DeepPath - A Deep Reinforcement Learning Method for Knowledge Graph Reasoning using TensorForce

Aruna Neervannan

University of Illinois, Urbana Champaign

Abstract

Implement a large scale knowledge graph using a reinforcement learning library, TensorForce to learn multi-hop relational paths using a policy-based agent, additionally experimenting with other agents available in the environment. This approach includes a reward function that takes the accuracy, diversity and efficiency into consideration.

Keywords Reinforcement Learning, knowledge graph, TensorForce, Policy Gradients,

1 Introduction

The idea that we learn by interacting with our environment is probably the first to occur to us when we think about the nature of learning. When an infant plays, waves its arms, or looks about, it has no explicit teacher, but it does have a direct sensori-motor connection to its environment. Exercising this connection produces a wealth of information about cause and effect, about the consequences of actions, and about what to do in order to achieve goals. This approach is much more focused on goal-directed learning from interaction and is called Reinforcement Learning [1].

2 Purpose

The purpose of this project is to study the problem of learning to reason in large scale knowledge graphs (KGs). More specifically, I describe a novel reinforcement learning framework for learning multi-hop relational paths: I use a policy-based agent with continuous states based on knowledge graph embeddings, which reasons in a KG vector-space by sampling the most promising relation to extend its path. In contrast to prior work, our approach includes a reward function that takes the accuracy, diversity, and efficiency into consideration. Experimentally, I show that our proposed method outperforms a path-ranking based algorithm and knowledge graph embedding methods on Freebase and Never-Ending Language Learning datasets

2.1 Motivation

In recent years, deep learning techniques have obtained many state-of-theart results in various classification and recognition problems (Krizhevsky et al., 2012; Hinton et al., 2012; Kim, 2014). However, complex natural language processing problems often require multiple inter-related decisions, and empowering deep learning models with the ability of learning to reason is still a challenging issue. To handle complex queries where there are no obvious answers, intelligent machines must be able to reason with existing resources, and learn to infer an unknown answer.

2.2 Knowledge Graph

Knowledge graph, a concept introduced by Google, mainly describes real world entities and their interrelations, organized in a graph covering various topical domains. Knowledge is about collecting infor-

mation about objects in the real world. The object could be a person, book, movie or any other type of objects. Knowledge graphs help improve search results by understanding what the user is searching for. It acquires and integrates information into an ontology (which describes the semantics of the data) and applies a reasoner to derive new knowledge.
Knowledge graphs being actual graphs, in the proper mathematical sense, allow for the application of various graph-computing techniques and algorithms (for example, shortest path computations, or network analysis), which add additional intelligence over the stored data. The ontology can be extended and revised as new data arrives. With these semantics defined, the results could go deeper and broader resulting in unexpected findings.

2.3 DeepPath

DeepPath[2] implements a knowledge graph using a supervised policy with continuous states based on knowledge graph embeddings and goes through a training phase similar to the technique used by AlphaGo. The supervised policy is trained with a randomized breadth-first search (BFS) making it very similar to the Model-based RL but not with the goal of model finding. The goal here is to find the optimal path between the source and target by conducting a two-sided BFS and maximizing the cumulative reward using Monte-Carlo Policy Gradient.

