Action constrained Reinforcement Learning for reasoning using Knowledge Graphs

A Dual Degree Project Report

submitted by

MOHAN BHAMBHANI

under the guidance of

PROF. BALARAMAN RAVINDRAN

DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING
INDIAN INSTITUTE OF TECHNOLOGY MADRAS.

May 2018

THESIS CERTIFICATE

This is to certify that the thesis titled **Action constrained Reinforcement Learning for reasoning using Knowledge Graphs**, submitted by **Mohan Bhambhani**, to the Indian Institute of Technology, Madras, for the award of the degree of **Bachelors of Technology** and degree of **Masters of Technology**, is a bona fide record of the research work done by him under our supervision. The contents of this thesis, in full or in parts, have not been submitted to any other Institute or University for the award of any degree or diploma.

Dr. Balaraman Ravindran
Research Guide
Professor
Dept. of Computer Science & Engineering
IIT-Madras, 600036

Place: Chennai

Date: 10th May 2018

ACKNOWLEDGEMENTS

Firstly, I would like to express my sincere gratitude to my advisor Prof. Balaraman Ravindran for the continuous support, critical suggestions, invaluable guidance, patience, motivation and immense knowledge. I would also like to thank him for the freedom to explore areas of Deep Learning, Computer Vision, Knowledge Graph Reasoning, Network Representation Learning and Reinforcement Learning.

I take this opportunity to express my deep sense of gratitude to Professor Mitesh Khapra for introducing me to Deep Learning and research in Deep Learning.

My sincere thanks to Prof. Jennifer Neville, Purdue University, for her time during Summer 2017. The research and academic guidance provided by her during my internship gave a lot of clarity to my ambitions and approach.

I am highly thankful to Abhishek, Ujjawal and Priyesh for fruitful discussions and critical suggestions. I sincerely admire the contribution of all my friends and lab mates Gurneet, Ishu, Sanchit, Tarun, Akash, Sankaran, Nikita, Shweta and Beethika for extending their support, timely motivation and discussions on life and universe during the course of entire study.

Finally, I thank my parents Premchand Bhambhani and Jagruti Bhambhani for the selfless love and care and for their sacrifices to shape my life. Also I express my thanks to my brother Govind for his support.

ABSTRACT

KEYWORDS: Reasoning; Knowledge graphs; Scene understanding; Reinforcement Learning; Deep Learning.

Reinforcement learning has been very successful in various tasks requiring sequential decision making. Traditionally in these environments the agent learns to pick a action from set of universal actions at each time step. But not all environments have same action set for all the states. We explore two tasks which involve sequential decision making where actions sets for different states are different.

The first task is multi-hop reasoning in knowledge graphs. Here, a reinforcement learning agent must learn to propose good representative paths in the knowledge graph to link prediction in the knowledge graph. We propose to improve the-state-of-the-art method by providing additional graph structure information at each step.

Another task is Visual relationship detection. Most of the state-of-the-art methods in visual relationship detection tasks ignore the semantic constraints of interactions between objects of different types. We propose to incorporate knowledge of the constraints from a Knowledge graph (that contains such constraints) by constraining the relationships that can be predicted, to a set of semantically feasible relationships using a knowledge graph. We then model relationship detection as sequential decision making problem by learning to predict relationships by performing traversal in the knowledge graph.

