed and achieve promising performance. Zhu et al. [11] address the problem of relation orders in paths by projecting the head entity and the tail entity of each relation into different spaces. Lv et al. [46] learn the meta-learning parameters from high-frequency relations and quickly adapt to few-shot relations. Bansal et al. [10] propose a graph attention model for learning query-dependent entity embeddings based on graph neighborhood. Modeling graph neighborhood [47, 48, 49] has proven effective to enhance the entity representation. Wang et al. [12] incorporate the graph attention mechanism with the LSTM[50] units to alleviate the pretraining problem. They invent two metrics, MSR and MRR, to measure the learning difficulty of relations, and better fine-tune the model. They also adopt the diversity metric, query-dependent entity embeddings and TransD-based representation mechanisms [21]. Based on the above studies, we propose the DAPath model. Different from them, our method is trained in one-pass without the pre-training or fine-tuning process.

3. Approach

First, we describe the task formulation and define some notations. Suppose we have an incomplete knowledge graph $\mathcal{G} = \{(h, r, t)|h, t \in \mathcal{E}, r \in \mathcal{R}\}$, where $\mathcal{E}$ and $\mathcal{R}$ represent the entities and relations respectively. The link prediction task aims to infer the tail entity t as the answer for the question of $(h, r_q, ?)$ while the fact prediction task aims to evaluate whether an unknown fact (h, r_q, t) holds or not. Different from the existing relation r, $r_q \notin \mathcal{R}$ is the query relation that does not exist in KG. The h and t are not directly connected, instead, there is a long inference path ($h \xrightarrow{r_1} ent_1 \xrightarrow{r_2} ent_2 ... \xrightarrow{r_m} t$) from h to t. ent_i is the i-th entity in the path. Our goal is to learn a model that can resolve the path-finding problem, that is, let

the model automatically mine the reasoning paths for the pair of h and t. Recently, RL-based reasoning has achieved impressive performance on this task. Under the framework of RL, the path-finding problem is formulated as a Markov decision process (MDP).

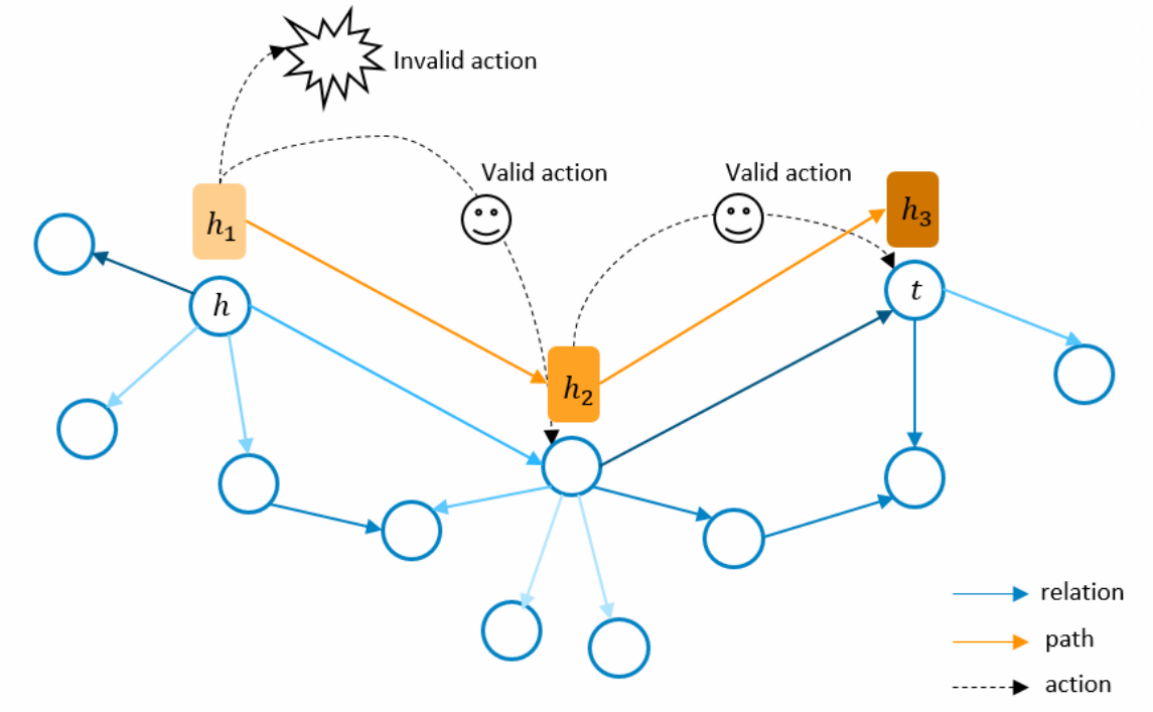


Figure 1: DAPath framework overview, where the weight of the color represents the weight of the position in the reasoning path

Our main effort lies in proposing a framework that integrates three portable components to augment the RL-based reasoning, that is, the distance-aware (DA) reward, the graph self-attention mechanism and the GRU-based relation sequence modeling. The framework overview is shown in Figure 1. Blue vertices and blue lines represent entities and relations in this KG. h_i denotes the hidden state of the agent at the i-th step. The orange line indicates the path found. The weight of the color represents the weight of the position in the reasoning path. The black dotted line indicates the actions taken by the robot. After selecting a valid action, the robot will move forward, otherwise, the robot will stay at the origin and be punished. In the following, we will describe the framework, the policy network and the training method.

3.1. RL framework for KG reasoning

We first introduce basic elements of the RL framework in the KG reasoning, including the external environment, state, action, reward and optimization. As shown in Figure 2, the input of the agent is a state composed of the current entity and the tail entity. The output is the next relation predicted by the agent.

It is a trial and error process for an agent to find reasoning paths. The application of reinforcement learning in knowledge graph reasoning is based on the assumption that as long as the agent can reach the tail

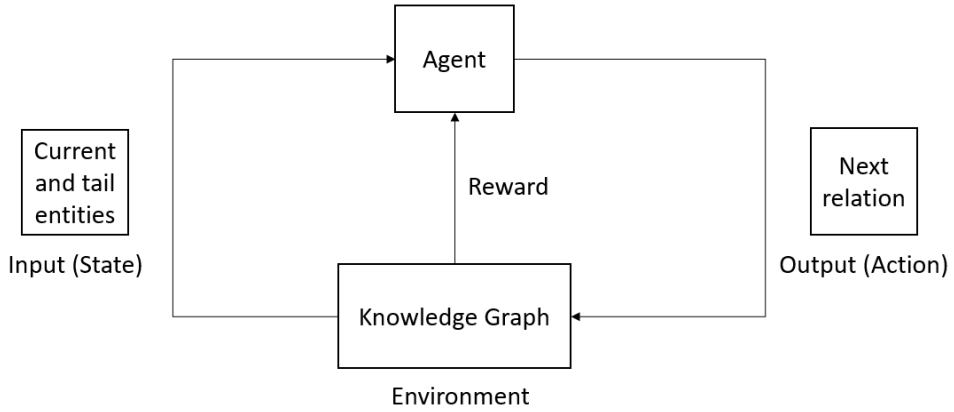


Figure 2: Schematic diagram of RL framework for KG reasoning

entity t from the head entity h within a certain number of steps, we can consider this path as a potential reasoning path. DeepPath [1] introduces the reinforcement learning framework to the knowledge graph reasoning for the first time. The main task is to find the path from the head entity to the tail entity in the knowledge graph. This approach simply samples the knowledge graph, trains the policy network, and retrains the policy network through a manually designed reward function. The agent performs a multi-hop reasoning task. Every time the agent takes an action, a state transition occurs and the corresponding reward generated. The pathfinding process is consistent with the concept of episode in reinforcement learning. Next, we will introduce each component in detail.

Environment The entire KG is considered an experimental environment for an agent, excluding query relations. This environment will remain the same throughout the training process. This environment defines the dynamics of the interaction between the agent and the KG. For example, by interacting with KG, the agent will transition to a new state.