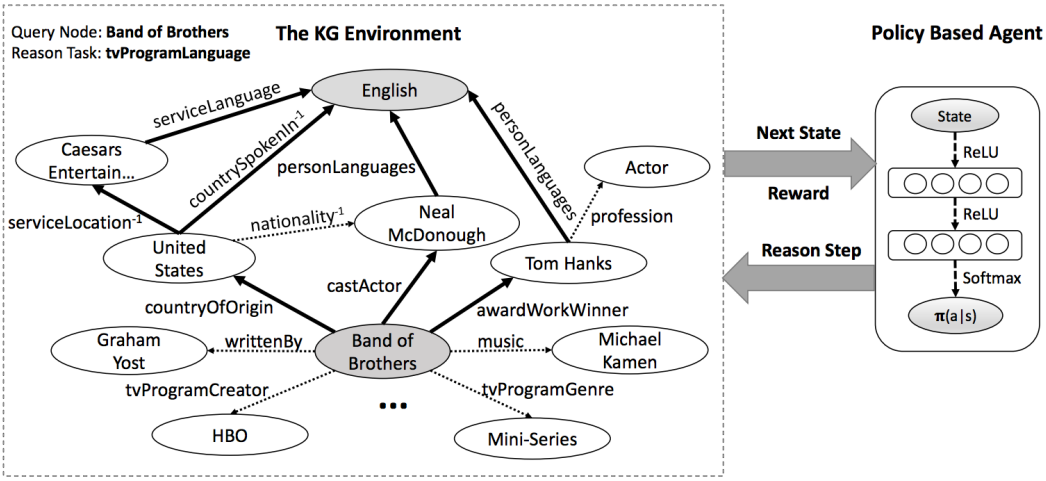

Figure 1: Overview of our RL model. **Left:** The KG environment $\mathcal{E}$ modeled by a MDP. The dotted arrows (partially) show the existing relation links in the KG and the bold arrows show the reasoning paths found by the RL agent. $^{-1}$ denotes the inverse of an relation. **Right:** The structure of the policy network agent. At each step, by interacting with the environment, the agent learns to pick a relation link to extend the reasoning paths.

DeepPath proposes a novel approach for controllable multi-hop reasoning: the path learning process is framed as a reinforcement learning (RL) problem. In contrast to PRA (Path Ranking Algorithm) proposed by Lao et all, 2010, which uses a random-walk with restarts based on an inference mechanism to perform multiple bounded depth-first search processes to find relational paths, DeepPath uses translation based knowledge based embedding method (Bordes et al., 2013) to encode the continuous state of our RL agent, which reasons in the vector space environment of the knowledge graph. The agent takes

incremental steps by sampling a relation to extend its path. To better guide the RL agent for learning relational paths, we use policy gradient training (Mnih et al., 2015) with a novel reward function that jointly encourages accuracy, diversity, and efficiency. Empirically, we show that our method outperforms PRA and embedding based.

This implementation of DeepPath using TensorForce removes the supervised learning from the picture. The agent learns from its environment and finds the optimal paths between entities at the same time trying to maximize the reward.

3 Reinforcement Learning for Relation Reasoning

The RL system consists of two parts (see Figure 1). The first part is the external environment E which specifies the dynamics of the interaction between the agent and the KG. This environment is modeled as a Markov decision process (MDP). A tuple $< S, A, P, R >$ is defined to represent the MDP, where S is the continuous state space, A = a1, a2, ..., an is the set of all available actions, P(St+1 = s0—St = s, At = a) is the transition probability matrix, and R(s, a) is the reward function of every (s, a) pairs.

Relation	Reasoning Path
filmCountry	filmReleaseRegion featureFilmLocation → locationContains^{-1} actorFilm^{-1} → personNationality
personNationality	placeOfBirth→ locationContains^{-1} peoplePlaceLived → locationContains^{-1} peopleMarriage → locationOfCeremony → locationContains^{-1}
tvProgramLanguage	tvCountryOfOrigin → countryOfficialLanguage tvCountryOfOrigin → filmReleaseRegion^{-1} → filmLanguage tvCastActor → filmLanguage
personBornInLocation	personBornInCity graduatedUniversity → graduatedSchool^{-1} → personBornInCity personBornInCity → atLocation^{-1} → atLocation
athletePlaysForTeam	athleteHomeStadium → teamHomeStadium^{-1} athletePlaysSport → teamPlaysSport^{-1} athleteLedSportsTeam
personLeadsOrganization	worksFor organizationTerminatedPerson^{-1} mutualProxyFor^{-1}

Table 5: Example reasoning paths found by our RL model. The first three relations come from the FB15K-237 dataset. The others are from NELL-995. Inverses of existing relations are denoted by $^{-1}$.

The second part of the system, the RL agent, is represented as a policy network π_θ(s, a) = p(a s; θ) which maps the state vector to a stochastic policy. The neural network parameters are updated using stochastic gradient descent. Compared to Deep Q Network (DQN) (Mnih et al., 2013), policy-based RL methods turn out to be more appropriate for the knowledge graph scenario. One reason is that for the path

finding problem in KG, the action space can be very large due to complexity of the relation graph. This can lead to poor convergence properties for DQN. Besides, instead of learning a greedy policy which is common in value-based methods like DQN, the policy network is able to learn a stochastic policy which prevent the agent from getting stuck at an intermediate state.

4 Dataset

Used the NELL-995 dataset that is suitable for multi-hop reasoning from the 995th iteration of the NELL system. The triples with relation *generalizations* or *haswikipediaurl* were removed. These two relations appear more than 2M times in the NELL dataset, but they have no reasoning values. Only the triples with Top-200 relations are chosen. To facilitate path finding, we also add the inverse triples. For each triple (h,r,t), $(t,r1,h)$ were appended to the datasets. With these inverse triples, the agent is able to step backward in the KG.
For each reasoning task r_i, all the triples with r_i or r_i^{-1} were removed from the KG. These removed triples are split into train and test samples. For the link prediction task, each h in the test triples (h,r,t) is considered as one query. A set of candidate target entities are ranked using different methods. For fact prediction, the true test triples are ranked with some generated false triples

5 TensorForce

TensorForce is a python-based reinforcement learning (RL) library built on top of TensorFlow. As noted in previous section, TensorFlow is an open source machine learning software library developed by Google. Reinforcement learning is evolving rapidly but it is also very hard to learn. They main objective of this library is to modularize various elements of machine learning. Reinforcement learning is an active area of research. However, the practical usage of RL based applications has demonstrated following issues consistently.

