Incorporating Graph Attention Mechanism into Knowledge Graph Reasoning Based on Deep Reinforcement Learning

Heng Wang[⋆] **Shuangyin Li**[∗] **Rong Pan**[†] **Mingzhi Mao**[†]
[⋆]Tencent, China
[†]School of Data and Computer Science, Sun Yat-sen University, China
[∗]School of Computer Science, South China Normal University, China
jimmyhwang@tencent.com
{panr@mail,mcsmmz@mail}.sysu.edu.cn
shuangyinli@m.scnu.edu.cn

Abstract

Knowledge Graph (KG) reasoning aims at finding reasoning paths for relations, in order to solve the problem of incompleteness in KG. Many previous path-based methods like PRA and DeepPath suffer from lacking memory components, or stuck in training. Therefore, their performances always rely on well-pretraining. In this paper, we present a deep reinforcement learning based model named by **AttnPath**, which incorporates LSTM and Graph Attention Mechanism as the memory components. We define two metrics, Mean Selection Rate (**MSR**) and Mean Replacement Rate (**MRR**), to quantitatively measure how difficult it is to learn the query relations, and take advantages of them to fine-tune the model under the framework of reinforcement learning. Meanwhile, a novel mechanism of reinforcement learning is proposed by forcing an agent to walk forward every step to avoid the agent stalling at the same entity node constantly. Based on this operation, the proposed model not only can get rid of the pretraining process, but also achieves state-of-the-art performance comparing with the other models. We test our model on FB15K-237 and NELL-995 datasets with different tasks. Extensive experiments show that our model is effective and competitive with many current state-of-the-art methods, and also performs well in practice.

1 Introduction

Knowledge Graphs (KGs), such as NELL (Carlson et al., 2010), Freebase (Bollacker et al., 2008) and WordNet (Miller, 1995) play an increasingly critical role in many downstream NLP applications, e.g., question answering (Dubey et al., 2018), information retrieval (Liu et al., 2018), personalized recommendations (Wang et al., 2019), etc. However, KGs are always incomplete, which would affect many downstream tasks. Many links could miss among the entities in a KG. Thus, it is an important and challenge task on how to complete the KG by predicting the missing links between the entities through the method based on reasoning. For instance, if $athletePlaysForTeam(X, Y)$ and $teamPlaysInLeague(Y, Z)$ both exist in the KG, then we can infer that $athletePlaysInLeague(X, Z)$, i.e, filling the missing edge $athletePlaysInLeague$ between X and Z.

There are mainly three ways to accomplish this task, such as Rule-Based (Wang and Cohen, 2016; Yang et al., 2017), Embedding-Based (Bordes et al., 2013; Lin et al., 2015) and Path-Based (Lao et al., 2011). Meanwhile, it provides a new perspective to bring Deep Reinforcement Learning (DRL) into the task of predicting the missing links, such as DeepPath (Xiong et al., 2017), a type of path-based method. DeepPath is the first work which incorporates DRL into KG reasoning. It achieves significant improvements compared with PRA, but still has several drawbacks. First, it lacks memory components, resulting in requiring **pretraining**. The operation of pretraining is demanded to provide many known (or existed) paths to the model training. This brute-force operation may make the model prone to overfit on the given paths from pretraining. Second, it's inappropriate to set the same hyperparameter for different relations in a KG when training, which ignores the diversity of connections among the entities. Last, when the agent selects an invalid path, it will stop and reselect, which leads to select this invalid path constantly and finally stuck in one node.

Thus, in this paper, we present a novel deep reinforcement learning model and an algorithm, which aims at tackling the drawbacks mentioned above. The proposed model also belongs to the path-based framework. Our contributions can be summarized as follows:

- We propose **AttnPath**, a model which incorporates **LSTM** and **graph attention** as memory components, and is no longer subject to pretraining.

- Two metrics are defined (**MSR** and **MRR**), to quantitatively measure the difficulty to learn a relation's replaceable paths, which are used to fine-tune the model.

- A novel mechanism of reinforcement learning is proposed by forcing an agent to walk forward every step to avoid the agent stalling at the same entity node constantly.

We test **AttnPath** on FB15K-237 and NELL-995 datasets with two downstream tasks: fact prediction and link prediction. We also test on the success rate in finding paths and show the effectiveness of the Graph Attention Mechanism in the experiments.

2 Related Works

To date, there are many works proposed to solve the problem of KG incompleteness. Rule-Based methods, like ProPPR (Wang and Cohen, 2016) and Neural LP (Yang et al., 2017), generate reasoning rules manually or by mathematical logic rules, and then apply them to fill missing links based on existing triples. Although this type of methods has a solid mathematical background, they are hard to scale to large KGs, since they directly operate on symbols, while the number of possible reasoning paths is exponential to the number of the entities. Embedding-Based methods, like TransE (Bordes et al., 2013) and TransR (Lin et al., 2015), map entities and relations into a low-dimensional and continuous vector space, which captures the feature of distance between entities and relations. Then, they judge whether the query relation exists by comparing the distance between the two trained entities' embeddings and the query relation's embedding. This type of methods requires all triples in the KG to participate in training, and are only suitable for single-hop reasoning.