TABLE OF CONTENTS

ACKNOWLEDGEMENTS i

ABSTRACT ii

NOTATION vii

1 INTRODUCTION 1

1.1 Reasoning using Knowledge graphs 2

1.2 Scene Understanding 3

1.3 Reinforcement Learning 4

1.4 Contributions 5

2 Background and related work 6

2.1 Reasoning using Knowledge Graphs 6

2.2 Scene understanding 9

2.3 Action constrained Reinforcement Learning 12

2.4 Representation Learning in graph-structured data 14

3 Incorporating Graph Structure for reasoning in Knowledge Graphs 16

3.1 Motivation 16

3.2 Task 16

3.3 Background 17

3.4 Method 19

3.5 Preliminary experiments 20

3.6 Experiments 21

3.7 Results 22

4 Visual Relationship Detection 24

4.1 Motivation 24

4.2 Background 25

4.3 Task . . . 25
4.4 Method . . . 26
4.4.1 Knowledge graph construction . . . 27
4.4.2 Knowledge graph embeddings . . . 27
4.4.3 State space . . . 28
4.4.4 State constrained action space . . . 29
4.4.5 Reward . . . 30
4.4.6 Value network . . . 30
4.4.7 Training . . . 30
4.4.8 Inference . . . 31
4.5 Preliminary experiments . . . 31
4.6 Experiments . . . 32
4.6.1 Dataset . . . 32
4.6.2 Implementation details . . . 32
4.6.3 Evaluation . . . 32
4.7 Results . . . 33
4.7.1 Comparison with baseline models . . . 33

5 Conclusion **35**

LIST OF TABLES

1.1 Objects, relationships and object attributes for the example image . 4

3.1 Results for fact prediction 22

3.2 Results for link prediction 23

4.1 Convergence of Losses 32

4.2 Results 33

4.3 Predicted relationships 33

4.4 Predicted relationships 34

LIST OF FIGURES

1.1 Example 4

2.1 Action constrained Reinforcement Learning 13

3.1 BlogCatelog - Bias and variance trends for different category of nodes 20

4.1 An example 33

4.2 An example 34

NOTATION

$G(V, \mathcal{E})$	Knowledge graph, V is set of nodes and $\mathcal{E}$ set of edges
(h, r, t)	An edge in knowledge graph. h is head entity, r relation and t tail entity
$\mathbf{a} \oplus \mathbf{b}$	Concatenation of 2 vectors
$\mathbf{a} \cdot \mathbf{b}$	Dot priduct of 2 vectors

CHAPTER 1

INTRODUCTION

Artificial Intelligence (AI) is a field in Computer Science that aims to build intelligent machines that can think like humans and mimic a person. The major goals of Artificial Intelligence are learning, reasoning, and perception.

The most successful approach towards building a AI machine is machine learning. Machine Learning approaches AI with learning. The key idea of Machine Learning is that intelligence can be achieved with ability to learn automatically from experience. Tom Mitchell defines algorithms in Machine Learning as 'A computer program is said to learn from experience E with respect to some class of tasks T and performance measure P if its performance at tasks in T, as measured by P, improves with experience E.' Machine Learning is widely used in applications to images, text, speech, videos and networks.

Another approach towards building an AI machine is Knowledge Engineering. It creates rules in order to imitate the thought process of a human. Intelligent systems in this type of approach involves a large expandable Knowledge base. These Knowledge bases are then used to make inferences as a human would. It is expected that if an expert can formulate a Knowledge base that covers all the information then the system can surpass a human. But, achieving that is impossible.

Of course, neither of the two approaches are complete and lack a lot before then can achieve the goal of AI. One of the main weakness of the approaches based on Machine Learning is in incorporating world knowledge, because there is limited amount of knowledge that can be learnt. While, in Knowledge Engineering based systems can not learn to make complex inferences from the data. So, often a combination of these methods is used to build a more better system. Such a system will have access to large knowledge base and could learn complex patterns in the knowledge base to make better inferences. One way to that is learning to reason in Knowledge graphs.

A Knowledge Graph is a multi-relational graph containing entities as nodes and relations as different types of edges. Each edge is represented as a triple of the form

(entity, relation, entity). The presence of the edge means that two entities are connected by a specific relation. For example, if the edge (Messi, playsFor, Barcelona) is present in a particular Knowledge graph contains the information that Messi plays for Barcelona.

In the rest of the chapter, we introduce the problems that this thesis attempts in solving or improving and methods used in doing so. In Section 1.1 we describe reasoning in Knowledge graphs. In Section 1.2 we describe Scene Understanding. Section 1.3 contains brief introduction to Deep Learning. And Section 1.4 introduces Reinforcement Learning to the reader.

1.1 Reasoning using Knowledge graphs

Knowledge graphs contain lot of information in a structured manner and hence, they are used in many Big data systems. Google's search engine, Bing, IBM Watson all use Knowledge graphs to make the systems intelligent.