- Most implementations are custom built for a specific application. It means that RL code and simulation environment code are typically tightly coupled making it harder to separate when it is time to exit prototyping phase and take it to production.

- We have observed most implementations have custom coded the neural network for training. While it is convenient to do that for a researcher, it becomes very difficult for a future researcher to plugin a different type of neural network or simply remove it.

- The state and action options for an AI environment are typically unique which typically leads to implementation of an agent good enough for that particular environment. The authors of this library identified this problem early on and decided to build an agent capable of handling any number of states and/or actions.

TensorForce allows researchers to use declarative framework to implement deep reinforcement learning algorithms. Instead of describing agent is described using configuration object. Similarly, the neural network for training is defined as an ordered list of layers. It is easier to why the object based declarative interface facilitates a modular approach for RL algorithms. At the time of writing this document, following RL algorithms are available out of the box.

- Random Baseline agent
- Vanilla policy gradient
- Trust Region Policy
- Deep Q-Learning/ double deep Q-Learning
- Normalized advantage functions
- Asynchronous advantage actor-critic (A3C) there is no A3CAgent. A3C describes a mechanism for asynchronous updates.

It is important to note that TensorForce enforces the separation between model and agent. The Agent class provides APIs to use RL while the model class implements the core RL algorithms. Another key feature of this library is the ability to define various components as a JSON object. For example, the neural network configuration can be stored as a JSON object in a separate file. This separation enforces and encourages modular framework and provides an easy way to swam out neural networks without having to rewrite the code or logic.

The choice of TensorForce library for the DeepPath project is a logical choice. Researchers have implemented the environment, actions, rewards, agent and states in custom python libraries. TensorForce API library will allow us to modularize the implementation of DeepPath. Our work will empower future researchers to swap out select components without having to worry about doing significant code rewrite. In the next section, we will provide a brief overview of OpenAI gym an open source platform for building and testing agents in a gaming environment.

6 Conclusion and Future Work

The environment was setup in TensorForce and initialized the states and the actions and the agent was allowed to learn with the goal of finding the optimal path by maximizing the reward signal. A RL agent was trained to find reasoning paths in the knowledge base. Unlike previous path finding models that are based on random walks, the RL model allows us to control the properties of the found paths. These effective paths can also be used as an alternative to PRA in many path-based reasoning methods. For two standard reasoning tasks, using the RL paths as reasoning formulas, our approach generally outperforms two classes of baselines.

Using TensorForce for the implementation removed the supervised learning aspects of DeepPath and made it a complete reinforcement learning system while the results expected did not match that from the original paper but it is certainly the next step in using reinforcement learning to understand relationships and find shortest/optimal paths in knowledge graphs. Future work will involve experimenting with the policy gradients and also letting the agent learn for a longer period when computing resources for the same are available.

References

[1] Reinforcement Learning: An Introduction - Richard S. Sutton and Andrew G. Barto

[2] DeepPath: A Reinforcement Learning Method for Knowledge Graph Reasoning. (https://arxiv.org/abs/1707.06690)

[3] Mikael Henaff, Joan Bruna, and Yann LeCun. 2015. Deep Convolutional Networks on Graph-Structured Data. CoRR abs/1506.05163 (2015). arXiv:1506.05163 http://arxiv.org/abs/1506.05163

[4] Kelvin Guu, John Miller, and Percy Liang. 2015. Traversing knowledge graphs in vector space. In EMNLP.

[5] Yankai Lin, Zhiyuan Liu, Maosong Sun, Yang Liu, and Xuan Zhu. 2015. Learning entity and relation embeddings for knowledge graph completion. In AAAI, pages 21812187.

[6] Zhen Wang, Jianwen Zhang, Jianlin Feng, and Zheng Chen. 2014. Knowledge graph embedding by translating on hyperplanes. In AAAI, pages 11121119. Citeseer.

[7] Daojian Zeng, Kang Liu, Siwei Lai, Guangyou Zhou, Jun Zhao, et al. 2014. Relation classification via convolutional deep neural network. In COLING, pages 23352344.