Knowledge Graph Reasoning with Reinforcement Learning

Youtube: https://www.youtube.com/watch?v=hU74jGBTGZU&feature=youtu.be
Team members: Chongyan Chen, Hari Priya Kandasamy

Introduction

Knowledge graph is a way of representing information in a structured way using nodes and edges where nodes represent any subject and edges describe the relation between them. Knowledge graph reasoning interprets combined semantic information from different sources using a graph base representing ontologies and derive new information from the existing knowledge. Various applications would be more promoted if knowledge graph reasoning is applied, e,g., recommendation system, visual question answering, and structured search.

In recent years, several approaches have been proposed for reasoning graphs including the Path-Ranking Algorithm (PRA) (Lao et al. 2011), Recurrent Neural Networks + PRA, (Neelakantan et al, 2015). The Traditional knowledge graph reasoning methods like PRA follow a discrete setup and use random walks to extract the paths and supervised learning for ranking and Policy gradient methods like REINFORCE operate in a continuous space.

The motivation of the paper is to reproduce and extend DeepPath (Xiong et. al 2018) which implements supervised policy learning (which optimizes the problem after converting it to parameterized policy) and policy gradient methods REINFORCE to reason the graph and identify the improvements of the model due to the second stage retraining leveraging.

The main issue with DeepPath is that they use REINFORCE, a basic policy gradient method leading to poor convergence. REINFORCE causes sample inefficiency taking an episode to do the update. However, there are more advanced state-of-the-art approaches like PPO and Actor critic methods which give much better results leading to convergence. DeepPath also does not perform a fine-grained analysis of the experiments carried out.

Thus, We implement Proximal Policy Optimization(PPO), an on-policy method that helps overcome the challenge of REINFORCE having a high variability, by following a stochastic policy to sample the action space and also integrate supervised policy learning with PPO. We leverage these three algorithms for three tasks: (1) link prediction: predict the links between the nodes, (2) node prediction: find the target entity, given the current node and path, and (3) fact prediction: ranking the true triples among false triples to indicate whether the facts are true or false. We also performed a fine-grained analysis of the approaches and presented findings.

Our proposed research questions are (1) Can knowledge graph reasoning be improved by reinforcement learning compared to other state-of-the-art knowledge graph reasoning algorithms? (2) Does the length reward function and diverse reward function proposed in DeepPath really help for the KG reasoning problem?

The main contributions of this paper are five fold: i) We implemented different tasks in knowledge graph reasoning which gives us a glimpse of the performance of the Reinforcement Learning algorithm in the knowledge graph reasoning domain. ii) We highlight the significance of using supervised policy learning along with Policy gradient methods for overcoming the challenge of a bigger action space in the knowledge graph setting. iii) We evaluate the reward functions for KG reasoning iv) We explored the SOTA RL algorithm, PPO-Clip on KG reasoning v) We provided fine-grained analysis for the experiments and and compared each algorithms.

Literature Review

Traditional Path-Finding Approach for Knowledge Graph

Random Walk. Path-Ranking Algorithm [12] is based on random-walk. One of its drawbacks is that it can only be used for discrete spaces and is not suitable for the VQA area where some of the entities and relations could be similar. [4] Another drawback is that it requires CPU intense operations which could get expensive.

CNN for multi-hop reasoning. [2] provides a CNN approach for multi-hop reasoning based on paths in the dependency tree joining two entities with lexical items and dependency paths. The work on link prediction is improved by the usage of continuous representations for entity pairs with textual relations. The main drawback is that this approach does not model the relational path explicitly and has issues when it comes to control and flexibility

RL Knowledge Graph Path-Reasoning Approach

DeepPath for knowledge graph path-reasoning. DeepPath[1] takes into account the precision, pathfinding using similarity measures and producing useful results all at once by following an agile process to encourage the agent to find paths with varying syntax and semantics. Unlike the traditional path-finding approach, it operates in a continuous space.The complex relation set makes it harder for the agent to find reliable paths.