State The state encodes the location information of the agent in the KG with a fixed-length vector. The state vector at i-th step is composed of three parts.

$$s_i = [e_i; e_t - e_i; e_a] \tag{1}$$

where e_i and e_t are the embeddings of the current entity and the target entity. e_a denotes the graph neighborhood representation which will be described in subsection 3.2.2. $[;]$ denotes the concatenation operation. e_0 is initialized as the embedding of the head entity.

Action Our model considers each relation type as an action. We define the action space $\mathcal{A} = \{a_1, a_2, ..., a_d\}$, where $a_i \in \{0, 1\}$ indicates whether to take the i-th action with 1 indicating the positive label and 0 otherwise. Starting from the head entity, the agent uses the policy network to take the most promising action in the current state, thereby extending the path until it reaches the tail entity. The policy function maps the state

vector s_t to a probability distribution over all possible actions.

$$\pi(s_i, a_i; \theta) = p(s_i|a_i; \theta) \tag{2}$$

where $p(s_i|a_i; \theta)$ is calculated by a neural network. θ denotes the model parameters.

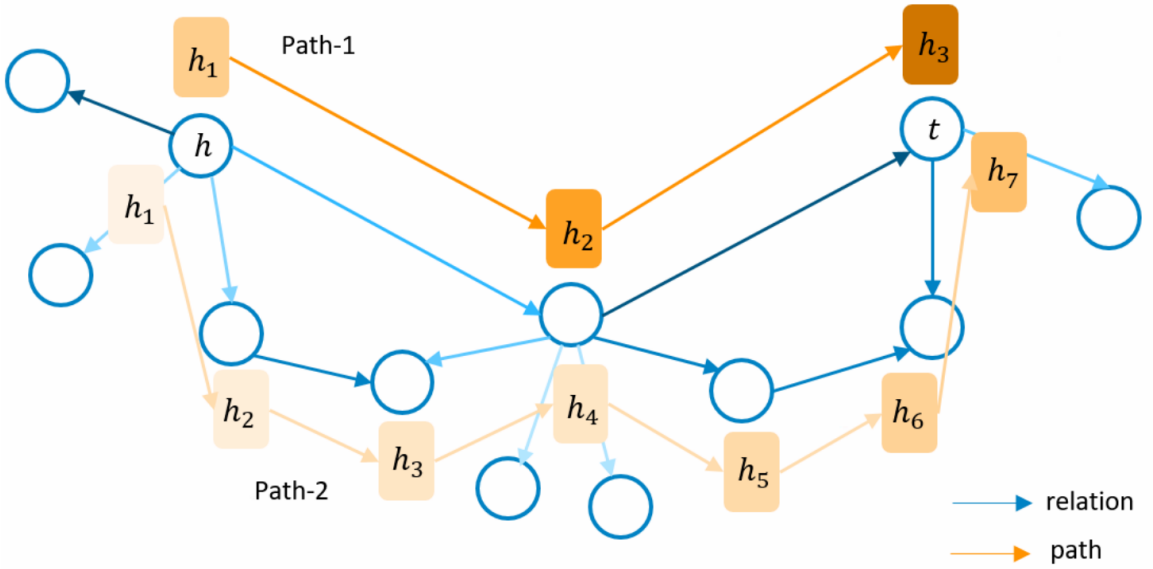


Figure 3: Schematic diagram of reasoning path, where the weight of the color represents the weight of the position in the reasoning path

Reward The reward function is an indicator of the effectiveness of the chosen actions. Most prior studies assume that the model only has a final reward when it finishes an episode. In addition to the final reward, we assume that the actions taken at different positions have different rewards. We name it the distance-aware (DA) reward. The intuition is that this reward is concerned with the path length and distance from the head entity. As shown in Figure 3, the weight of path-1 is generally higher than the weight of path-2. Although both paths work, the model is more focused on learning from the path 1. Actions taken at the positions close to the target entity have more influence than actions taken earlier.

Our DA reward is composed of two parts which are the distance-aware factor and the global reward. The DA reward of the i-th step is shown below.

$$r_i^{(da)} = f(i, n) \cdot r_{global} \tag{3}$$

where $f(i, n)$ is the distance-aware factor defined in equation (4). n is the total length of the path. r_{global} is the global reward defined in equation (7).

$$f(i, n) = p(i) \cdot g(n) \tag{4}$$

The DA factor is composed of the hierarchical factors of path length factor $g(\cdot)$ and position factor $p(\cdot)$. We visualize the trend of the DA factor under different parameter settings as shown in Figure 4. The x- and y- axes represent the distance to the head entity and the reward factor. The bars represent the DA factor at different positions. The red and blue lines indicate the trend of the position factor and length factor, respectively. Different colored bars indicate the maximum length of different paths (Length-n).

We observe that we can set the trend of the factor flexibly by controlling two parameters k_2 and τ. k_2 can control the magnitude of the length factor. τ can control the relationship between path length and path importance. Formula (5) is the position factor.

$$p(i) = k_1 \ln i + b \tag{5}$$

where k_1 and b are hyperparameters. We set $k_1 = 1.0$ and $b = 1.0$. Formula (6) is the length factor.

$$g(n) = k_2[\sum_{i=1}^{n} p(i)]^{\tau} \tag{6}$$

where k_2 and τ are hyperparameters. We set $k_2 = 5.0$ and $\tau = -0.6$. When $\tau < 0$, formula (6) is a monotonically decreasing function (blue curve), and when $\tau > 0$, formula (6) is a monotonically increasing function. The intuition is that in most cases, as i increases, the logarithmic function grows more slowly than the power function. Therefore, we can combine them to get a downward trend to reduce the impact of long reasoning paths on model learning. Equation (6) can describe the relationship between the path length and the accumulation of each step of equation (5).

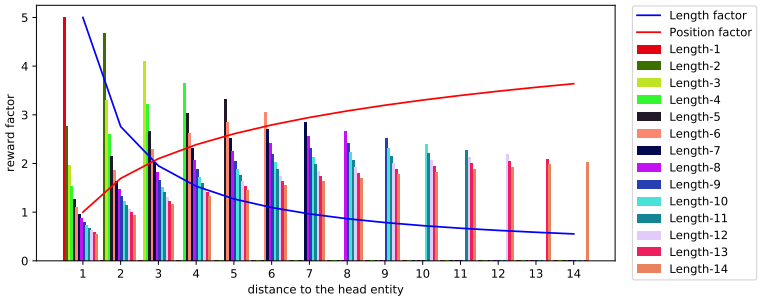
The global reward is defined as follows.

$$r_{global} = \begin{cases} +1 & \text{if the path reaches } t \\ -1 & otherwise \end{cases} \tag{7}$$

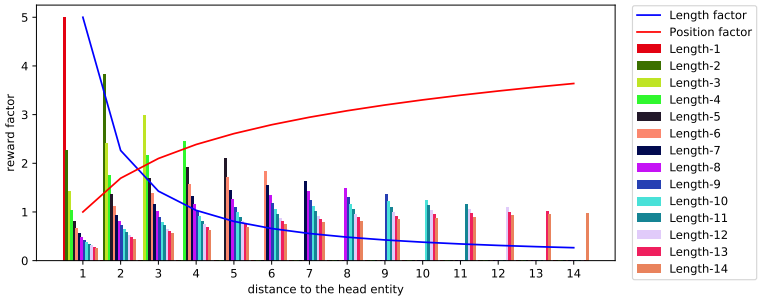
When the agent successfully finds a valid path, the agent will receive a positive reward, otherwise, it will be punished. Receiving a positive reward will make the agent optimize the model in the direction of the gradient ascent during the learning process, so the agent will increase the possibility of choosing these actions next time, and thus tend to choose this strategy with a higher probability. Conversely, receiving a negative reward will optimize the model in the direction of gradient descent during the learning process, the agent will reduce the possibility of choosing such actions next time, and thus tend not to choose this strategy. Through the above-mentioned trial-and-error process [51], the robot has completed autonomous learning.

