Enhanced DeepPath

An Enhanced Reinforcement Learning Method for Knowledge Graph Reasoning

# Knowledge Graph

Semantic Networks were invented to address the growing need for a knowledge representation framework that can capture a wide range of entities — real-world objects, events, situations or abstract concepts and relations.

Every Company/Group/Individual creates their own version of the Knowledge Graph to limit complexity and organize information into data and knowledge such as Google’s Knowledge Graph. But Knowledge Representation brings the ability to represent entities and relations with high reliability, explainability, and reusability. [1]

Knowledge graphs can be used in QA, recommender systems, search engines, etc.

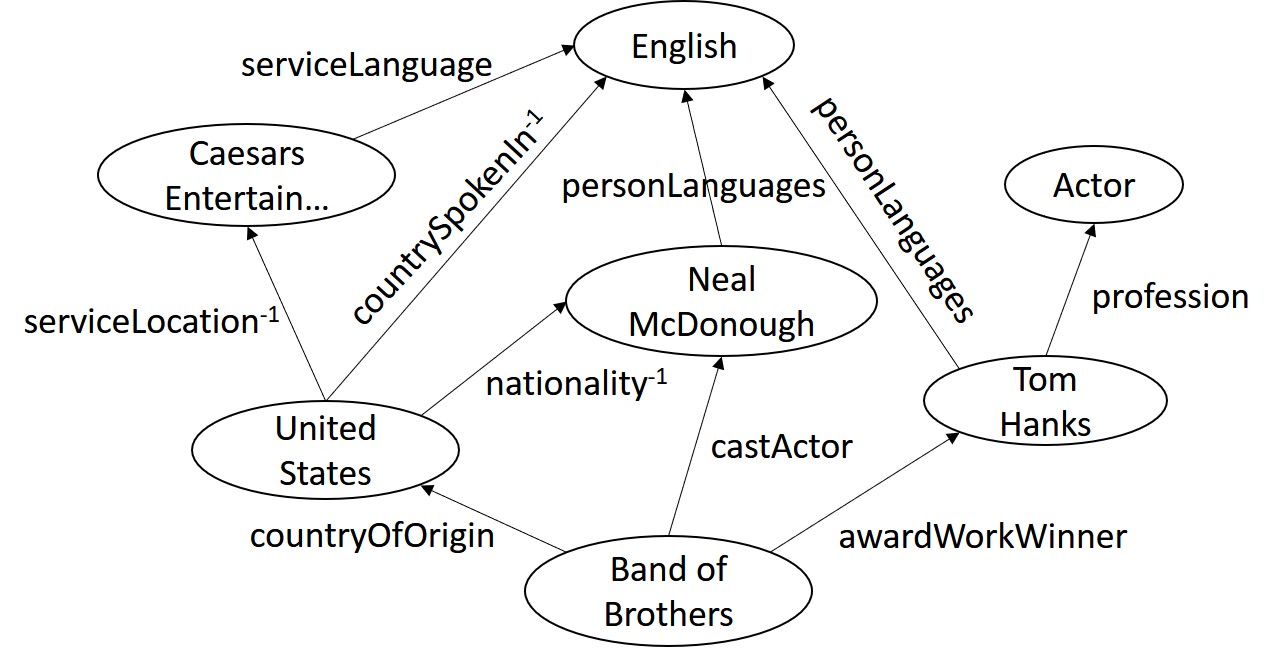


Figure 1 Knowledge Graph

# Reasoning on Knowledge Graph

What we are dealing with is the triples of the entity-relations in form of *subject-predicate-object*. Reasoning in KG is the ability to calculate the set of triples that logically follow from a knowledge graph and a set of rules. Such logical consequences are materialized as new triples in the graph. [2]

Consider following rules:

* Berlin is in Germany
* Germany is in Europe

As the relation in is intuitively transitive, then we can obtain a rule such as *Berlin is in Europe*. However, the triplet *Berlin is in Europe* is missing from KG. We can obviously add this rule by hand to complete graph, but these removes some benefits such as:

* Rules are too many and this task manually can be cumbersome
* We are not using the implicit logic of being transitive for rules such as in.

in

in

in

Figure 2 Reasoning on KG

# Datasets

There are two datasets that has been used in this experiment. Both of these datasets are subset of larger datasets.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Dataset** | **# of Entities** | **# of Relations** | **# of Triples** | **# of Tasks** |
| FB15k-237[3] | 14,505 | 237 | 310,116 | 20 |
| NELL-995 | 75,492 | 200 | 154,213 | 12 |

Figure 3 Datasets

**FB15k-237:** Constructed from FB15k (Bordes et al., 2013), redundant relations removed

**NELL-995:** Constructed from the 995th iteration of NELL system (Carlson et al., 2010b)

For both of these datasets, following processing steps have been incorporated:

* Remove useless relations: *haswikipediaurl*, *generalizations, …*
* Add inverse relation links to the knowledge graph (-1 superscripts)
* Remove the triples with task relations

They perform the reasoning tasks on 20 relations which have enough reasoning paths. These tasks consist of relations from different domains like Sports, People, Locations, Film, etc.

To facilitate path finding, they also add the inverse triples. For each triple , they append  
 to the datasets. With these inverse triples, the agent is able to step backward in the KG.

For each *reasoning task* , we remove all the triples with or from the KG. These removed triples are split into train and test samples.   
For the *link prediction task*, each in the test triples 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.