REINFORCE for knowledge graph path-reasoning. [3][4][9] are three methods that leverage REINFORCE for path reasoning. DAPath [3] model uses one-pass training to find reliable paths and there are different rewards depending on the position of the agent. Hierarchical Reinforcement Learning (HRL) [4] is a concept where the problem is broken down into sub-problems where the agent makes sense of the latest semantics through a course of action. Every step is split into high-level and low-level policies where past history is transferred and the clusters are identified based on the similarity or the link between them. Then the agent estimates a path from the initial entity to the goal entity, and traverses to the next entity which has the highest chance of reaching the goal, with the help of the policies. Graphhopper[9] uses REINFORCE to maximize the reward function obtained after sequential transitions from one node to another until the result for the question is obtained.

Action-Critic for knowledge graph path-reasoning. ADRL [10] is an Action-Critic algorithm for path-reasoning that first uses CNN to project the knowledge graph to a low-dimensional state, then uses LSTM to generate the trajectories. Next, self-attention mechanism is leveraged to obtain and share the weights of the nodes and edges. Finally, the actor-critic algorithm is used to optimize the whole framework.

SOTA Reinforcement Learning

Proximal Policy Optimization. Given an MDP, proximal policy optimization(PPO) [13] alters the policy stochastically in a step-by-step manner leading to maximized rewards. The action selection is done by exploration at the starting stage and gradually shifts to exploit the information based on the update rule. PPO-Penalty automatically modifies the penalty function to strike the balance between exploration and exploitation.PPO-clip changes the policy rapidly. The main challenge here is to make sure it doesn't have a huge difference between the old and new policy. Early stopping is a technique that can be implemented to ensure that the algorithms stop taking steps beyond a limit. We leverage Proximal Policy Optimization as one of our methods.

Soft Actor-Critic. Generally, online policy methods have relatively poor sample efficiency while offline policy methods tend to be unstable. Soft actor-critic [14], being an off-policy method, can handle high sample complexity and is without brittle convergence issues. The main reason is that it aims to maximize not only the expected rewards, but also the entropy. By maximizing the entropy, the policy acts randomly as much as possible, and thus, tends to be very stable. Together with this property and its ability to handle complexity, soft actor-critic is suitable for a real-world problem. We leverage the Soft Actor-Critic method as one of our methods.

Graphhopper. Graphhopper is the first and only model that is designed for VQA scene graph reasoning. Though Graphhopper [9] and DeepPath both conduct a policy-guided random walk to reach the answer, Graphhopper is different from DeepPath in that the query in DeepPath is symbolically structured while the query in Graphhopper is free-form. The main idea of Graphhopper is that it uses LSTM to define the history-dependent action distribution and leverage REINFORCE to receive the largest expected rewards. Grapphopper[9] provides open-source code to generate scene graphs for the GQA dataset. Our work is based on [9]'s environment settings.

Our Work is Different: To date, research about applying RL to knowledge graph pathfinding problem are still utilizing some old-fashioned RL methods, such as REINFORCE and Actor-Critic. Our work is different in that we are leveraging the SOTA RL methods- proximal policy optimization [13]. In addition, the DeepPath didn't discuss the discussed effect of supervised policy leaning explicitly and didn't verify the importance of each reward function while we explore this aspect, which offers new insights into RL for knowledge graph pathfinding.

Methodology

In this section, we formulate the following tasks for knowledge graph reasoning.These tasks can be modeled as a reinforcement learning problem following sequential decision making.

i) link prediction - Finding the links between the initial and the target entity,
ii) fact prediction - predicting whether an unknown fact is true or false,
iii)node prediction - predicting the target node

Based on (Xiong et. al, 2017)'s framework, the above tasks are learned using the methods outlined below.