Optimization We aim to maximize the expected total reward. More formally, our objective function is defined as

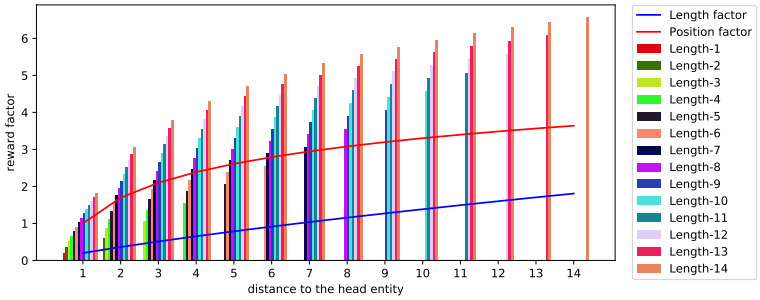
$$J(\theta) = \mathbb{E}_{s_1,a_1,s_2,\ldots,s_i,a_i,\ldots}[R(a_i, s_i; \theta)] \tag{8}$$



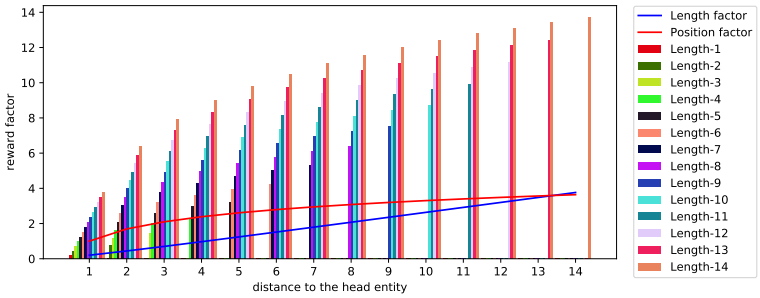
(a) $k_2 = 5.0, \tau = -0.6$



(b) $k_2 = 5.0, \tau = -0.8$



(c) $k_2 = 0.2, \tau = 0.6$



(d) $k_2 = 0.2, \tau = 0.8$

Figure 4: Distance-aware factor distribution with different τ

where $R(a_i, s_i; \theta)$ is the expected total rewards for an episode defined in equation (9). $a_i \sim \pi(a_i, s_i; \theta)$ is the probability of all relations. $s_{i+1} \sim p(s_{i+1}|s_i; \theta)$ is the transition function that is equal to 1, since the state s_{i+1} is fully determined by the state s_i and a_i.

$$R(a_i, s_i; \theta) = \sum_i r_i^{(da)} \log \pi(a_i, s_i; \theta) \tag{9}$$

According to the policy gradient theorem [52] and the Monte-Carlo Policy Gradient (REINFORCE algorithm) [53], we use the following equation to calculate the gradient.

$$\nabla_\theta J(\theta) \approx \nabla_\theta \sum_{i=1}^{t} r_i^{(da)} \log \pi(a_i, s_i; \theta) \tag{10}$$

Then we can update our model parameters.

$$\theta \leftarrow \theta + \alpha \nabla_\theta J(\theta) \tag{11}$$

where α is the learning rate.

3.2. Policy Network

3.2.1. GSA-GRU

We adopt GSA-GRU in the policy network, enabling the agent to retain its experience and memory. The agent can take the most promising action.

$$h_i = GRU(h_{i-1}, [e_i; e_t - e_i; e_a]) \tag{12}$$

where h_i denotes the i-th hidden state of the path. Defined in equation (17), e_a is the graph neighorhood representation. h_0 is initialized to a zero vector. The hidden state h_i of GRU are shown as follows.

$$h_i = z_i \odot h_{i-1} + (1 - z_i) \odot \phi_h(W_h[e_i; e_t - e_i; e_a] + U_h(r_i \odot h_{i-1}) + b_h) \tag{13}$$

where z_i is the update gate vector. $\phi_h(\cdot)$ denotes the hyperbolic tangent function. W and U are linear composition matrices. b is the bias parameter.

$$z_i = \sigma_g(W_z[e_i; e_t - e_i; e_a] + U_z h_{i-1} + b_z) \tag{14}$$

where $\sigma_g(\cdot)$ denotes the sigmoid function.

$$r_i = \sigma_g(W_r[e_i; e_t - e_i; e_a] + U_r h_{i-1} + b_r) \tag{15}$$

where r_i is the reset gate vector. Finally the agent take an action through single layer neural network.

$$\hat{y}_i = \text{softmax}(tanh(W h_i + b)) \tag{16}$$

where W and b are weight and bias parameters.

3.2.2. Graph Neighborhood Representation

The information of the neighborhood entities is represented in following equation.

$$e_a = \sum_{i \in N(e_c)} \alpha_i \cdot W_V u_i \tag{17}$$

where W_V is the weight parameters. $N(e_c)$ denotes the neighbors' index of the central entity e_c. Defined in equation (18), u_i denotes the representation of the i-th entity and relation connected with the central entity. Defined in equation (19), α_i is the weight of the i-th neighbor. k is the number of neighboring entities.

$$u_i = [e_i; r_{c,i}], i \in N(e_c) \tag{18}$$

where $r_{c,i}$ is the relation between e_c and the i-th neighboring entity.

$$\alpha_i = \frac{\exp\left(q^T W_K u_i\right)}{\sum_{j=1}^{k} \exp(q^T W_K u_j)} \tag{19}$$

where W_K is the weight parameters. Defined in equation (20), q is the query vector.

$$q = \sum_{i \in N(e_c)} W_Q u_i \tag{20}$$

where W_Q is the weight parameters. The above process is also known as the single-head attention which is highly extensible, because adding more heads will generate multi-head attention as follows.

$$e_a = [e_{a,1}; e_{a,2}; ...e_{a,m}] \tag{21}$$

where m is the number of heads. $e_{a,k}$ represents the k-th head.

3.3. Training method

To optimize the policy network for pathfinding, we use the REINFORCE algorithm [53]. This method prefers to take action that can get a higher reward. We use gradient descent to maximize the expected total rewards in equation (8) for an episode. We directly train our model based on the agent's own pathfinding process. Our approach does not need to pre-train or fine-tune the model, which significantly reduces the complexity of the problem.

Besides, when the agent chooses an invalid action, we not only punish the agent but also force the agent to choose a valid action to move forward every step [12], avoiding the agent getting stuck on the same vertex, as shown in Figure 1. Formally, suppose the agent is at the position e_c and predicts an action $a_i \notin r_c$ where r_c denotes the valid actions (relations connected to e_c). The agent will resample an action from the valid actions r_c based on the predicted probability. If there are multiple valid entities after taking an action, the agent will randomly select an entity from them. This forcing operation is like the "eyes" of the agent, allowing the agent to look at the path before walking. Then, we use all the mined paths as binary features to train a single-layer neural network to calculate a score for each candidate t of (h,r,?).

4. Experiments

4.1. Experimental Settings

4.1.1. Dataset

NELL-995 dataset[2] is developed by Xiong et al. [1] based on the NELL[3] system [54]. The statistics are shown in Table 1. This dataset contains 154,213 triples, e.g. *(directorColinHanks, parentOfPerson, personTomHanks)*, *(cityYork, cityLiesOnRiver, riverMohawkRiver)*. The dataset contains 12 tasks, e.g. *athletePlaysForTeam*, *personLeadsOrganization*.

FB15K-237 dataset is developed by Toutanova et al. [55] sampled from FB15K [17]. The statistics are shown in Table 1. This dataset contains 310,116 triples, e.g. *(/m/0411q, /people/person/profession, /m/016z4k)*, *(/m/09c7w0, /location/location/contains, /m/0rs6x)* where the entities and relations are represented by Universal Resource Identifiers (URI). We conduct experimental on 20 tasks, e.g. *capitalOf*, *filmDirector*.

Table 1: Statistics of the datasets

Dataset	Entities	Relations	Triples	Tasks
NELL-995	75,492	200	154,213	12
FB15K-237	14,505	237	310,116	20

4.1.2. Training and Hyperparameters

We conduct experiments based on pre-trained TransE [17] KG embeddings. We set the hidden size of the GRU unit to 1024D. For regularization, we apply L_2 regularization with $lambda = 0.01$. We set the maximum depth to 50. We use 4 heads in the multi-head attention. We use the Adam optimization algorithm to update the model parameters with the learning rate of 0.001. We conduct experiments on an Intel(R) Xeon(R) CPU E5-2630 v4 @ 2.20GHz (Mem: 251G) and the GPU TITAN RTX (24G).