Path-Based, like PRA (Lao et al., 2011) and DeepPath (Xiong et al., 2017), train an agent to navigate on a KG, find replaceable paths for a certain relation, and then, use them as features to the downstream tasks. Path Ranking Algorithm (PRA) is the first path-based reasoning method. Neelakantan et al. develop a compositional model based on RNN, which composes the implications of a path and reasons about conjunctions of multi-hop relations non-atomically (Neelakantan et al., 2015). Guu et al. propose a soft edge traversal operator, which can be recursively applied to predict paths and reduce cascaded propagation errors faced by single-hop KG completion methods like TransE and TransR (Guu et al., 2015). Toutanova et al. propose a dynamic programming algorithm which incorporates all relations' paths of bounded length in a KG, and models both relations and intermediate nodes in the compositional path representations (Toutanova et al., 2016). Such representations can aid in generating more high-quality reasoning paths.

Das et al. improve DeepPath (Xiong et al., 2017) to MINERVA (Das et al., 2018), which views KG from QA's perspective. It gets rid of pretraining, introduces LSTM to memorize paths traversed before, and trains an agent to circulate on a certain entity if it believes this entity is the right answer. Lin et al. improve these two methods by introducing reward shaping and action dropout (Lin et al., 2018). Reward shaping replaces fixed penalty for useless selection with a dynamic penalty, which can either base on margin-based pretrained embeddings like TransE, or probability-based embeddings like ConvE (Dettmers et al., 2018). While action dropout randomly masks a certain proportion of valid actions, in order to reduce irrelevant paths to the query relation. DIVA (Chen et al., 2018) regards paths as latent variables, and relations as observed variables, so as to build a variational inference model to accomplish the KG reasoning task. It also uses beam search to broaden the search horizon. M-Walk (Shen et al., 2018) utilizes another RL algorithm called Monte Carlo Tree Search (MCTS), to tackle the problem of sparse rewards. The attention mechanism is first introduced into multi-hop KG reasoning by (Wang et al., 2018). However, it only computes the attention weights of the query's embedding and all found paths' embeddings. They are used to help to judge whether the answer found by the vanilla model is true.

3 AttnPath: Incorporating Memory Components

In this section, we will introduce the details of the proposed AttnPath. We will also show the definition of mean selection rate (MSR) and mean replacement rate (MRR), and also how they are used to fine-tune the model for different query relations.

3.1 RL framework for KG reasoning

Since we use Reinforcement Learning (RL) as the training algorithm of a sequential decision model, we first introduce basic elements of the RL framework in the KG reasoning, including environment, state, action, and reward.

Environment: In this task, the environment refers to the whole KG, excluding the query relation and its inverse. The environment retains consistent in the whole training process.

State: The state of an agent is concatenated with three parts: the embedding part, the LSTM part, and the graph attention part. We will show the computation of the LSTM part and the graph attention part in the next section, and introduce the embedding part first.

Following (Xiong et al., 2017), the embedding part $\mathbf{m}_t$ is concatenated with two subparts. The first one is $\mathbf{e}_t$, which is the embedding of the current entity node. The other one is $\mathbf{e}_{target} - \mathbf{e}_t$, denoting the distance between the tail entity node and the current node. Different from DeepPath which uses TransE (Bordes et al., 2013) as pretrained embeddings, we take advantage of TransD (Ji et al., 2015), which is an improvement of TransE and also a common-used benchmark. Under TransD, for a query relation, we project all the entities onto the vector space of this query relation. Then, an entity's projected embedding $\mathbf{e}_\perp$ becomes

$$\mathbf{e}_\perp = (\mathbf{r_p e_p^\top} + \mathbf{I})\mathbf{e}, \tag{1}$$

where $\mathbf{p}$ indicates the projection vectors. So $\mathbf{m}_t$ should be $[\mathbf{e}_{t\perp}; \mathbf{e}_{target\perp} - \mathbf{e}_{t\perp}]$.

Action: For the KG reasoning task, an action refers to an agent choosing a relation path to step forward. Based on the framework of DRL, it chooses the relation according to the probabilities obtained by the model. Actions are either valid or invalid. Valid action means that there is an output relation connected with the current entity, while invalid action denotes that there is no relation.

Reward: Reward is a feedback to the agent according to whether the action is valid, and whether a series of actions can lead to ground truth tail entities in a specified number of times. We adopt reward shaping trick proposed by (Lin et al., 2018). For the invalid actions, the reward is -1. For the actions which don't lead to ground truth, we choose the output of ConvE (Dettmers et al., 2018) as the reward. Since ConvE outputs probabilities, which are in the range of $(0, 1)$, we resort a logarithm operator to enlarge the range of this reward and improve discrimination. For the actions which lead to ground truth, i.e, a successful episode, the reward is the weighted sum of the global accuracy, the path efficiency, and the path diversity. The global accuracy is set to 1 by convention, and the path efficiency is the reciprocal of the path length, because we encourage the agent to step as fewer times as possible. The path diversity is defined as

$$r_{\text{DIVERSITY}} = -\frac{1}{|F|}\sum_{i=1}^{|F|} \cos(\mathbf{p}, \mathbf{p}_i), \tag{2}$$

where $|F|$ is the number of found paths, and $\mathbf{p}$ is the path embedding, simply the sum of all relations' embeddings in the path. The above definition guarantees that the reward of the valid actions are always larger than the invalid actions, and the reward of the successful episodes are always larger than the unsuccessful ones.