A Knowledge Graph encodes many facts, where a fact is a edge in the knowledge graph. Facts are of type $(subject, verb, object)$. Each object can have many facts associated with it. A representation like this contains a lot of information and can be used to reason about questions and queries.

In the literature many tasks have been defined on Knowledge graphs to improve reasoning using Knowledge graphs. They are Link prediction, Entity resolution and link based clustering. Link prediction is important since most of the times, Knowledge graphs do not contain all the facts and often many facts are missing. Entity resolution refers to identifying the entities in the knowledge graph being referred in the text or document. This thesis focuses on Link prediction. We will describe link prediction in more detail.

Assume there exists some 'true' knowledge base $\mathcal{K}$, which is a set of triples (e_s, r, e_t). This 'true' Knowledge graph is not observed, instead, we have some incomplete and noisy Knowledge graph $\tilde{\mathcal{K}}$. $\tilde{\mathcal{K}}$ does not contain all of the triples in $\mathcal{K}$ and may contain some triples not in $\mathcal{K}$. The goal of inference in Knowledge graphs is to find the set of missing facts $\mathcal{F} = \mathcal{K} - \tilde{\mathcal{K}}$. (Gardner, 2015)

While using Knowledge graphs for reasoning, we assume some model for the miss-

ing facts. Sometimes the assumption made is closed domain, *i.e.*, no relation exists other than the edges in the Knowledge graph. Most of the times the assumption made is open domain, *i.e.*, facts may exist that are not encoded in the network and $\mathcal{F} \neq \phi$. In such cases we assume some inherent properties about the missing edges. The methods on which this thesis is based are graph-based methods. Here, we assume that existing and missing edges follow some common structure, i.e., entities are similar if they have edges with similar relations and vice-versa. So, all the missing edges also follow this structure. Another class of methods, vector based methods, assume that there exists an edge if the learnt vectors of two entities and the relation are similar.

1.2 Scene Understanding

State-of-the-art perception models have almost perfected detection of individual objects in an image. But, completely comprehending an image is much more than just object detection. For better understanding of an image one must be able to perceive subtle interactions between objects. These models fail to recognise rich semantic relationships. Here too, recent work has shifted focus to tasks like generating image descriptions Karpathy and Fei-Fei (2015), visual relationship detection Lu *et al.* (2016) and scene graph generation Xu *et al.* (2017) for more deep understanding of images.

Structuring visual scene in form of a scene graph Johnson *et al.* (2015) has led to better understanding of images. It captures objects, their relationships and their attributes. An example scene graph has been shown in the Figure **??**. Such representation provides more information for various visual tasks like image retrieval Johnson *et al.* (2015) and also puts forward higher level visual task of generating this given a image. This has spurred interest in detection of various components of the scene graph like relationships and attributes. As defined in Xu *et al.* (2017), *scene graph* is a visually-grounded graph over the object instances in an image, where the edges depict their pairwise relationships.

Scene graph representation has been used in various visual tasks like image retrieval Johnson *et al.* (2015), 3D scene synthesis Chang *et al.* (2014), Visual question answering Teney *et al.* (2017). But the scene graph for these models is obtained from external sources like annotations. Recent work Lu *et al.* (2016) Dai *et al.* (2017) Xu

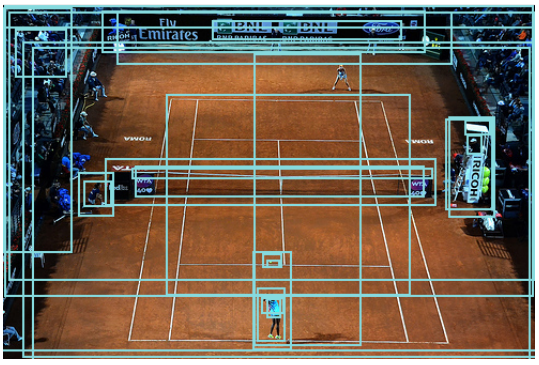


Figure 1.1: Example

Table 1.1: Objects, relationships and object attributes for the example image

Objects	Relationships			Attributes	
	Subject	Predicate	Object	Object	Attribute
Tennis court	Net	has a	strip	Tennis court	red clay
line judges	line judge	on	tennis court	ball	yellow
chair umpire	player	on	tennis court	net	black
spectators	player	serving	ball		
player	Net	on	tennis court		
player	player	wearing	light blue		
Ball boy					
Net					
ball					