KG reasoning Environment

The KG reasoning environment can be viewed as a Markov Decision Process and the link prediction problem is sequential: An RL model, starting from the source nodes, needs to take sequential decisions to reach the target node. At each step, it selects an outgoing edge and transitions to a new node. When it reaches the target node, the episode ends. For example, hierarchical approaches [4] are designed to reach the next entity which is expected to have high chance of reaching the target entity.

State space

A state-space detects the position of the agent in the environment. The entities and the knowledge graph denote atom symbols.The entity and links are entitled by capturing the lingual data of symbols using translation-based embeddings which connect every symbol to a linear space with fewer dimensions.The state is given by $(e_t, e_{target} - e_t)$ where et is the embedding of the current node and e_{target} is the embedding of the target node.

Action Definition

In general, the action space is all the relations between nodes in the knowledge graph [1][3][6]. Each relation type is an action. In another word, taking an action is defined as choosing a relation path to take one step further. A valid action is where there is a valid relation and the path can be extended by one step while invalid action is when a valid relation does not exist and the path can not be extended.

The goal of the agent for the link prediction task is to find paths between the nodes that are promising. An agent uses a policy network to choose a relation that expands its path to each step leading to the goal, connecting the source and the target entity. An action space is defined as A={r1,r2,.....,rn}where r1,r2,rn are all relations found in the knowledge graph.

Environment

The environment following a Markov Decision Process with a state space S, actions A and a reward function R(s,a) specifies the communication of the agent with the knowledge graph to transition to a new state from the current state. The agent follows a policy distribution connecting the vector state to a stochastic policy. Since the agent is able to learn a stochastic policy, policy-based methods are preferred for our problem as the complexity of relation space leads to poor convergence in knowledge graphs.

Reward Function

Overall, we will give a positive reward if the action receives the target node/finds the correct path/predicts the fact correctly, otherwise it will get a non-positive reward. The reward function contains two factors: global accuracy and path efficiency (Xiong et al., 2018). The global reward is 1 if it reaches the target node. Otherwise, it gets -0.05 as its reward. The path efficiency is 1 divided by the length of the path because shorter paths tend to be more reliable and are more efficient.

Algorithm

REINFORCE

We use REINFORCE to boost the expected value of the overall reward the agent receives after one episode using Monte Carlo simulation

The objective function is J(θ) = Ea~π(a|s;θ) (X t Rst,at)

where J denotes the total reward after one episode.

The main problem with Monte Carlo REINFORCE is high variability leading to poor convergence. This could be improved by advanced policy gradient algorithms.

Proximal Policy Optimization(PPO)

Proximal Policy Optimization (Schulman et al., 2017**)** is a policy gradient method that is based on the idea of improving the policy using data. The main idea of PPO is to keep the new policy close to the old policy for each update denoted by a probability ratio r(θ). PPO restricts the policy update at each step to address the issues of variance and convergence time. An advantage value(given by the difference between the predicted value and the discounted reward) is used to assess if an action taken is good. A positive advantage value A_k.indicates that the agent has taken a good action and the gradient is updated to reuse the action when we come across the state in the future. The Advantage values are obtained by running the policy and a gradient update is made on the policy network by taking a maximum of the objective function over k epochs.

$$\theta_{k+1} = \arg\max_{\theta} \frac{1}{|\mathcal{D}_k|T} \sum_{\tau \in \mathcal{D}_k} \sum_{t=0}^{T} \min \left(\frac{\pi_\theta(a_t|s_t)}{\pi_{\theta_k}(a_t|s_t)} A^{\pi_{\theta_k}}(s_t, a_t), \quad g(\epsilon, A^{\pi_{\theta_k}}(s_t, a_t)) \right)$$

$$\phi_{k+1} = \arg\min_{\phi} \frac{1}{|\mathcal{D}_k|T} \sum_{\tau \in \mathcal{D}_k} \sum_{t=0}^{T} \left(V_\phi(s_t) - \hat{R}_t \right)^2$$

Experiments

Dataset

NELL-995 dataset contains multiple-hop reasoning originating from the NELL (Never-Ending Language Learning) system. The relations without reasoning value are removed, and 200 relations that appear most frequently are selected. To enable the backward reasoning, 200 inversed paths are added by converting (h,r,t) into (t,r^{-1} , h).