4.1.3. Evaluation

The metrics used to evaluate the quality of our model are Mean Average Precision (MAP), Mean Reciprocal Rank (MRR) of all correct entities and the proportion of correct entities that rank no larger than N (Hits@N). These metrics are widely used to evaluate the ranking results. For link prediction, our objective is to rank the candidate target entities based on their confidence to (h,r,?). For fact Prediction, instead of ranking the target entities, this task directly ranks all the positive and negative triples for a particular relation.

[2] https://github.com/xwhan/DeepPath

[3] http://rtw.ml.cmu.edu/rtw/

4.2. Results of Link Prediction

We compare with path-based method DeepPath [1] and embedding-based methods TransE [17] and TransR [20]. Table 2 lists the MAP scores of the link prediction task on two datasets. Our DAPath[4] achieves improvement on the overall MAP score, as shown in the last row of the table. Wang et al. [12] also improve the DRL-based method, but they use different metrics to fine-tune the model and the representation mechanism is different from our experiments. Therefore, we did not make a quantitative comparison with them. We only train this model in one-pass and adopt the representation mechanism of DeepPath. We observe that the DAPath generally achieves higher results on different relations. However, TransR achieves a higher results on "teamPlaySports", "bornLocation", "filmWrittenBy" relations. Because the process of forcing the agent to move forward may produce some low-quality paths, it is important to improve the quality of this forcing process. DeepPath achieves higher results on "athleteHomeStadium", "birthPlace", "tvLanguage" etc. This is because they did not force the agent to move forward and the pretraing process can help alleviate the problem of low-quality paths.

Table 2: MAP results of link prediction on two datasets

NELL-995					FB15K-237				
Tasks	DeepPath	TransE	TransR	**DAPath**	Tasks	DeepPath	TransE	TransR	**DAPath**
athletePlaysForTeam	0.750	0.627	0.673	**0.762**	organizationFounded	0.309	0.390	0.339	**0.475**
athletePlaysInLeague	0.960	0.773	0.912	**0.965**	birthPlace	**0.531**	0.403	0.417	0.519
athleteHomeStadium	**0.890**	0.718	0.722	0.887	personNationality	0.823	0.641	0.720	**0.841**
athletePlaysSport	0.957	0.876	0.963	**0.965**	filmDirector	0.441	0.386	0.399	**0.447**
teamPlaySports	0.738	0.761	**0.814**	0.789	filmWrittenBy	0.457	0.563	**0.605**	0.579
orgHeadquaterCity	0.790	0.620	0.657	**0.793**	filmLanguage	0.670	0.642	0.641	**0.696**
worksFor	0.711	0.677	0.692	**0.728**	tvLanguage	**0.969**	0.804	0.906	0.962
bornLocation	0.757	0.712	**0.812**	0.768	capitalOf	0.783	0.554	0.493	**0.829**
personLeadsOrg	**0.795**	0.751	0.772	**0.795**	teamSports	**0.955**	0.896	0.784	0.903
orgHiredPerson	0.742	0.719	0.737	**0.745**	musicianOrigin	**0.514**	0.361	0.379	0.446
...					...				
Overall	0.796	0.737	0.789	**0.806**	Overall	0.572	0.532	0.540	**0.586**

Table 3 lists the MRR and Hits@N results of different models. On the two metrics, path-based deep learning methods (Single-Model, MINERVA, DeepPath, DAPath, BiLSTM-CNN-Att, etc.) usually achieve better results than embedding-based methods. This is because the pathfinding mechanism helps to mine multi-hop reasoning features. It can be seen from Hits@N scores that the path-based methods are easier to rank the correct entities in higher positions. However, RL-based methods need to train different models for each task. BiLSTM-CNN-Att achieves the highest results, but this method needs to use a graph search

[4] `https://github.com/prayagtiwari/DAPath`

algorithm to find and preserve the high quality paths between two entities. Our method does not use graph search algorithms, but learns completely in the process of random exploration.

Table 3: MRR and Hits@N results on two datasets

Dataset	NELL-995				FB15K-237			
Metrics	MRR	Hits@10	Hits@3	Hits@1	MRR	Hits@10	Hits@3	Hits@1
Auto KGE [56]	–	–	–	–	0.360	0.552	–	0.267
PRA [38]	0.696	–	0.747	0.637	0.412	–	0.331	0.322
Meta-KGR(DistMult) [46]	0.248	0.345	–	0.197	0.458	0.580	–	0.403
Meta-KGR(ConvE) [46]	0.253	0.347	–	0.197	0.469	0.588	–	0.412
KBAT [57]	0.530	0.695	0.564	0.447	0.518	0.626	0.540	0.460
CapsE [36]	–	–	–	–	0.523	0.593	–	–
GAATs [58]	–	–	–	–	0.525	0.637	0.550	0.494
DistMult [27]	0.863	–	0.907	0.801	0.541	–	0.554	0.413
ConvE [33]	0.909	–	0.929	0.904	0.567	–	0.629	0.444
Single-Model [41]	0.859	–	0.914	0.788	0.575	–	0.567	0.512
MINERVA [42]	0.879	–	0.931	0.813	0.615	–	0.659	0.490
DeepPath [1]	0.838	0.973	0.815	0.726	0.642	0.951	0.655	0.461
BiLSTM-CNN-Att [59]	**0.898**	–	0.951	0.838	**0.660**	–	0.708	0.544
DAPath	0.851	**0.976**	0.837	0.737	0.647	0.95	0.663	0.48

Table 4 lists the detailed results of the comparison between the two models on the NELL-995 dataset. The DAPath model achieves better results in most cases, while the DeepPath model achieves better results in "athletePlaysInLeague", "orgHeadquaterCity" and "orgHiredPerson" relations. Table 5 lists the detailed results of the comparison between the two models on the FB15K-237 dataset. The DAPath model achieves better results in most cases.

4.3. Results of Fact Prediction

Fact prediction (FP) aims to evaluate whether an unknown fact is true or false. We compare with DeepPath [1], TransE [17], TransH [19], TransR [20] and TransD [21]. Table 6 shows MAP scores of the fact prediction task. We observe that DAPath achieves improvements.

We analyze the distribution of reasoning paths learned by DApath and DeepPath, as shown in Figure 5. The x- and y- axes represent the path length and the number of paths respectively. We observe that DAPath tends to learn more short paths than DeepPath. This is because our DA reward mechanism enables the agent to emphasize the short paths. For example, the agent of DAPath found more than 30 possible paths of