# DeepPath

In this section, the different aspects of DeepPath paper will be discussed as the core of adopting RL in KG reasoning. In summary, the proposed approach can be expressed by following:

* Learning the paths instead of using random walks
* Model the path finding as an MDP
* Train a RL agent to find paths
* Using a reward function that considers accuracy, efficiency, and diversity simultaneously
* Use the learned paths as horn clauses

## Related Works

Related works in this area can be dichotomized into two main categories:

1. Path based methods: Which usually find potential path types between entities pairs then compute rank of paths using Random Walk. The main issues with these approaches are that in the dense area of graph where a node is connected to a large number of nodes, fan out dominates the algorithm and speeds decreases drastically.
2. Embedding based methods: The idea is that the vector + should be close to vector. This has been incorporated into loss function. Some of these models create too many models which make them fail to scale or even cannot model the relational path explicitly.

Note that the RL approach in DeepPath, uses embedding from TransE to embed the entities with size of 200 where is the input size of policy network too.

## Reinforcement Learning

The RL system consists of two parts; 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 is defined to represent the MDP where S, A, P and R are state space, action space, transition probability and rewards respectively. The important property in this section is that the state space is continuous which has been constructed from KG embeddings such as TransE.

The second part of the system, the RL agent, is represented as a policy network which maps the state vector to stochastic policy. The NN parameters are updated using stochastic gradient descent in . This config enables agent to learn much complex paths where greedy policies can fail.

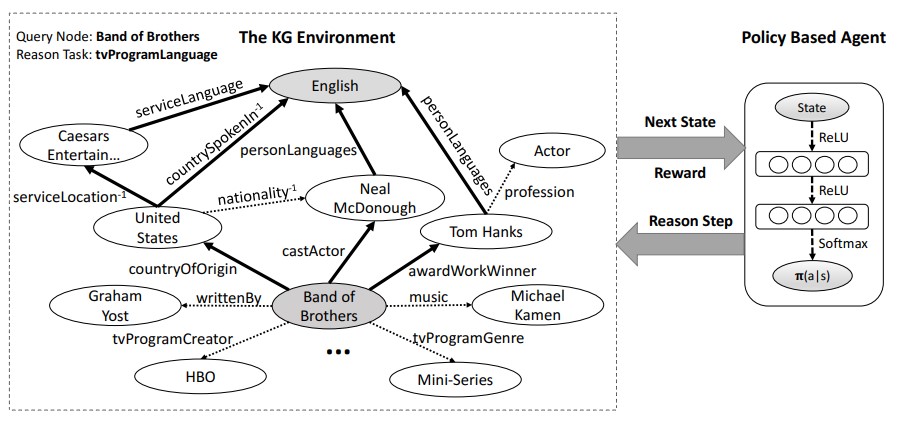


Figure 4 RL model.

On the left side, the KG environment modeled by an 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. On the right side, the structure of the policy network agent can be seen. At each step, by interacting with the environment, the agent learns to pick a relation link to extend the reasoning paths.

### Actions

Given the entity pairs with relation r, we want the agent to find the most informative paths linking these entity pairs.

### States

It is impossible to discreetly model all entities and relations which are discrete in first place. To overcome this issue, they use translation embeddings such as TransE to represent entities and relations. This embedding map all symbols into low dimensional vectors. In this model, each state holds the agent’s position in the environment . The destination and source entities (states) are linked using relations (actions). For step we have where denotes the embedding of entity.

### Reward Function

Three factors incorporate in the defined reward function to achieve accuracy, efficiency and diversity which are as follow:

* Global accuracy: As the KG can be enormously big, and due to that, agent makes many mistakes w.r.t. correct decisions which can be increased exponentially, they introduced binary outcome of success or failure.
* Path efficiency: It has been observed that shorter paths have more intuitive logics so limiting the amount of agent’s interaction with environment can help where is a sequence of relations.
* Path Diversity: The agent tends to find paths with similar syntax and semantics, which usually have high correlations because mostly are redundant. To ensure diversity, they introduce a diversity functions based on cosine similarity functions to enforce agent to learn paths far (higher distance) from what it has learned so far.  
  PS. Here is the section some improvements have been introduced by using dropout to forcing diversity.

### Policy Network

They use a fully-connected neural network to parameterize the policy function that maps the state vector s to a probability distribution over all possible actions. The neural network consists of two hidden layers, each followed by a rectifier nonlinearity layer (ReLU). The output layer is normalized using a softmax function.   
PS. Here is the section some improvements have been introduced by using dropout to forcing diversity.

## Training with Policy Gradient

## Challenge

## Supervised (Imitation) Policy Learning

## Inference Using Learned Paths

## Results

# Enhancement

The introduced enhancements are defined to work on the RL framework to just improve the path finding algorithm better. All other sections are similar to the previous framework and the only differences have been described in the following sections which exactly matches the previously defined framework DeepPath.

## Reinforcement Learning

We can use language embedding and other pretrained models to enhance our agent’s behavior. To do so, we can incorporate path finding score as part of reward or in term of policy network, we can explicitly delete some relations to enforce agent’s action to be exposed to more diversity by making more exploration which can be achieved by using dropouts in the defined policy neural network. You can find the changes in more details in the following sections.

### Reward Function

### Policy Network

# Conclusion

# References

[1] Sudip Chowdhury, Knowledge Graph: The Perfect Complement to Machine Learning, 2019, medium.

[2] Peter Crocker, The intuitions behind Knowledge Graphs and Reasoning, 2018, Towards Data Science.

[3] Toutanova et al. Representing text for joint embedding of text and knowledge bases

# Appendix

Link to deeppath source code

Link to salesforce source code

Link to my source code

## Acknowledgment

There is nothing new in my source code, all I have done is first fixed some errors and issues regarding Python 2.7 and older version of Tensorflow and merged the idea of section “enhancement” from second source code into first one. So, if there is any improvement in results, all credits belong to the real authors, I have just used theirs and constructed new blocks without mathematical verification.