Tasks

We evaluated three algorithms (supervised policy learning, supervised policy learning + REINFORCE, and supervised policy learning + PPO) on three different tasks (link prediction, node prediction, and fact prediction).

Knowledge Graph Reasoning consists of three subtasks: link prediction, node prediction, and fact prediction. Given a triple {head node, actions, tail node}:

Link prediction, also called reasoning or path-finding task, is a task which given the source node and the target node, the algorithm needs to predict the path between the source node and the target node. In link prediction task, the r and r^{-1} are removed.

Node prediction is when the head node or the tail node is missing while the action(s) is given. The task is to predict the missing node: {?, action, tail node}; {head node, action, ?}. Here, when h is given as a query, and a set of tail is ranked.

Fact prediction is a task to predict whether a given statement is true or false by ranking the true statement with generated false statements.

Evaluation Metric

The evaluation metric for link prediction is the success rate and the evaluation metric for node and fact prediction is mean average precision (MAP).

Success Rate: Given an entity pair (node_start, node_end), success is defined as finding the path that links the start node and the end node. Table 2 shows the success rate for the three algorithms.

Average Precision: Mean Average precision(MAP) is used as a metric to compare our model against the baseline model. MAP is given by

$$\mathrm{MAP} = \frac{\sum_{q=1}^{Q} \mathrm{AveP(q)}}{Q}$$

where Q is the amount of queries q and
AveP is the average precision for a query q.

Training Pipeline

Our code is based on DeepPath[1] and the implication of PPO in spinning up[2]. We have a two-stage training process, the first stage is supervised policy learning and the second stage is REINFORCE/PPO retraining. Supervised policy learning is implemented based on DeepPath (Xiong et al. 2017) to prevent poor convergence while the agent explores a large action space. The framework is to retrieve paths between the source entity es and the target entity et from the knowledge graph using search algorithms and train the agent with a policy gradient method.

Policy gradient methods constantly improve the policy by updating the parameter θ with the goal of finding the optimal policy πθ (Sutton 2018). For each subtask, e.g., “athleteHomeStadium”, finding the path for one entity pair is defined as one episode. Beginning with one starting node, one relation (action) is picked following the policy to extend the node by one step, this relation might lead to a new node, or lead to a failed step.

---

[1] https://github.com/xwhan/DeepPath; https://github.com/Ingvarstep/DeepPath_PyTorch[1]

[2] https://github.com/openai/spinningup/blob/master/spinup/algos/pytorch/ppo/ppo.py

Results

Algorithm Time Efficiency

It takes roughly 12-13 seconds to run one episode. For the first stage, in total, it takes 4 hours to train the first stage and the second stage for one task.

Link Prediction

Algorithms	Success Rate
Supervised Policy Learning (global+length reward)	3.7%
Supervised Policy Learning+ REINFORCE (global+length reward)	5.3%
Supervised Policy Learning+ REINFORCE (global reward)	6%
Supervised Policy Learning+ PPO (global+length reward)	4.3%

Table 1: The average success rate for AthleteHomeStadium task.

Reasons for low success rate: The NELL-95 dataset consists of 100 unique nodes and 200 unique relations. Following (Xiong et al. 2017), the 200 unique relations are inversed to create another 200 relations. Thus, there could be 400^{10} invalid paths.

Performance of PPO: We suspect the reason that leads to the poor performance of PPO is that it needs a dense reward system for coverage while the NELL-995 environment has a sparse reward system with reward = 0.1*global_reward + 0.8*length_reward + 0.1*diverse_reward if the path is found else -0.05.