Table 4: MRR and Hits@N results of link prediction on the NELL-995 dataset

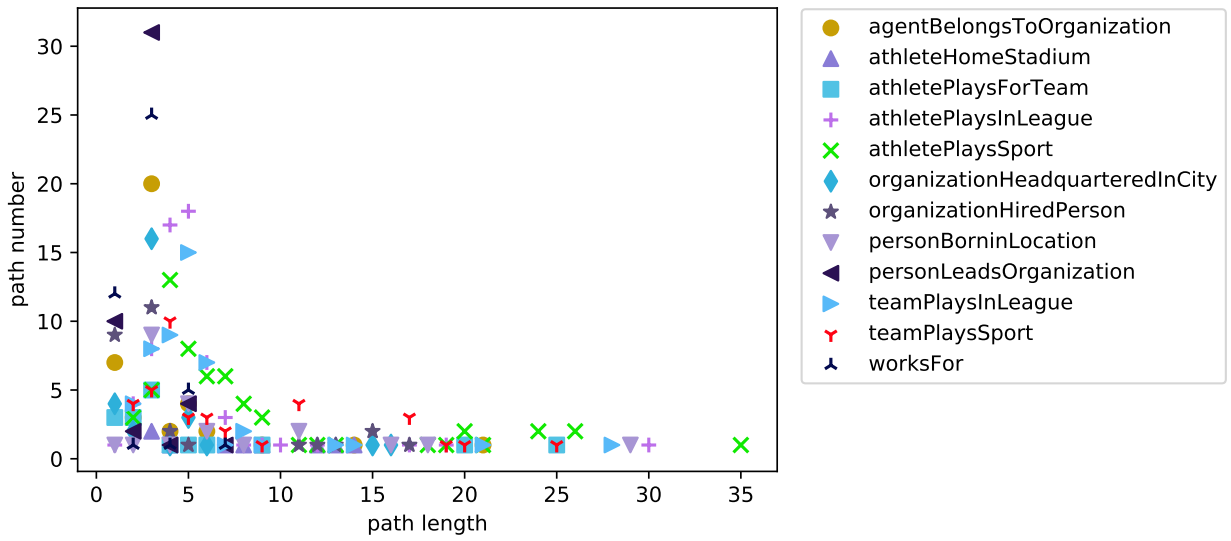
Datasets	DeepPath				DAPath			
Tasks	MRR	Hits@10	Hits@3	Hits@1	MRR	Hits@10	Hits@3	Hits@1
athletePlaysForTeam	0.712	0.993	0.744	0.601	**0.762**	0.996	0.826	0.654
athletePlaysInLeague	**0.966**	0.992	0.985	0.948	0.963	1	0.988	0.936
athleteHomeStadium	0.850	0.976	0.823	0.815	**0.887**	0.976	0.873	0.862
athletePlaysSport	0.963	1	0.983	0.943	**0.965**	1	0.986	0.943
teamPlaySports	0.781	0.96	0.94	0.644	**0.786**	1	0.87	0.673
orgHeadquaterCity	**0.936**	1	0.947	0.772	0.932	0.983	0.936	0.772
worksFor	0.731	0.955	0.651	0.555	**0.765**	0.965	0.749	0.574
bornLocation	0.732	0.948	0.729	0.646	**0.771**	0.974	0.835	0.657
personLeadsOrg	0.817	0.971	0.765	0.652	**0.826**	0.963	0.76	0.67
orgHiredPerson	**0.822**	0.935	0.726	0.652	0.809	0.913	0.725	0.633
...								
Overall	0.838	0.973	0.815	0.726	**0.851**	0.976	0.837	0.737

length 5 for "personLeadsOrganizatio". The agent of DeepPath only finds 10 paths. DAPath also can find some longer paths. This is because when our model finds a long path, the positions close to the target entity also have a high weight. Therefore, the model does not directly discard some relatively long possible paths.

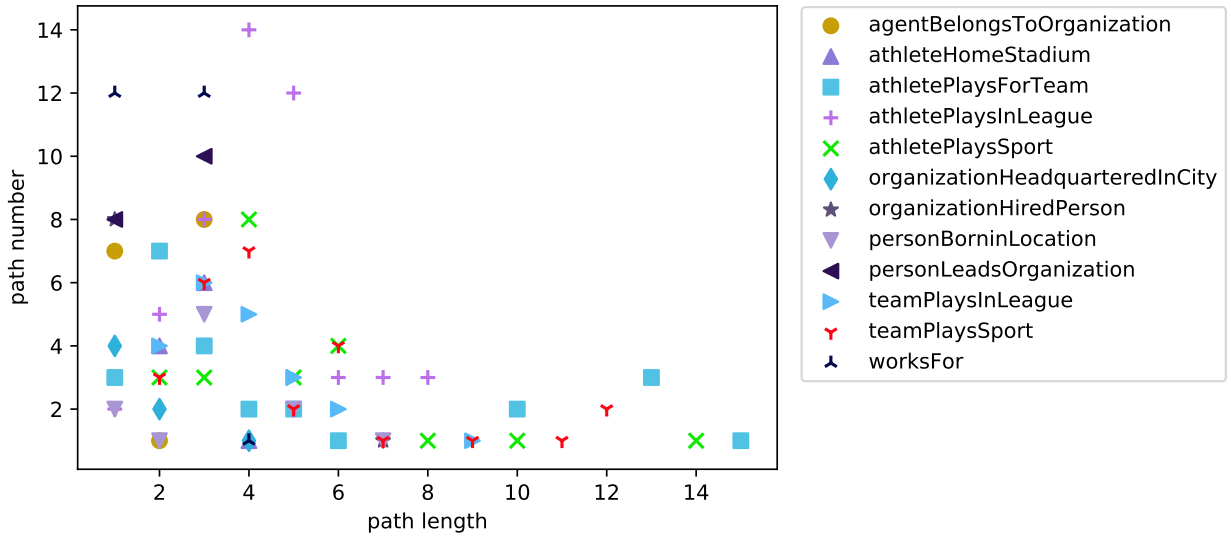
Figure 6 visualize the path length (x-axis) and the number of paths (y-axis) respectively. We observe the distribution of different length paths mined by our model is more uniform. The baseline model only mines more paths for certain relations, e.g. "tvProgramCountryOfOrigin" and fewer paths for others. Our model can mine richer reasoning paths for most relations. This means our model can learn more balanced paths on a large KG.

4.4. Ablation Study

Table 7 shows the ablation studies under different model settings. A first observation is that the three components (GRU, GSA, DA) are essential for good results. Removing one of them slightly degrades performance. This indicates that these three components help improve final results. Since the DA-reward is composed of the hierarchical factors of path length and positions, the influence of DA is greater, and we have the flexibility to adjust this factor. This factor can be widely used in other reinforcement learning scenarios, such as robot path planning. When we remove all three components, the model can still achieve comparable results with DeepPath. However, we train our model in one-pass, eliminating the pre-training process and cutting the training time in half. Our method eliminates the problem of finding paths using heuristic search,