3.2 LSTM and Graph Attention as Memory Components

In our model, we utilize a three-layer LSTM, enabling the agent to memorize and learn from the actions taken before. Denote the hidden state of LSTM at step t by $\mathbf{h}_t$, and the initial hidden state $\mathbf{h}_0$ by $\mathbf{0}$. Then we obtain

$$\mathbf{h}_t = \text{LSTM}(\mathbf{h}_{t-1}, \mathbf{m}_t), \tag{3}$$

where $\mathbf{m}_t$ is defined in Eq. (1). This is the LSTM part of the state described above.

Typically, an entity has several different aspects, for example, a football player may be connected with professional relations like $playsForTeam$ or $playsInLeague$, and also family relations like $spouse$ or $father$. For different query relations, it's better for the agent to focus more on relations and neighbors which are highly related to the query relations. So we introduce Graph Attention mechanism (GAT) into our model, which is proposed by (Velickovic et al., 2018).

GAT is indeed self-attention on the entity nodes. We use a single-layer feedforward neural network to calculate attention weights, with a linear transformation matrix $\mathbf{W}$ and a weight vector $\vec{\mathbf{a}}$ shared across all entities. LeakyReLU with negative input slope $\alpha = 0.2$ is chosen as nonlinearity. So the attention weight from entity i to entity j is calculated as

$$a_{ij} = \text{LeakyReLU}(\vec{\mathbf{a}}^\top [\mathbf{W}\mathbf{e}_{i\perp}; \mathbf{W}\mathbf{e}_{j\perp}]). \tag{4}$$

For entity i, we only compute attention weights to all its direct-connected neighbors, and normalize them with SoftMax. So the normalized attention weight is

$$\alpha_{ij} = \frac{\exp(a_{ij})}{\sum_{k \in \mathcal{N}_i} \exp(a_{ik})}. \quad (5)$$

Now we can compute the graph attention part for the state, which is simply the weighted sum of all neighbors' embedding on the attention space with

$$\mathbf{a}_i = \sum_{k \in \mathcal{N}_i} \alpha_{ik} \mathbf{W} \mathbf{e}_k. \quad (6)$$

Thus, the state vector $\mathbf{s}_{i,t}$ of entity i in time t is

$$\mathbf{s}_{i,t} = [\mathbf{m}_{i,t}; \mathbf{h}_t; \mathbf{a}_i], \quad (7)$$

which in turn inputs a three-layer feedforward neural network, whose final output is a Softmax probability with length equal to the number of all relations in KG. The agent selects an action and obtains a reward. After it successfully reaches the tail entity or doesn't reach in a specified number of times, the rewards of the whole episode are used to update all parameters. The optimization is done using the REINFORCE (Williams, 1992) algorithm and updates θ with the following stochastic gradient:

$$\nabla_\theta J(\theta) \approx \nabla_\theta \sum_{t=1}^{T} R(\mathbf{s}_T | e_s, r) \log \pi_\theta(a_t | s_t), \quad (8)$$

where e_s is head entity, r the query relation, and $\pi_\theta(a_t|s_t)$ the probability of all relations. Figure 1 shows our **AttnPath** model.

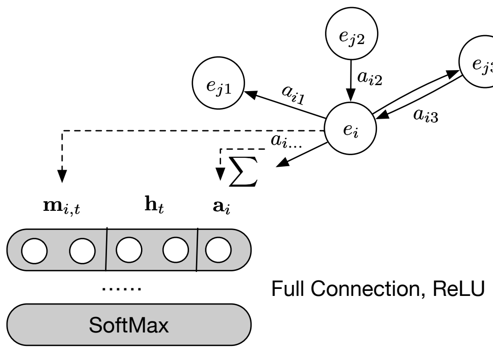

Figure 1: AttnPath: an RL framework composed with LSTM and graph attention for KG reasoning.