Exploring reward function: does length reward really matters?
The answer is probably no. Due to time limitations, we only tried solely using global reward and find that it gets 6% for the success rate for supervised policy learning + REINFORCE (other parameters and settings are the same). This finding indicates a proper reward function matters and the length reward function might not be recommended.

We show the qualitative results for the link prediction task in Table 2.

Relation	Reasoning Path

AthleteHomeStadium	concept:athleteplaysforteam -> concept:teamhomestadium concept:athleteledsportsteam -> concept:teamhomestadium concept:coachesteam -> concept:teamhomestadium concept:personbelongstoorganization -> concept:teamhomestadium
AgentBelongsToOrganization	concept:agentcollaborateswithagent -> concept:agentcollaborateswithagent_inv -> concept:agentcompeteswithagent_inv concept:proxyfor -> concept:agentcontrols -> concept:agentcontrols concept:subpartof -> concept:subpartof_inv -> concept:agentcontrols_inv concept:agentcollaborateswithagent -> concept:agentcollaborateswithagent -> concept:agentcontrols
AthleteplaysInLeague	concept:athleteflyouttosportsteamposition -> concept:sporthassportsteamposition_inv -> concept:athleteplayssport_inv -> concept:personbelongstoorganization -> concept:agentcompeteswithagent -> concept:agentcompeteswithagent -> concept:agentcompeteswithagent -> concept:agentbelongstoorganization

Table 2: Examples of the reasoning path found by the Supervised Policy Learning + REINFORCE algorithm . "_inv" denotes the inverse of existing relations.

Fact Prediction

Algorithms	**MAP**
Supervised Policy Learning+REINFORCE	35.95%
TransE	23.21%
TransR	28.13%
TransH	25.28%
TransD	23.89%

Table 3: MAP for "ConceptWorkFor" relation on fact prediction task

Table 3 shows that the RL model performs better than other embedding based algorithms for the "ConceptWorkFor" relation on the fact prediction task. Since Xiong et al. (2017) reported the overall result of the fact prediction task, we here report one concrete example of this fact prediction task as a supplement.

Node Prediction

Algorithms	MAP
Supervised Policy Learning+REINFORCE	89.04%
TransR	72.21%
TransE	71.75%

Table 5: MAP for AthleteHomeStadium link prediction task. Eleven paths are used.

As shown in Table 5, The Supervised Policy Learning + REINFORCE algorithm outperforms other embedding-based models for the AtheleteHomeStadium link prediction task.

Relationships between the number of reasoning paths and MAP scores:

We run a linear regression to see if there is a relationship between the number of reasoning paths and the MAP scores. The slope is 0.0004, the intercept is 0.8, the r-value is 0.033, the p-value is 0.9 >0.05, and the standard deviation error is 0.043. Supervising, there is no significant relationship between a number of reasoning paths and MAP scores.

Conclusion

In this paper, we reproduced and extended the experiments of the DeepPath algorithm (Xiong et al., 2017). Our contributions are five-fold. Firstly, Xiong et al. only present the result after REINFORCE retraining (the second stage) without explicitly illustrating the effect of supervised policy learning (the first stage) while we explicitly present the improvement from the first stage to the second stage. Secondly, since REINFORCE is an old-fashion RL algorithm, we extended the experiment by leveraging PPO-Clip, a more advanced on-policy RL algorithm as another baseline. Thirdly, we examined the length reward and the global reward and doubted the importance of the length reward. Fourthly, we explored the relationship between a number of paths and MAP for the node prediction task. Finally, we provided fine-grained analysis for the experiments and presents results that are not covered in Xiong et al.'s paper as supplements.

For future work, we will try other reward functions, such as *distance-aware (DA)* following DAPath [3] as well as ConvE following MemoryPath [6]. We would also explore other SOTA algorithms such as soft actor-critic, apply the algorithm to the VQA domain, and conduct more fine-grained analyses such as plotting the performance distribution and providing aggregate metrics, such as Median, IQM, and Optimality Gap for the scores.