et al. (2017) Liang *et al.* (2017) Newell and Deng (2017), has focused on the problem of scene graph generation or more specifically visual relationship detection. The final output scene graph of these models has objects as nodes, relationships as edges. Also, the objects in the output are visually grounded on the input image. Figure 1.1 shows an example [1] and table shows its object, relationships and object attributes.

1.3 Reinforcement Learning

Reinforcement learning is very close to how humans learn. Like humans, an agent acts in an environment and receives a payoff for its actions. From this payoff the agent must learn to maximise the total payoff. Thus an agent must learn a mapping from states to actions to maximise its rewards. This mapping is called policy. This is difficult in general as the payoffs may be delayed. So it becomes difficult to associate the payoffs with the actions that led to them.

[1] Taken from https://visualgenome.org/VGViz/explore?query=court

Reinforcement learning problem is modelled as an Markov Decision Process (MDP) $< S, A, R, P, \gamma >$. S is set of states and A set of Actions. R is a reward function which is defined on state-action pairs. P is a transition function that assigns a probability of transitioning from one state to another by taking a particular action. γ is discount factor which represents the significance of the future rewards.

There are mainly 2 type of methods for reinforcement learning. Value based methods learn a score for action in states. Then it can take action that has maximum score. On the other hand, policy based methods directly optimise policy to maximise rewards.

Good representation learning and generalising power of Deep Learning has enabled learning good policies without explicitly vising the state in high dimensional image domain. Various methods in Value based learning (Deep Q learning) and Policy based methods (A3C, DDPG) have thus been developed.

1.4 Contributions

Recently, there has been significant work on reasoning in Knowledge graphs. Many methods have been proposed very recently for multi-hop reasoning in Knowledge graphs using Reinforcement learning. Agent performs traversals in the knowledge graph and if a good path is found agent receives positive payoff. We attempt to improve these methods by adding graph structure information.

Secondly, we propose to use Knowledge graph for structural information for scene understanding. As the number of classes increase the performance of deep learning methods degrades. We propose to solve this problem by providing additional structure information in form of a knowledge graph. This provides extra information of co-occurrence of various objects and relationship.

CHAPTER 2

Background and related work

In this chapter we make the reader more familiar with the problem statement, complexities involved and various methods proposed to address them. Section 2.1 contains talks about the literature in reasoning in knowledge graphs. Section 2.2 talks more on scene understanding and various methods proposed to improve the visual understanding of an AI system. Section 2.3 talks about action constrained Reinforcement learning. It is the core problem that needs to be addressed in order to use reinforcement learning on these tasks. But, there is not much literature about it. Section 2.4 talks about representation learning methods on graph structured data that can be used to improve performance on these tasks by incorporating structure information from a graph or network.

2.1 Reasoning using Knowledge Graphs

Various methods have been proposed for link prediction in Knowledge graph. Most of them can be divided into Embedding based, graph based or a combination of both. In this thesis the focus is on Graph based methods. Graph based methods may use embedding based methods for representation of entities and relationships. But, from their assumption that missing relationship also follow the graph structure of the original Knowledge graph, they are classified as graph based methods.

Vector based methods learn vector representations for entities and predicates using tensor factorisation or neural based methods. But such methods fail to find more complex relationships in the knowledge graph that require multiple hops on knowledge graph to reason.