Algorithm 1 Training Algorithm of the agent for AttnPath

```
1: for $episode \leftarrow 1$ to $N$ do
2:   Initialize LSTM's hidden state $\mathbf{h}_0$ to $\mathbf{0}$
3:   Initialize state vector $\mathbf{s}_0$
4:   Initialize $num\_steps$ to 0
5:   while $num\_steps < max\_steps$ do
6:     Randomly sample action $a \sim \pi_\theta(a_t|s_t)$
7:     if Action invalid (leads to nothing) then
8:       Add $< s_t, a >$ to $\mathcal{M}_{neg}$
9:       Normalize $\pi_\theta(a_t|s_t)$ on valid actions and sample $a_f$
10:      Add $< s_t, a_f >$ to $\mathcal{M}_{pos}$
11:    else
12:      Add $< s_t, a >$ to $\mathcal{M}_{pos}$
13:    end if
14:    Increment $num\_steps$
15:    if $success$ or $num\_steps = max\_steps$ then
16:      break
17:    end if
18:  end while
19:  Update $\theta$ using $g \propto \sum_{\mathcal{M}_{neg}} \log\pi(a_t|s_t;\theta)(-1)$
20:  if $success$ then
21:    $R_{total} \leftarrow \lambda_1 r_{\text{GLOBAL}} + \lambda_2 r_{\text{EFFICIENCY}} + \lambda_3 r_{\text{DIVERSITY}}$
22:    Update $\theta$ using $g \propto \sum_{\mathcal{M}_{pos}} \log\pi(a_t|s_t;\theta) R_{total}$
23:  else
24:    Compute shaped reward $R_{shaping}$ for $\mathcal{M}_{pos}$
25:    Update $\theta$ using $g \propto \sum_{\mathcal{M}_{pos}} \log\pi(a_t|s_t;\theta) R_{shaping}$
26:  end if
27: end for
```

3.3 Mean Selection / Replacement Rate

For different query relations, it is required to train different models for each of them. While, in practice, the difficulty values of each relation are quite different. Some relation may have more replacement relations, indicating that an agent can easily choose a replacement path walking from head entity to tail. So we invent two metrics, **Mean Selection Rate (MSR)** and **Mean Replacement Rate (MRR)** here, to quantitatively measure the difficulty value of each relation.

Denote all triples related to relation r by a set of $\mathcal{T}_r = \{(h, r_o, t) | r_o = r\}$. The selection rate for relation r with regard to triple (h, r, t) is defined as

$$SR((h,r,t)) = \frac{|\{(h_o, r_o, t_o) | h_o = h, r_o = r\}|}{|\{(h_o, r_o, t_o) | h_o = h\}|}, \tag{9}$$

i.e, the proportion of relation r occupying h's outgoing paths.

Thus, MSR is the average of SR on $\mathcal{T}_r$:

$$MSR(r) = \sum_{(h_o, r, t_o) \in \mathcal{T}_r} SR(h_o, r, t_o) / |\mathcal{T}_r|. \tag{10}$$

Lower MSR denotes that it's more difficult to learn r, because the entities connected with relation r may have more aspects.

The replacement rate for relation r with regard to triple (h, r, t) is defined as

$$RR((h,r,t)) = \frac{|\{(h_o, r_o, t_o) | h_o = h, r_o \neq r, t_o = t\}|}{|\{(h_o, r_o, t_o) | h_o = h, t_o = t\}|}, \tag{11}$$

i.e, the proportion of relations that directly connects h and t, except relation r.

Similarly, MRR is the average of RR on $\mathcal{T}_r$:

$$MRR(r) = \sum_{(h_o, r, t_o) \in \mathcal{T}_r} RR(h_o, r, t_o) / |\mathcal{T}_r|. \tag{12}$$

Higher MRR denotes a relation may have more replacement relations, so it's easier to learn since an agent can directly choose an alternative relation to reach the destination.

In our model, we have three ways to prevent overfitting: L2 regularization, dropout, and action dropout. However, for the relations which are easier to learn (high MSR and MRR), we wish to impose more regularization to encourage the agent to find more diverse paths, without overfitting on the immediate success. Otherwise, for the relations which are more difficult to learn (low MSR and MRR), we'd better focus on the success rate of finding a path, so we should impose less regularization.

For simplicity, we use exponential to compute the difficulty coefficient of relation r. It is defined as $\exp(MSR(r) + MRR(r))$ and multiplied with the base rate of three regularization methods, respectively. The base rate of regularization methods is KG-based, shared across all relations in the same KG.

Table 1: Statistics of the datasets. #Rel. and #Tri. doesn't include the inverse triples.

Dataset	#Ent.	#Rel.	#Tri.	#Task
FB15K-237	14,505	237	310,116	20
NELL-995	75,492	200	154,213	12

3.4 Overall Training Algorithm

Based on the proposed model, we present a novel training algorithm shown in Algorithm 1.

One of our contributions in our algorithm is, when the agent selects an invalid path, our model not only penalizes it, but also forces it to select a valid relation to step forward. The probabilities from the neural network are normalized across all valid relations, which in turn act the probabilities of the forced action.

After the initializations, Line 6 samples an action according to the output of the network. When the agent selects an invalid action, Line 7 $\sim$ 10 is executed, and Line 9 $\sim$ 10 forces the agent to step forward. When the agent selects a valid action, Line 12 is executed. Lines 19, 22 and 25 update parameters for the invalid actions, the valid actions in a successful episode, and the valid actions in an unsuccessful episode, respectively, with reward -1, R_{total} and $R_{shaping}$.