References

[1] Xiong, W., Hoang, T., & Wang, W. Y. (2017). Deeppath: A reinforcement learning method for knowledge graph reasoning. *arXiv preprint arXiv:1707.06690*.

[2] Toutanova K.,Chen D., Pantel Patrick. (2015). Representing Text for Joint Embedding of Text and Knowledge Bases. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 1499–1509, Lisbon, Portugal. Association for Computational Linguistics.

[3] Tiwari, P., Zhu, H., & Pandey, H. M. (2021). DAPath: Distance-aware knowledge graph reasoning based on deep reinforcement learning. *Neural Networks*, *135*, 1-12.

[4] Wan, G., Pan, S., Gong, C., Zhou, C., & Haffari, G. (2021, January). Reasoning like human: Hierarchical reinforcement learning for knowledge graph reasoning. In *Proceedings of the Twenty-Ninth International Conference on International Joint Conferences on Artificial Intelligence* (pp. 1926-1932).

[5] Zheng, P., Xia, L., Li, C., Li, X., & Liu, B. (2021). Towards Self-X cognitive manufacturing network: An industrial knowledge graph-based multi-agent reinforcement learning approach. *Journal of Manufacturing Systems*, *61*, 16-26.

[6] Li, S., Wang, H., Pan, R., & Mao, M. (2021). MemoryPath: A deep reinforcement learning framework for incorporating memory component into knowledge graph reasoning. Neurocomputing, 419, 273-286.

[7]Lin, X. V., Socher, R., & Xiong, C. (2018). Multi-hop knowledge graph reasoning with reward shaping. *arXiv preprint arXiv:1808.10568*.

[8] Chen, X., Jia, S., & Xiang, Y. (2020). A review: Knowledge reasoning over knowledge graph. Expert Systems with Applications, 141, 112948.

[9] Koner, R., Li, H., Hildebrandt, M., Das, D., Tresp, V., & Günnemann, S. (2021, October). Graphhopper: Multi-hop Scene Graph Reasoning for Visual Question Answering. In International Semantic Web Conference (pp. 111-127). Springer, Cham.

[10] Wang, Q., Hao, Y., & Cao, J. (2020). ADRL: An attention-based deep reinforcement learning framework for knowledge graph reasoning. Knowledge-Based Systems, 197, 105910.

[11] Lin, X. V., Socher, R., & Xiong, C. (2018). Multi-hop knowledge graph reasoning with reward shaping. arXiv preprint arXiv:1808.10568.

[12] Lao, N., Zhu, J., Liu, L., Liu, Y., & Cohen, W. W. (2010). Efficient relational learning with hidden variable detection. Advances in Neural Information Processing Systems, 23.

[13] John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, Oleg Klimov(2017).Proximal Policy Optimization Algorithms.arXiv:1707.06347.

[14] Haarnoja, T., Zhou, A., Abbeel, P., & Levine, S. (2018, July). Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. In International conference on machine learning (pp. 1861-1870). PMLR.

[15] Antol, S., Agrawal, A., Lu, J., Mitchell, M., Batra, D., Zitnick, C. L., & Parikh, D. (2015). Vqa: Visual question answering. In Proceedings of the IEEE international conference on computer vision (pp. 2425-2433).

[16] Gurari, D., Li, Q., Stangl, A. J., Guo, A., Lin, C., Grauman, K., ... & Bigham, J. P. (2018). Vizwiz grand challenge: Answering visual questions from blind people. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (pp. 3608-3617).

[17] Anderson, P., He, X., Buehler, C., Teney, D., Johnson, M., Gould, S., & Zhang, L. (2018). Bottom-up and top-down attention for image captioning and visual question answering. In Proceedings of the IEEE conference on computer vision and pattern recognition (pp. 6077-6086).