In Path Ranking algorithm, (Lao *et al.*, 2011) random walks of bounded length are used as features for link prediction of the missing edges. This was the first approach that attempted to predict links that with multi-hop reasoning. Let $\pi_L(i, j, k, t)$ denote a path of length L of the form $e_i \xrightarrow{r_1} e_2 \xrightarrow{r_2} e_3 \cdots \xrightarrow{r_L} e_j$, where t represents the sequence

of edge types $t = (r_1, r_2, \ldots, r_L)$. $\Pi_L(i, j, k, t)$ represents all such paths of length L from i to j. Probability of these paths are calculated. While calculating it is assumed that at each step, outgoing edge is uniformly sampled. Feature vector is computed with the probabilities of each of such paths. PRA uses path probabilities as features for predicting the probability of missing edges.

Another approach based on random walks on Knowledge graph (Wang and Cohen, 2015), combines Knowledge graph and text to find missing links in Knowledge graph. They propose to use recursive random walk to integrate knowledge from Knowledge graph and text. A potential bottleneck of random walk based methods is that the probabilities of paths vanish when they pass through entities with large number of edges. Also in such cases the number of paths increase exponentially. These methods rely on pre-computed paths and hence are often more susceptive to heuristics involved in finding these path types.

Other methods have been proposed for Knowledge graph completion using neural networks. (Toutanova *et al.*, 2015) use Convolutional neural networks for multi-hop reasoning. (Neelakantan *et al.*, 2015) use recurrent neural networks for modelling relational paths. These methods rely a lot on random walks sampled from methods like Path Ranking algorithm and fail to automatically propose and find good relational paths.

Recently many methods have been proposed that try to find good representative relational paths for knowledge base completion or knowledge graph reasoning. These methods use Reinforcement learning. The agent learns to propose good relational paths that may be representative of a relation. These methods find very good paths and also their query time is less as the number of paths proposed is often in the range of 10 for a relation.

The problem with applying reinforcement learning in Knowledge graphs is that the action space is different for each state. Standard reinforcement learning setting has a action set common to all the states. The literature on action constrained environments in reinforcement learning is less. Not many methods have been proposed on such environments.

In Deeppath (Xiong *et al.*, 2017), a agent learns to traverse in a knowledge graph environment and to propose good relational paths. To tackle the problem of state-

constrained action space, the authors propose to first do supervised learning from expert trajectories which are essentially random walks on the knowledge graph. The agent thus learns to take the actions that actually represent the edges in the knowledge graph. Thus, after imitation learning the agent will get well versed in traversing in the knowledge graph.

Then, the agent is trained to find good relational paths in the knowledge graph. For this the agent is asked to find a path from a start entity to a target entity. The state contains the embedding of the current entity and the target entity. The agent is rewarded for reaching the target entity, reaching the entity in shorter number of steps and for finding diverse paths from the start entity to the target entity.

(Das *et al.*, 2018) propose an interesting framework to model the task of finding good relational paths using reinforcement learning. Their model is trained to answer queries rather than finding a path from start entity to target entity. The agent is trained to answer query $(e_s, r, ?)$. The agent must learn to reach the entity e_t if the edge (e_s, r, e_t) is present in the knowledge graph without passing through the edge (e_s, r, e_t).

Minerva Das *et al.* (2018) proposes to tackle the problem of constrained state space by passing the possible actions along with reward and state. For this action embeddings for all the possible actions is also passed along with the state embedding. The policy is then predicted by taking dot product of the transformed state embedding with transformed action embeddings. Minerva is trained using REINFORCE.

ReinforceWalk (Shen *et al.*, 2018), uses the approach similar to AlphaGo Zero (Silver *et al.*, 2017). The main problem that the above methods faced is the rewards are very sparse and are only obtained at the end of the traversal. They propose to use model-free, value based method Q-learning along with model-based Monte Carlo Tree Search. The values of the states in the traversed path are approximated by MCTS and network is updated to output that estimated values. Thus the training in ReinforceWalk is much faster than as compared to a policy gradient agent.

Knowledge graph reasoning has also been used for question answering domain. (Yuyu Zhang and Song, 2017) propose a variational method for question answering using knowledge graph. For multi-hop reasoning using knowledge graph for question answering two problems are encountered. First, one must select an entity in the knowl-

edge graph to start the traversal. And then one must traverse in the knowledge graph such that the query is answered. To optimise for both together they propose to use variatioanl methods alaong with REINFORCE.