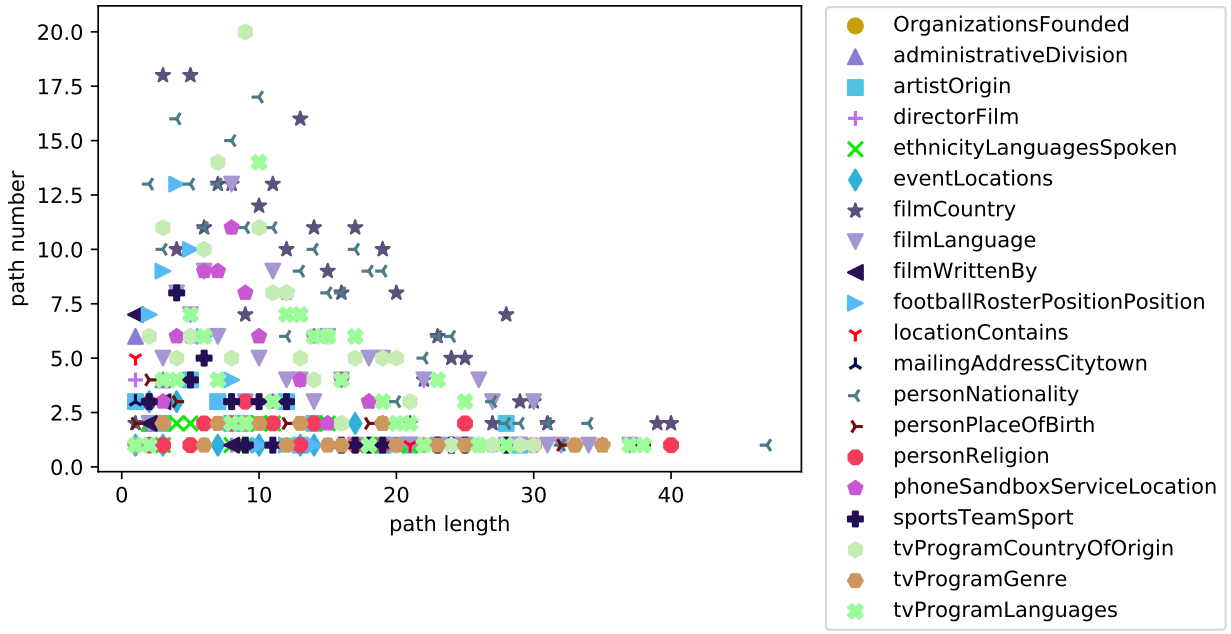


(a) DAPath

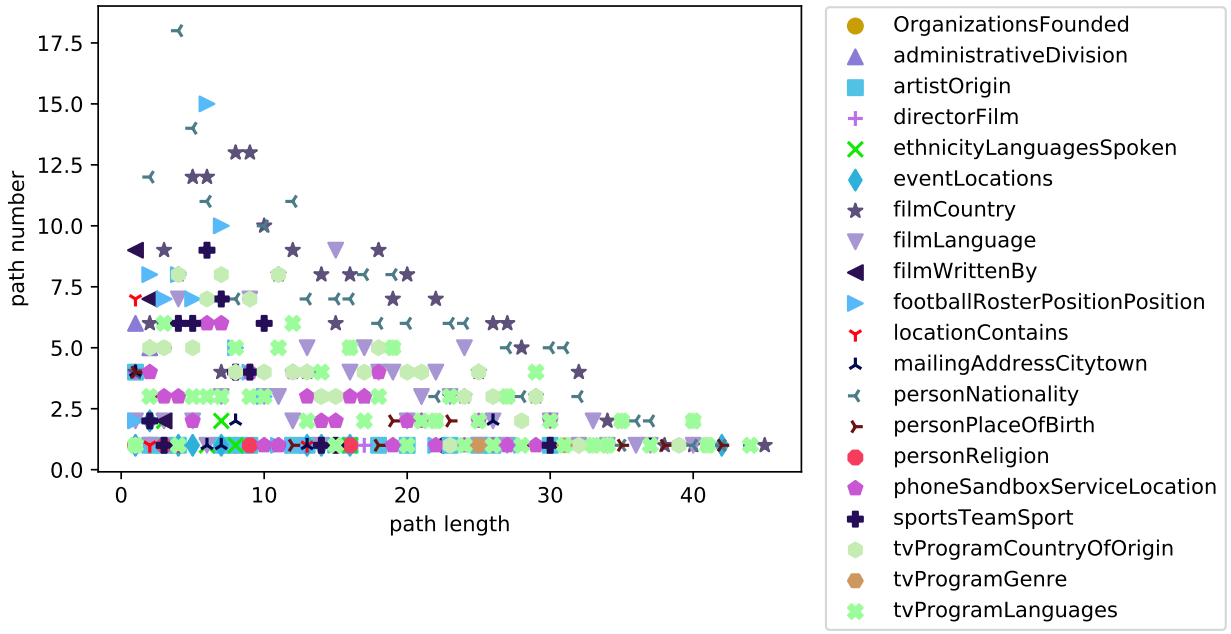


(b) DeepPath

Figure 5: Visualization of the length-number of reasoning paths of the NELL-995 dataset



(a) DAPath



(b) DeepPath

Figure 6: Visualization of the length-number of reasoning paths of the FB15k-237 dataset

Table 5: MRR and Hits@N results of link prediction on the FB15K-237 dataset

	DeepPath				DAPath			
Tasks	MRR	Hits@10	Hits@3	Hits@1	MRR	Hits@10	Hits@3	Hits@1
organizationFounded	0.338	0.895	0.267	0.058	**0.483**	0.907	0.477	0.163
birthPlace	**0.526**	0.982	0.588	0.36	0.520	0.984	0.575	0.358
personNationality	0.835	0.995	0.901	0.736	**0.847**	0.996	0.902	0.756
filmDirector	0.401	0.908	0.404	0.089	**0.435**	0.943	0.469	0.167
filmWrittenBy	0.569	0.977	0.564	0.384	**0.579**	0.964	0.621	0.404
filmLanguage	0.707	1	0.973	0.957	**0.709**	0.976	0.711	0.571
tvLanguage	**0.970**	1	0.973	0.957	0.968	0.995	0.967	0.967
capitalOf	0.815	0.982	0.738	0.524	**0.825**	0.97	0.86	0.695
teamSports	**0.952**	1	0.94	0.879	0.946	1	0.948	0.853
musicianOrigin	**0.499**	0.971	0.575	0.33	0.465	0.951	0.48	0.287
...								
Overall	0.642	0.951	0.655	0.461	**0.647**	0.95	0.663	0.48

thus enabling complete autonomous learning. Therefore, our method is suitable for processing large-scale knowledge graph.

4.5. Case Study

It is instructive to analyze the reasoning paths. We hand-picked 4 relations in the two datasets. Table 8 shows the top 8 frequent paths. For the first relation, our approach is more likely to find direct and confident relations such as "film/producedBy → person/nationality".

For the second relation, in addition to some high probability paths such as "artist/origin" and "place-

Table 6: Fact prediction results on two datasets

Methods	NELL-995	FB15K-237
TransE	0.383	0.277
TransH	0.389	0.309
TransR	0.406	0.302
TransD	0.413	0.303
Deeppath	0.493	0.311
DAPath	**0.528**	**0.328**

Table 7: Model setting ablations on NELL-995 dataset

Task (MAP)	Link prediction	Fact prediction
DAPath	0.806	0.528
–GRU	0.796	0.508
–GSA	0.799	0.511
–DA	0.794	0.502
–GRU, GSA	0.792	0.499
–GRU, GSA, DA	0.789	0.491

Table 8: Top 8 frequent reasoning paths found for four relations using two methods where "-1" indicates inverse relation

ID	Relaton	DAPath	DeepPath
1	filmCountry	filmReleaseRegion netflixGenre/titles^{-1} filmFilmDistributorRelationship/region featuredFilmLocations film/producedBy → person/nationality film/music → person/nationality regularTVAppearance/actor → person/nationality featuredFilmLocations → bibsLocation/country	filmReleaseRegion netflixGenre/titles^{-1} awardWinner → person/nationality awardNomination/nominatedFor^{-1} → filmReleaseRegion film/prequel → filmReleaseRegion performance/film^{-1} → ethnicity/people^{-1} → ethnicity/geographicDistribution filmCrewRole → filmCrewRole^{-1} → filmReleaseRegion film/language → tvProgram/languages^{-1} → tvProgram/countryOfOrigin
2	personNationality	artist/origin placeLived/location → location/contains^{-1} person/placeOfBirth → bibsLocation/country ethnicity/people^{-1} → gardeningHint/splitTo performance/film → film/country deceasedPerson/placeOfDeath → bibsLocation/country ethnicity/people^{-1} → ethnicity/geographicDistribution deceasedPerson/placeOfDeath → location/contains^{-1}	person/placeOfBirth person/placeOfBirth → bibsLocation/country placeLived/location → bibsLocation/country person/placeOfBirth → location/contains^{-1} placeLived/location → administrativeDivision/country awardWinner^{-1} → film/country placeLived/location → location/contains^{-1} ethnicity/people^{-1} → ethnicity/geographicDistribution
3	personBornInLocation	personBornInCity hasSibling^{-1} → personBornInCity personGraduatedFromUniversity → personGraduatedSchool^{-1} → personBornInCity personDiedAtAge → personDiedAtAge^{-1} → personBornInCity hasHusband → hasWife → personBornInCity personMovedToStateOrProvince → personMovedToStateOrProvince^{-1} → personbornincity personBornInCity → atLocation^{-1} → atLocation personGraduatedSchool → personGraduatedSchool^{-1} → personBornInCity	personBornInCity personMovedToStateOrProvince personBornInCity → cityLocatedInState personGraduatedFromUniversity → personGraduatedFromUniversity^{-1} → personBornInCity personGraduatedFromUniversity → personGraduatedSchool^{-1} → personBornInCity personBelongsToOrganization → personBelongsToOrganization^{-1} → personBornInCity personBornInCity → agentActsInLocation → atLocation personBornInCity → LocationLocatedWithInLocation → LocationLocatedWithInLocation^{-1}
4	athletePlaysForTeam	athleteLedSportsTeam agentCollaboratesWithAgent^{-1} personBelongsToOrganization athleteHomeStadium → teamHomeStadium^{-1} athleteLedSportsTeam → teamPlaysAgainstTeam agentbelongstoorganization → teamplaysinleague^{-1} athleteHomeStadium → athleteHomeStadium^{-1} → athleteLedSportsTeam athleteplaysinleague → personBelongsToOrganization^{-1} → personBelongsToOrganization	athleteLedSportsTeam personBelongsToOrganization agentCollaboratesWithAgent^{-1} athleteHomeStadium → teamHomeStadium^{-1} coachwontrophy → teamwontrophy^{-1} athleteLedSportsTeam → teamPlaysAgainstTeam athleteLedSportsTeam → teamPlaysAgainstTeam^{-1} athleteplaysinleague → teamplaysinleague^{-1}

Lived/location → location/contains^{-1}", our approach also find some innovative paths, e.g. "performance/film → film/country" and "deceasedPerson/placeOfDeath → bibsLocation/country".