4 Experiments

In this section, we will perform extensive experiments to validate the effectiveness of our proposed AttnPath. For each task, we will mainly focus on three quantitative metrics: success rate (**SR**) in finding paths, MAP of fact prediction (**FP**), and MAP of link prediction (**LP**). We will also demonstrate some reasoning paths and triples to show that graph attention is effective in finding more high-quality paths and mining which aspect of the entity is important in a specific task.

4.1 Datasets and Settings

Two datasets, FB15K-237 (Toutanova et al., 2015) and NELL-995 (Xiong et al., 2017), are used in our experiments. Statistics of these two datasets are displayed in Table 1. Following the previous works, for each triple (h, r, t), we add its inverse triple (t, r^{-1}, h), to allow the agent to step backward.

We summary the hyperparameters involved in our experiments here. The pretrained embedding dimension is set to 100. The LSTM hidden dimension is set to 200. The attention dimension is set to 100. Thus, s is a 500-dimension vector by concatenating the above three vectors with two times of the pretrained embedding dimension. λ_1 is 0.1, λ_2 is 0.8, λ_3 is 0.1. For FB15K-237 dataset, we set base L2 regularization, Dropout rate and action dropout rate to 0.005, 0.15, and 0.15, respectively. Also, for NELL-995, we set them to 0.005, 0.1 and 0.1, respectively. We choose Adam (Kingma and Ba, 2015) as the optimizer with different learning rates of 0.001, β_1 0.9 and β_2 0.999. For each task, we train 500 episodes, and for each episode, max_steps is set to 50.

We verify the learned paths for each triple in FP and LP tasks when training, by the BFS-based method (Xiong et al., 2017).

4.2 Success Rate in Finding Paths

Our model is ignorant of the environment and triples at first, since it doesn't rely on pretraining. Thus, we record total SR, and SR in the recent 10 episodes to validate the agent's ability to learn the paths. For the tasks with less than 500 training samples, the samples are iterated first, then randomly sampled to reach 500 episodes. For the tasks with more than 500 training samples, we randomly select 500 out for training.

We select two relations, $athletePlaysInLeague$ and $organizationHeadquarteredInCity$ from NELL-995, to investigate its total SR and SR in recent 10 episodes. The former has relatively lower MSR and MRR, and the latter has higher. Figure 2 shows the results. It can be discovered that DeepPath outperforms our method at first, however, after 50 $\sim$ 150 epochs, our models surpass it. From SR-10 of AttnPath Force, we find that the initial SR is approximate to MRR, because the model knows nothing at all at first, so it selects a path randomly. As the training proceeds, the performance gains steady improvement. Similar results can also be found in other relations on FB15K-237.

4.3 Fact Prediction

Fact prediction (**FP**) aims at predicting whether an unknown fact is true or false. The proportion of positive triples and negative triples is about 1 : 10. For each relation, we use all found paths and reciprocal of length as weight, to accumulate the score of each triple based on whether the path is valid between h and t. Scores are ranked across all test set and Mean Average Precision (MAP) is used as the evaluation metric. The results are shown in Table 2. It can be seen that AttnPath outperforms TransE / R and DeepPath significantly. AttnPath MS / RR, which uses MSR and MRR to fine-tune the hyperparameters, also gains performance improvement. AttnPath Force is also effective. By forcing the agent to walk forward every step, it improves the SR of finding a path, which in turn enriches the feature of the downstream tasks. This is especially important for the relations, which lack direct-connected replacement paths and require path with long-term dependency. In fact, our methods achieve **SOTA** results on both the two datasets.

Table 2: Fact prediction MAP (%). Results of TransE / H / R / D are cited from (Xiong et al., 2017). DeepPath is retrained with TransD for a fair comparison, which performs slightly better than (Xiong et al., 2017) based on TransE.

Method	FB15K-237	NELL-995
TransE	27.7	38.3
TransH	30.9	38.9
TransR	30.2	40.6
TransD	30.3	41.3
DeepPath TransD	31.3	53.5
AttnPath	31.5	59.8
AttnPath MS / RR	34.6	65.4
AttnPath Force	**37.9**	**69.3**

4.4 Link Prediction

Link prediction (**LP**) aims at predicting the target entities. For each (h, r) pair, there is a ground truth t and about 10 generated false t. It is divided into a training set and a test set. We use the found paths as binary features, and train a classification model on the training set, applying it on the test set. LP also uses MAP as the evaluation metric, and the detailed results are shown in Table 3 [1]. Alike FP, our model also attains better results in LP, and the **SOTA** results on FB15K-237 dataset[2].

However, we also notice that AttnPath doesn't attain the best result under a small part of query relations, even lower than TransE / R. By analyzing triples related to these relations, we found that: 1) they have more outgoing edges of the other relations pointing to the entities which are not the

[1]We omit to show detailed result of TransH / D here, while (Xiong et al., 2017) also doesn't give detailed result of it, and it is mentioned in this paper that vanilla DeepPath already outperforms main embedding-based methods, including TransE / H / R / D.

[2]DIVA (Chen et al., 2018) claims that it attains 88.6% MAP on NELL-995 dataset.