2.2 Scene understanding

Object detection is a problem of detection and classification of many objects present in the image. In contrast to classical machine learning problems, here the output can be of variable length. As the object may be grounded anywhere in the image the search space becomes continuous. This is handled by using various heuristics. The number of possible objects can be reduced by using a sliding window protocol. This assumes that a object will not be smaller than a particular size.

The first paper to use Deep learning and get amazing results on object detection was **R-CNN** Girshick *et al.* (2014). It extracts possible objects using a region proposal method called selective search. It uses similarity between pixels to segment them into a region proposal. Features of the region are extracted by passing to through Convolutional neural network and a classification technique is used on the top to perform classification. It achieved great results but it was very slow and power consuming.

Fast-RCNN Girshick (2015) was a improvement to make it faster and better. Instead of passing each proposal through CNN individually it suggested to pass the whole image once and then use Region of interest pooling to get region features. This was faster and also with fully connected layers it becomes end-to-end differentiable. **Faster-RCNN** Ren *et al.* (2015) added a region proposal network in order to get rid of the selective search algorithm. This made the network trainable end-to-end. Region proposal network uses variable size anchors. They are slided through the image giving many possible bounding boxes. Additionally it uses foreground and background prediction to make object proposals better.

In most of the relationship detection literature, Faster-RCNN is used to get objects from the image.

One of the more popular ways of representing a image is text description. But, scene graphs can contain more information in a more organised manner. Also, they

have object grounded in the image so there is no referential uncertainty. Thus, scene graphs are better as compared to text descriptions. Scene graphs have been used for various visual tasks such as image retrieval Johnson *et al.* (2015). It can also potentially be used for visual question answering.

Previously **VRD** Sadeghi and Farhadi (2011), there has been attempt to solve relationship detection problem by considering visual phrases or each relationship as a different class. But these methods do not scale well as we saw in Section 1. Lu *et al.* Lu *et al.* (2016) first ran visual relationship detection on large scale. They considered over dataset containing over 37k relationships on 5000 images. It had 100 object categories and 70 relationship categories. It learns 2 convolutional neural networks. First it learns to classify objects. Secondly, another convolutional neural network is trained on union of bounding boxes to classify predicates. It then combines the predictions of the outputs to return the final set of relationships.

$$V(R_{i,k,j}|O_1, O_2) = P_i(O_1)(z_k^T CNN(O1, O2) + s_k)P_j(O_2)$$

Additionally, it learnt priors from pretrained word vectors. They also learn to incorporate world knowledge using word vectors. A relationship projection function is learnt to get relationship embedding from word vectors.

In **DR-Net** Dai *et al.* (2017), first, object detection is performed using Faster-RCNN. Pairs are filtered using their position in the image. For the filtered pair of objects DR-Net module takes in subject features, object features and predicate features. Features are concatenation of appearance features obtained from convolutional neural networks and spatial features of locations in the image. The paper proposes to model relationship detection as a conditional distribution over predicates given the object categories. The model could also improve the object and subject type. Conditional Random Fields have been used to model conditional distribution. x_s and x_o are the features for the subject and the object; x_r is the feature representation of the relationship

$$\begin{aligned} p(r,s,o|x_r, x_s, x_o) &= \frac{1}{2} exp(\phi(r, s, o|x_r, x_s, x_o; W) \\ \phi =& \Psi_a(s|x_s; W_a) + \Psi_a(o|x_o; W_a) + \Psi_r(r|x_r; W_r) \\ &+ \Psi_{rs}(r, s|W_{rs}) + \Psi_{ro}(r, o|W_{ro}) + \Psi_{so}(s, o|W_{so}) \end{aligned}$$

SCG (Xu *et al.*, 2017) is very similar to the previous paper. The interesting part here is instead of predicting each relationship in isolation they use unique message passing architecture (similar to Graph neural network) to perform the prediction. They exploit the unique property of scene graph being bipartite. The construct a initial graph using the visual features. The also construct the dual of the graph where edges in scene graph become nodes and nodes become edges. This leads to improved performance as the model can now capture co-occurrences of objects and relationships.