For the third relation, our approach finds some instructive reasoning paths, such as "personBornInCity hasSibling^{-1} → personBornInCity". This relationship means that in many cases couples are likely to be born in the same place. Therefore, our model can infer the birthplace of the husband from the birthplace of the wife. "personGraduatedSchool → personGraduatedSchool^{-1} → personBornInCity" implies that the agent discovers that the schoolmates might have been born in the same place. Therefore, the agent can infer the

birthplace of the person from the schoolmates.

For the fourth relation, " athleteHomeStadium → athleteHomeStadium^{-1} → athleteLedSportsTeam" indicates that the agent can infer the belonging information of the team members from the information of the captain. "athleteplaysinleague → personBelongsToOrganization^{-1} → personBelongsToOrganization " means that the agent find this common sense that teammates belong to the same organization.

5. Conclusion

This paper proposes the DAPath for knowledge graph reasoning using a DRL framework. The model can assign different rewards based on the portions of the agent. We use a graph self-attention mechanism to capture more comprehensive entity information from the neighborhood. We incorporate the GSA mechanism with GRU to alleviate the model from pretraining. Our model eliminates the pre-training or fine-tuning process and achieves better results, which significantly reduces the problem complexity. We found that our model can mine more balanced paths for each relation and find commonsense reasoning paths. It seems promising to incorporate the relation extraction models [60, 61] with the RL-based link prediction method to extract more evidence from the text to facilitate the inference process.

Acknowledgment

"This project has received funding from the European Union's Horizon 2020 research and innovation programme under the Marie Sklodowska-Curie grant agreement No 721321"

References

[1] W. Xiong, T. Hoang, W. Y. Wang, Deeppath: A reinforcement learning method for knowledge graph reasoning, in: Proceedings of EMNLP, Association for Computational Linguistics, 2017, pp. 564–573.

[2] V. Gullapalli, A stochastic reinforcement learning algorithm for learning real-valued functions, Neural networks 3 (6) (1990) 671–692.

[3] M. Haruno, M. Kawato, Heterarchical reinforcement-learning model for integration of multiple cortico-striatal loops: fmri examination in stimulus-action-reward association learning, Neural networks 19 (8) (2006) 1242–1254.

[4] A. Johnson, A. D. Redish, Hippocampal replay contributes to within session learning in a temporal difference reinforcement learning model, Neural Networks 18 (9) (2005) 1163–1171.

[5] N. Schweighofer, K. Doya, Meta-learning in reinforcement learning, Neural Networks 16 (1) (2003) 5–9.

[6] B. Luo, H.-N. Wu, T. Huang, D. Liu, Reinforcement learning solution for hjb equation arising in constrained optimal control problem, Neural Networks 71 (2015) 150–158.

[7] S. Elfwing, E. Uchibe, K. Doya, Sigmoid-weighted linear units for neural network function approximation in reinforcement learning, Neural Networks 107 (2018) 3–11.

[8] P. A. Laurent, The emergence of saliency and novelty responses from reinforcement learning principles, Neural Networks 21 (10) (2008) 1493–1499.

[9] A. S. Iwashita, J. P. Papa, A. X. Falcão, R. de Alencar Lotufo, V. M. de Araujo Oliveira, V. H. C. de Albuquerque, J. M. R. S. Tavares, Speeding up optimum-path forest training by path-cost propagation, in: Proceedings ICPR, IEEE Computer Society, 2012, pp. 1233–1236.

[10] T. Bansal, D.-C. Juan, S. Ravi, A. McCallum, A2N: Attending to neighbors for knowledge graph inference, in: Proceedings of ACL, Association for Computational Linguistics, 2019, pp. 4387–4392.

[11] Y. Zhu, H. Liu, Z. Wu, Y. Song, T. Zhang, Representation learning with ordered relation paths for knowledge graph completion, in: Proceedings of EMNLP-IJCNLP, Association for Computational Linguistics, 2019, pp. 2662–2671.

[12] H. Wang, S. Li, R. Pan, M. Mao, Incorporating graph attention mechanism into knowledge graph reasoning based on deep reinforcement learning, in: Proceedings of EMNLP-IJCNLP, Association for Computational Linguistics, 2019, pp. 2623–2631.

[13] A. Graves, J. Schmidhuber, Framewise phoneme classification with bidirectional lstm and other neural network architectures, Neural networks 18 (5-6) (2005) 602–610.

[14] J. Chung, Ç. Gülçehre, K. Cho, Y. Bengio, Empirical evaluation of gated recurrent neural networks on sequence modeling, CoRR abs/1412.3555.

[15] W. Y. Wang, W. W. Cohen, Learning first-order logic embeddings via matrix factorization, in: S. Kambhampati (Ed.), Proceedings of IJCAI, IJCAI/AAAI Press, 2016, pp. 2132–2138.

[16] F. Yang, Z. Yang, W. W. Cohen, Differentiable learning of logical rules for knowledge base reasoning, in: Proceedings of NIPS, 2017, pp. 2319–2328.

[17] A. Bordes, N. Usunier, A. García-Durán, J. Weston, O. Yakhnenko, Translating embeddings for modeling multi-relational data, in: C. J. C. Burges, L. Bottou, Z. Ghahramani, K. Q. Weinberger (Eds.), Proceedings of NIPS, 2013, pp. 2787–2795.

[18] Q. Wang, Z. Mao, B. Wang, L. Guo, Knowledge graph embedding: A survey of approaches and applications, IEEE Trans. Knowl. Data Eng. 29 (12) (2017) 2724–2743.

[19] Z. Wang, J. Zhang, J. Feng, Z. Chen, Knowledge graph embedding by translating on hyperplanes, in: C. E. Brodley, P. Stone (Eds.), Proceedings of AAAI, 2014, pp. 1112–1119.

[20] Y. Lin, Z. Liu, M. Sun, Y. Liu, X. Zhu, Learning entity and relation embeddings for knowledge graph completion, in: B. Bonet, S. Koenig (Eds.), Proceedings of AAAI, 2015, pp. 2181–2187.

[21] G. Ji, S. He, L. Xu, K. Liu, J. Zhao, Knowledge graph embedding via dynamic mapping matrix, in: Proceedings of ACL, 2015, pp. 687–696.

[22] G. Ji, K. Liu, S. He, J. Zhao, Knowledge graph completion with adaptive sparse transfer matrix, in: D. Schuurmans, M. P. Wellman (Eds.), Proceedings of AAAI, Phoenix, Arizona, USA, AAAI Press, 2016, pp. 985–991.

[23] H. Xiao, M. Huang, X. Zhu, Transg : A generative model for knowledge graph embedding, in: Proceedings of ACL, Berlin, Germany, The Association for Computer Linguistics, 2016.

[24] H. Xiao, M. Huang, Y. Hao, X. Zhu, Transa: An adaptive approach for knowledge graph embedding, CoRR abs/1509.05490. `arXiv:1509.05490`.
URL `http://arxiv.org/abs/1509.05490`

[25] S. He, K. Liu, G. Ji, J. Zhao, Learning to represent knowledge graphs with gaussian embedding, in: J. Bailey, A. Moffat, C. C. Aggarwal, M. de Rijke, R. Kumar, V. Murdock, T. K. Sellis, J. X. Yu (Eds.), Proceedings of CIKM, Melbourne, VIC, Australia, ACM, 2015, pp. 623–632.

[26] A. Rossi, D. Firmani, A. Matinata, P. Merialdo, D. Barbosa, Knowledge graph embedding for link prediction: A comparative analysis, arXiv preprint arXiv:2002.00819.

[27] B. Yang, W. Yih, X. He, J. Gao, L. Deng, Embedding entities and relations for learning and inference in knowledge bases, in: Y. Bengio, Y. LeCun (Eds.), Proceedings of ICLR 2015, 2015.

[28] T. Trouillon, J. Welbl, S. Riedel, É. Gaussier, G. Bouchard, Complex embeddings for simple link prediction, in: M. Balcan, K. Q. Weinberger (Eds.), Proceedings of ICML, Vol. 48 of JMLR Workshop and Conference Proceedings, JMLR.org, 2016, pp. 2071–2080.