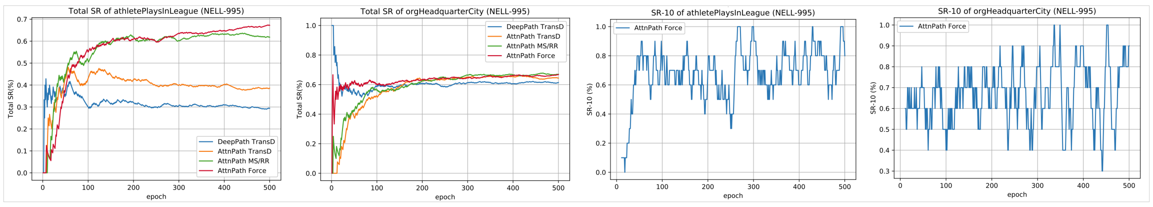

Figure 2: Total SR and SR-10 for two relations of NELL-995. DeepPath / AttnPath TransD means using TransD as the pretrained embeddings. AttnPath MS/RR is adding MSR and MRR to fine-tune hyperparameters. AttnPath Force is forcing the agent to move forward every step. These abbreviations are used throughout this section.

Table 3: Link Prediction MAP. Top half is **FB15K-237**, bottom half is **NELL-995**. TE, TR, DP, AP, APM and APF are abbreviations of TransE, TransR, DeepPath, AttnPath, AttnPath MS/RR, AttnPath Force, respectively. The result of TransE / R is cited from (Xiong et al., 2017). All the numbers are percentage (%).

Tasks	MSR	MRR	TE	TR	DP	AP	APM	APF
filmDirector	7.5	11.9	38.6	39.9	43.4	40.9	43.3	**43.7**
filmLanguage	3.7	0.1	64.2	64.1	70.1	56.6	70.9	**71.8**
filmWrittenBy	2.6	4.3	56.3	**60.5**	45.4	55.6	56.3	58.9
capitalOf	8.8	16.1	55.4	49.3	79.3	86.0	86.7	**87.2**
musicianOrigin	3.0	1.5	36.1	37.9	51.4	47.9	51.0	**52.6**
organizationFounded	4.2	1.3	39.0	33.9	31.8	31.1	34.9	**46.8**
personNationality	4.6	0.1	64.1	72.0	83.6	74.6	84.6	**84.6**
birthPlace	3.7	1.6	40.3	41.7	**55.0**	49.9	54.4	54.4
teamSports	6.7	0	89.6	78.4	**91.6**	70.2	91.2	91.3
tvLanguage	5.3	0.3	80.4	90.6	96.7	95.9	97.1	**96.8**
...								
Overall	—	—	53.2	54.0	63.5	58.5	65.3	**66.1**
athleteHomeStadium	32.4	0.0	71.8	72.2	89.3	89.3	88.7	**89.4**
athletePlaysForTeam	42.5	5.6	62.7	67.3	76.0	74.6	74.3	**76.1**
athletePlaysInLeague	34.9	0.3	77.3	91.2	96.2	95.6	95.1	**96.5**
athletePlaysSport	60.9	0	87.6	96.3	96.3	96.5	94.3	**97.0**
orgHeadquarterCity	42.6	25.0	62.0	65.7	92.8	92.9	93.2	**94.1**
orgHiredPerson	24.9	18.2	71.9	73.7	75.4	79.7	79.8	**81.6**
bornLocation	34.7	16.4	71.2	**81.2**	75.9	76.7	74.6	78.6
personLeadsOrg	32.6	36.9	75.1	77.2	79.6	80.9	81.0	**82.8**
teamPlaysSport	19.9	0	76.1	81.4	74.1	75.7	75.5	**82.1**
worksFor	45.6	27.7	67.7	69.2	71.2	71.3	71.8	**77.5**
...								
Overall	—	—	73.7	78.9	82.9	83.4	83.0	**85.8**

true tails, so MSR of these query relations are low. 2) Tail entities are only connected with edges of query relations and their inverse, while these edges are removed during training, so tail entities become isolated, without any possible replaceable paths. It will also lower MRR of these query relations. Take $birthPlace$ and $bornLocation$ as examples. If a person was born in a remote place, this place is difficult to be connected with the other entities, so it is prone to be isolated. However, such one-to-one relations are the strength of TransX methods.

4.5 Qualitative Analysis

4.5.1 Paths Found By DeepPath and AttnPath

We take $capitalOf$ from FB15K-237, and $athletePlaysInLeague$ from NELL-995, as examples, to analyze these paths found by DeepPath and AttnPath. Table 4 shows the top 5 frequent paths and their frequencies for all the methods. It shows that AttnPath is more capable of capturing long-term dependencies between relations, which is useful for relations lack of direct-connected replaceable paths. AttnPath can also find more important and concentrated paths, so the distribution of paths doesn't have a heavy long tail. In the training process, we also find that AttnPath is better at stepping backward when it enters a dead end.