The message passing and aggregation can be described as follows:

$$m_i = \sum_{j:i\to j} \sigma(v_1^T[h_i, h_{i\to j}])h_{i\to j} + \sum_{j:j\to i} \sigma(v_2^T[h_i, h_{j\to i}])h_{j\to i}$$
$$m_{i\to j} = \sigma(w_1^T[h_i, h_{i\to j}])h_i + \sigma(w_2^T[h_j, h_{i\to j}])h_j$$

m_i is the aggregated input to the i_th node of the primal graph. $m_{i\to j}$ is the input to the node in dual graph. h_i is the hidden state of the nodes in primal graph at the previous time stamp. While, $h_{i\to j}$ is the hidden state of the nodes in dual graph at the previous time stamp. Using these the hidden state is again computed and passed to adjacent nodes in the primal or dual graph.

Px2graph (Newell and Deng, 2017) tries to solve object detection and relationship detection problem in a single deep network. They use stacked autoencoder network to get 2 heatmaps. In the first heatmap a 1 at a pixel denotes there is an object centred at that pixel. In the second heatmap a 1 a a pixel denotes there is a relation centred at that pixel. Using the final layer features at the pixels where heatmap is 1, the object class, relationship class, object bounding box height and width are predicted.

Matching objects with relationship predicates is complex here. They have 3 additional fully connected layers. The compress the final layer features in very small dimension (8 is used in paper). They define 2 additional loss functions:

$$L_{pull} = \frac{1}{\sum_{i=1}^{n} K_i} \sum_{i=1}^{n} \sum_{k=1}^{K_i} (h_i - h'_{ik})^2$$
$$L_{push} = \sum_{i=1}^{n1} \sum_{j=i+1}^{n} \max(0, m - ||h_i - h_j||)$$

The relationship embedding is converted into 2 small dimensional embeddings. First one must be similar to the subject and second one must be similar to object. Pull loss

functions is defined in such a way that compressed embedding of object and one of the embeddings of the relationship are closer. Also, if there is no relationship between two objects their embeddings are pushed apart. In the equations h_i is the compressed object embedding and h_{ij} are the compressed relationship embeddings. This is the only end-to-end trained model in the literature.

VRL (Liang *et al.*, 2017) first builds a semantic graph to encode the semantic correlations between object types, predicates and attribute types. This encodes the possible relationships that could occur. It thus reduces the search space from order of tens of thousands to tens. For this they use a unique variation structured reinforcement learning. Previous methods had to perform relationship predictions over all the object pairs, thus chances of them getting falsely detected are high. This method puts a constraint on the relationship that can occur using the semantic action graph.

Scene graph here is generated by performing BFS traversal on a subgraph of the semantic graph. The traversal outputs a scene graph. For the BFS traversal, agent at each time step must select the next edges from the possible edges of the object. The environment is set up such a way that at each time it passes the state information to the agent. The state information includes visual features of two objects, whole image features and features of previous 2 states. It also passes a set of possible attributes, possible predicates of the object, and set of possible objects to the agent. Objects are visited only once. When a object is visited, the agent predicts 3 actions. First one if for the attribute of the object. Second one is predicate between the pair of objects. Last one selects a object (or a edge) to visit next. For a current object all the relationships are found and then only it visits next objects.

This method has lot of shortcomings like it can not predict any back edges during the traversal. The feature vector passed in over 21k dimensional.

2.3 Action constrained Reinforcement Learning

Action constraints exist in most of the environments in reinforcement learning. Environments as simple as grid world also have action constraints for the states on the boundary. Various heuristics are used to address these problems. For example, in the grid world people choose to add transitions in the transition function so that the state of

the agent does not change if it takes an invalid action. Reward function is also updated to give zero or negative reward for invalid actions.