[29] H. Liu, Y. Wu, Y. Yang, Analogical inference for multi-relational embeddings, in: D. Precup, Y. W. Teh (Eds.), Proceedings of ICML, Vol. 70 of Proceedings of Machine Learning Research, PMLR, 2017, pp. 2168–2178.

[30] S. M. Kazemi, D. Poole, Simple embedding for link prediction in knowledge graphs, in: S. Bengio, H. M. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, R. Garnett (Eds.), Proceedings of NIPS, 2018, pp. 4289–4300.

[31] M. Nickel, L. Rosasco, T. A. Poggio, Holographic embeddings of knowledge graphs, in: D. Schuurmans, M. P. Wellman (Eds.), Proceedings of AAAI, AAAI Press, 2016, pp. 1955–1961.

[32] I. Balazevic, C. Allen, T. M. Hospedales, Tucker: Tensor factorization for knowledge graph completion, in: Proceedings of EMNLP-IJCNLP, 2019.

[33] T. Dettmers, P. Minervini, P. Stenetorp, S. Riedel, Convolutional 2d knowledge graph embeddings, in: S. A. McIlraith, K. Q. Weinberger (Eds.), Proceedings of AAAI, AAAI Press, 2018, pp. 1811–1818.

[34] D. Q. Nguyen, T. D. Nguyen, D. Q. Nguyen, D. Q. Phung, A novel embedding model for knowledge base completion based on convolutional neural network, in: M. A. Walker, H. Ji, A. Stent (Eds.), Proceedings of NAACL-HLT, Association for Computational Linguistics, 2018, pp. 327–333.

[35] X. Jiang, Q. Wang, B. Wang, Adaptive convolution for multi-relational learning, in: J. Burstein, C. Doran, T. Solorio (Eds.), Proceedings of NAACL-HLT, Association for Computational Linguistics, 2019, pp. 978–987.

[36] D. Q. Nguyen, T. Vu, T. D. Nguyen, D. Q. Nguyen, D. Q. Phung, A capsule network-based embedding model for knowledge graph completion and search personalization, in: J. Burstein, C. Doran, T. Solorio (Eds.), Proceedings of NAACL-HLT, Association for Computational Linguistics, 2019, pp. 2180–2189.

[37] L. Guo, Z. Sun, W. Hu, Learning to exploit long-term relational dependencies in knowledge graphs, in: K. Chaudhuri, R. Salakhutdinov (Eds.), Proceedings of ICML, Vol. 97 of Proceedings of Machine Learning Research, PMLR, 2019, pp. 2505–2514.

[38] N. Lao, T. M. Mitchell, W. W. Cohen, Random walk inference and learning in A large scale knowledge base, in: Proceedings of EMNLP, ACL, 2011, pp. 529–539.

[39] N. Lao, W. W. Cohen, Relational retrieval using a combination of path-constrained random walks, Machine Learning 81 (1) (2010) 53–67.

[40] A. Neelakantan, B. Roth, A. McCallum, Compositional vector space models for knowledge base completion, in: Proceedings of ACL, Beijing, China, The Association for Computer Linguistics, 2015, pp. 156–166.

[41] R. Das, A. Neelakantan, D. Belanger, A. McCallum, Chains of reasoning over entities, relations, and text using recurrent neural networks, in: M. Lapata, P. Blunsom, A. Koller (Eds.), Proceedings of EACL, Association for Computational Linguistics, 2017, pp. 132–141.

[42] R. Das, S. Dhuliawala, M. Zaheer, L. Vilnis, I. Durugkar, A. Krishnamurthy, A. Smola, A. McCallum, Go for a walk and arrive at the answer: Reasoning over paths in knowledge bases using reinforcement learning, in: Proceedings of ICLR, 2018.

[43] X. V. Lin, R. Socher, C. Xiong, Multi-hop knowledge graph reasoning with reward shaping, in: Proceedings of EMNLP, Association for Computational Linguistics, 2018, pp. 3243–3253.

[44] W. Chen, W. Xiong, X. Yan, W. Y. Wang, Variational knowledge graph reasoning, in: M. A. Walker, H. Ji, A. Stent (Eds.), Proceedings of NAACL-HLT, Association for Computational Linguistics, 2018, pp. 1823–1832.

[45] Y. Shen, J. Chen, P. Huang, Y. Guo, J. Gao, M-walk: Learning to walk over graphs using monte carlo tree search, in: Proceedings of NIPS, 2018, pp. 6787–6798.

[46] X. Lv, Y. Gu, X. Han, L. Hou, J. Li, Z. Liu, Adapting meta knowledge graph information for multi-hop reasoning over few-shot relations, in: Proceedings of EMNLP-IJCNLP, Association for Computational Linguistics, 2019, pp. 3374–3379.

[47] T. N. Kipf, M. Welling, Semi-supervised classification with graph convolutional networks, arXiv preprint arXiv:1609.02907.

[48] M. S. Schlichtkrull, T. N. Kipf, P. Bloem, R. van den Berg, I. Titov, M. Welling, Modeling relational data with graph convolutional networks, in: Proceedings of ESWC 2018, 2018, pp. 593–607.

[49] P. Veličković, G. Cucurull, A. Casanova, A. Romero, P. Liò, Y. Bengio, Graph Attention Networks, International Conference on Learning Representations.

[50] Z. Karevan, J. A. Suykens, Transductive lstm for time-series prediction: An application to weather forecasting, Neural Networks 125 (2020) 1–9.

[51] R. S. Sutton, A. G. Barto, Reinforcement learning: An introduction, MIT press, 2018.

[52] R. S. Sutton, D. A. McAllester, S. P. Singh, Y. Mansour, Policy gradient methods for reinforcement learning with function approximation, in: S. A. Solla, T. K. Leen, K. Müller (Eds.), Proceedings of NIPS, The MIT Press, 1999, pp. 1057–1063.

[53] R. J. Williams, Simple statistical gradient-following algorithms for connectionist reinforcement learning, Machine Learning 8 (1992) 229–256.

[54] A. Carlson, J. Betteridge, B. Kisiel, B. Settles, E. R. H. Jr., T. M. Mitchell, Toward an architecture for never-ending language learning, in: M. Fox, D. Poole (Eds.), Proceedings of AAAI, AAAI Press, 2010.

[55] K. Toutanova, D. Chen, P. Pantel, H. Poon, P. Choudhury, M. Gamon, Representing text for joint embedding of text and knowledge bases, in: L. Màrquez, C. Callison-Burch, J. Su, D. Pighin, Y. Marton (Eds.), Proceedings of EMNLP, The Association for Computational Linguistics, 2015, pp. 1499–1509.

[56] Y. Zhang, Q. Yao, W. Dai, L. Chen, Autosf: Searching scoring functions for knowledge graph embedding, in: 36th IEEE International Conference on Data Engineering, ICDE 2020, Dallas, TX, USA, April 20-24, 2020, IEEE, 2020, pp. 433–444.

[57] D. Nathani, J. Chauhan, C. Sharma, M. Kaul, Learning attention-based embeddings for relation prediction in knowledge graphs, in: A. Korhonen, D. R. Traum, L. Màrquez (Eds.), Proceedings of ACL, Association for Computational Linguistics, 2019, pp. 4710–4723.

[58] R. Wang, B. Li, S. Hu, W. Du, M. Zhang, Knowledge graph embedding via graph attenuated attention networks, IEEE Access 8 (2020) 5212–5224.

[59] B. Jagvaral, W.-K. Lee, J.-S. Roh, M.-S. Kim, Y.-T. Park, Path-based reasoning approach for knowledge graph completion using cnn-bilstm with attention mechanism, Expert Systems with Applications 142 (2020) 112960.

[60] D. Wang, P. Tiwari, S. Garg, H. Zhu, P. Bruza, Structural block driven enhanced convolutional neural representation for relation extraction, Appl. Soft Comput. 86.

[61] S. Zheng, F. Wang, H. Bao, Y. Hao, P. Zhou, B. Xu, Joint extraction of entities and relations based on a novel tagging scheme, in: Proceedings of ACL, Association for Computational Linguistics, 2017, pp. 1227–1236.