4.5.2 Effectiveness of Graph Attention Mechanism

We sample several pairs of entity and relation, compute the entity's attention weights to its neighbors under this relation, and investigate neighbors with top 5 attention weights. Table 5 displays the examples. It shows that GAT is capable of pay-

Table 4: Top 5 frequent paths found for $capitalOf$ (FB15K-237) and $athletePlaysInLeague$ (NELL-995) using two methods. "_inv" indicates inverse relation.

Relation	**Reasoning Path**
capitalOf (DeepPath)	location^{-1} (83.2%); location^{-1} $\rightarrow$ location^{-1} (5.3%); administrativeArea^{-1} (4.4%); location^{-1} $\rightarrow$ location (1.8%); marriage^{-1} $\rightarrow$ spouse^{-1} $\rightarrow$ awardWinner $\rightarrow$ awardNomination $\rightarrow$ currency $\rightarrow$ rent^{-1} (0.9%)
capitalOf (AttnPath)	location^{-1} (89.1%); county (3.0%); location^{-1} $\rightarrow$ location^{-1} (2.4%); usCounty (1.8%); administrativeArea^{-1} (1.2%)
athletePlaysInLeague (DeepPath)	athleteplaysforteam $\rightarrow$ teamplaysinleague (28.5%); athleteplayssport $\rightarrow$ teamplayssport_inv $\rightarrow$ teamplaysinleague (24.5%); athleteledsportsteam $\rightarrow$ teamplaysinleague (13.2%); athleteflyouttosportsteamposition $\rightarrow$ athleteflyouttosportsteamposition_inv $\rightarrow$ athleteplaysforteam $\rightarrow$ teamplaysinleague (2.4%); athletehomestadium $\rightarrow$ leaguestadiums_inv (2.4%)
athletePlaysInLeague (AttnPath)	athleteplayssport $\rightarrow$ teamplayssport_inv $\rightarrow$ teamplaysinleague (70.0%); athleteplaysforteam $\rightarrow$ teamplaysinleague (19.5%); athleteledsportsteam $\rightarrow$ teamplaysinleague (4.3%); athleteflyouttosportsteamposition $\rightarrow$ athleteflyouttosportsteamposition_inv $\rightarrow$ athleteplayssport $\rightarrow$ teamplayssport_inv $\rightarrow$ teamplaysinleague (3.4%); athleteplaysforteam $\rightarrow$ teamplaysagainstteam_inv $\rightarrow$ teamplaysinleague (0.6%)

Table 5: Samples displaying which neighbor does the model pays the most attention to under certain relation. **Bold Text** means this entity is related to the query relation. The first four pairs are from FB15K-237 and the latter four are from NELL-995.

Entity	Relation	Neighbors
Ventura	location	**The United States**, **California**, **Pacific Time Zone**, Marriage, The Rock
Sikh	peopleLanguages Spoken	**Malay**, **Thai**, Bhubaneswar, **Hindi**, **Pashto**
Anthony Minghella	personNationality	**United Kingdom**, **England**, BAFTA Award for Best Direction, BAFTA Award for Best Film, University of Hull
	filmDirector	**BAFTA Award for Best Adapted Screenplay**, **The Talented Mr. Ripley**, **Academy Award for Best Picture**, England, Film Producer
Elon Musk	personLeads Organization	**Tesla Motors**, **Paypal**, **Tesla**, **SpaceX Museum**, **SpaceX company**
New Jersey Devils	teamPlaysSport	**Hockey**, Stanley Cup, Games, Capitals, Buffalo Sabres
Brandon Marshall	athleteHomeStadium	**Invesco Field**, Broncos, Golf, Super Bowl, Super Bowl XLII
	athletePlaysSport	**Golf**, Broncos, Invesco Field, Super Bowl XLII, Super Bowl

ing more attention to the neighbors, which is related to the query relation, especially for Anthony Minghella and Brandon Marshall, which pay different attention to neighbors with different query relations.

5 Conclusion and Future Work

In this paper, we propose **AttnPath**, a DRL based model for KG reasoning task which incorporates LSTM and Graph Attention Mechanism as memory components, to alleviate the model from pretraining. We also invent two metrics, MSR and MRR, to measure the learning difficulty of relations, and use it to better fine-tune training hyperparameters. We improve the training process to avoid the agent stalling in a meaningless state. Qualitative experiments and quantitative analysis show that our method outperforms DeepPath and embedding-based method significantly, proving its effectiveness.

In the future, we are interested in utilizing multi-task learning, to enable the model to learn reasoning paths for several query relations simultaneously. We would also like to research how to utilize GAT, MSR and MRR into other KG related tasks, such as KG representation, relation clustering and KB-QA.

Acknowledgments

This work is supported by the National Key R&D Program of China (2018YFB1004404), Key R&D Program of Guangdong Province (2018B010107005), National Natural Science Foundation of China (U1711262, U1401256, U1501252, U1611264, U1711261).

Rong Pan is the corresponding author.

References

Kurt Bollacker, Colin Evans, Praveen Paritosh, Tim Sturge, and Jamie Taylor. 2008. Freebase: a collaboratively created graph database for structuring human knowledge. In *SIGMOD*, pages 1247–1250.

Antoine Bordes, Nicolas Usunier, Alberto Garcia-

Duran, Jason Weston, and Oksana Yakhnenko. 2013. Translating embeddings for modeling multi-relational data. In *NeurIPS*, pages 2787–2795.

Andrew Carlson, Justin Betteridge, Bryan Kisiel, Burr Settles, Estevam R Hruschka Jr, and Tom M Mitchell. 2010. Toward an architecture for never-ending language learning. In *AAAI*, volume 5, pages 1306–1313.

Wenhu Chen, Wenhan Xiong, Xifeng Yan, and William Wang. 2018. Variational knowledge graph reasoning. In *NAACL-HLT*, pages 1823–1832.

Rajarshi Das, Shehzaad Dhuliawala, Manzil Zaheer, Luke Vilnis, Ishan Durugkar, Akshay Krishnamurthy, Alex Smola, and Andrew McCallum. 2018. Go for a walk and arrive at the answer: Reasoning over paths in knowledge bases using reinforcement learning. In *ICLR*.

Tim Dettmers, Pasquale Minervini, Pontus Stenetorp, and Sebastian Riedel. 2018. Convolutional 2d knowledge graph embeddings. In *AAAI*, pages 1811–1818.

Mohnish Dubey, Debayan Banerjee, Debanjan Chaudhuri, and Jens Lehmann. 2018. Strong baselines for simple question answering over knowledge graphs with and without neural networks. In *ISWC*, pages 108–126.

Kelvin Guu, John Miller, and Percy Liang. 2015. Traversing knowledge graphs in vector space. In *EMNLP*, pages 318–327.

Guoliang Ji, Shizhu He, Liheng Xu, Kang Liu, and Jun Zhao. 2015. Knowledge graph embedding via dynamic mapping matrix. In *ACL*, volume 1, pages 687–696.

Diederik P Kingma and Jimmy Ba. 2015. Adam: A method for stochastic optimization. In *ICLR*.

Ni Lao, Tom Mitchell, and William W Cohen. 2011. Random walk inference and learning in a large scale knowledge base. In *EMNLP*, pages 529–539.

Xi Victoria Lin, Richard Socher, and Caiming Xiong. 2018. Multi-hop knowledge graph reasoning with reward shaping. In *EMNLP*, pages 3243–3253.

Yankai Lin, Zhiyuan Liu, Maosong Sun, Yang Liu, and Xuan Zhu. 2015. Learning entity and relation embeddings for knowledge graph completion. In *AAAI*, volume 15, pages 2181–2187.

Zhenghao Liu, Chenyan Xiong, Maosong Sun, and Zhiyuan Liu. 2018. Entity-duet neural ranking: Understanding the role of knowledge graph semantics in neural information retrieval. In *ACL*, pages 2395–2405.

George A Miller. 1995. Wordnet: a lexical database for english. *Communications of the ACM*, 38(11):39–41.

Arvind Neelakantan, Benjamin Roth, and Andrew McCallum. 2015. Compositional vector space models for knowledge base completion. In *ACL*, pages 156–166.

Yelong Shen, Jianshu Chen, Po-Sen Huang, Yuqing Guo, and Jianfeng Gao. 2018. M-walk: Learning to walk over graphs using monte carlo tree search. In *NeurIPS*, pages 6787–6798.

Kristina Toutanova, Danqi Chen, Patrick Pantel, Hoifung Poon, Pallavi Choudhury, and Michael Gamon. 2015. Representing text for joint embedding of text and knowledge bases. In *EMNLP*, pages 1499–1509.

Kristina Toutanova, Victoria Lin, Wen-tau Yih, Hoifung Poon, and Chris Quirk. 2016. Compositional learning of embeddings for relation paths in knowledge base and text. In *ACL*, volume 1, pages 1434–1444.

Petar Velickovic, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Lio, and Yoshua Bengio. 2018. Graph attention networks. In *ICLR*.

William Yang Wang and William W Cohen. 2016. Learning first-order logic embeddings via matrix factorization. In *IJCAI*, pages 2132–2138.

Xiang Wang, Dingxian Wang, Canran Xu, Xiangnan He, Yixin Cao, and Tat-Seng Chua. 2019. Explainable reasoning over knowledge graphs for recommendation. In *AAAI*.

Zikang Wang, Linjing Li, Daniel Dajun Zeng, and Yue Chen. 2018. Attention-based multi-hop reasoning for knowledge graph. In *ISI*, pages 211–213.

Ronald J Williams. 1992. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine learning*, 8(3-4):229–256.

Wenhan Xiong, Thien Hoang, and William Yang Wang. 2017. Deeppath: A reinforcement learning method for knowledge graph reasoning. In *EMNLP*, pages 564–573.

Fan Yang, Zhilin Yang, and William W Cohen. 2017. Differentiable learning of logical rules for knowledge base reasoning. In *NeurIPS*, pages 2319–2328.