For, reinforcement learning in graphs, the number of edges per node, *i.e.*, the average degree of the nodes, is much less as compared to number of nodes in the graph. For knowledge graphs with say 1000 entities and 100 predicates, the number of possible edges in 10^5. But, average degree is generally much lesser than that. The problem becomes a lot tractable and scalable if the number of actions per state reduces from 10^5 to say 100. This motivation leads one to find alternative methods to apply reinforcement learning and learn more efficiently in graph like environments.

There have been mainly 2 type of approaches to address this. In the first one the Q-values or probabilities are predicted for all edges or group of edges and select only those which are feasible according to the knowledge graph. In VRL (Dai *et al.*, 2017), agent picks a predicate as action and it traverses along the edge that contains the selected relation. Set of feasible actions, thus, contains the set of predicate types in all the edges adjacent to the current node. Deeppath (Xiong *et al.*, 2017) first performs imitation learning from random walks performed on the knowledge graphs. This gives flexibility to also traverse to edges that may not actually exist in the knowledge graph but could potentially exist.

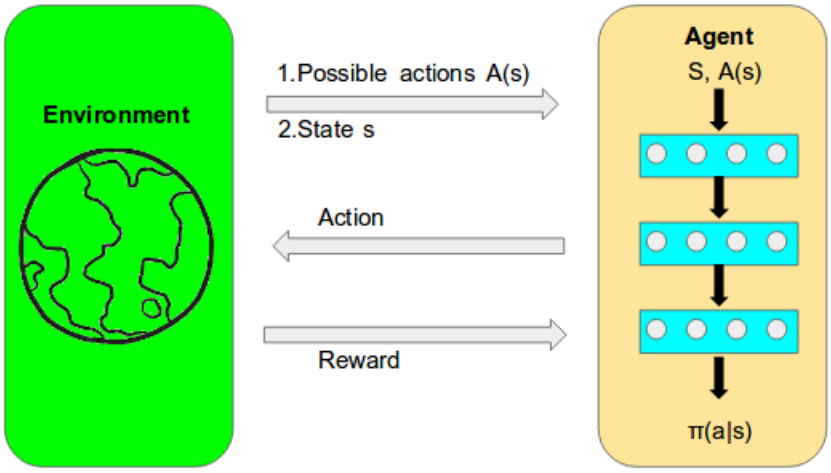


Figure 2.1: Action constrained Reinforcement Learning

Minerva (Das *et al.*, 2018) and Reinforcewalk (Shen *et al.*, 2018) select action by selecting the action for which the transformed action embeddings are most similar to the state embedding. In this approach agent is permutation invariant.

2.4 Representation Learning in graph-structured data

DeepWalk (Perozzi *et al.*, 2014) proposes to use the work on language models for vertex representation modelling in graphs. A document is considered analogous to graph, words to nodes in graph, edges to co-occurrence and sentences to a Random walks in a graph. To justify this the paper also shows that the power-law distribution of vertices appearing in short random walks follows a power-law, much like the distribution of words in natural language. It then shows how Skip-gram model can be used to get vertex representations. For each vertex as a starting node some number of random walks of fixed length are performed. For each random walk, for each node the representations are updated such that the probability co-occurrence of nodes occurring in the context is high. The paper uses Hierarchical softmax to approximate probability distribution.

Much of the above work and other methods perform the task of learning vectorial representations for nodes in a network. These methods (Perozzi *et al.*, 2014; Grover and Leskovec, 2016) learn only the vectorial representation for further tasks. They ignore any node or attribute information. They share a classifier for all the nodes with the structure information encoded in the vector representations. During classification the structure information is not taken into consideration directly.

Recently, much work has been done in graph neural networks. These methods have been proposed for tasks where we have feature vectors for the examples (as in all the machine learning algorithms) and additional structure information between these examples. The structure information provides additional information as to which examples are similar. Graph neural networks (Scarselli *et al.*, 2009), graph convolutional networks (Defferrard *et al.*, 2016) and other related methods address this by learning a intermediate representation that encodes both structure and feature vector information. The input here is a feature matrix of dimension $N \times D$ and a graph $\mathcal{G}(V, E)$ and outputs a feature matrix of dimension $N \times F$. Each neural network layer can be written as function of previous layers hidden representation and structure information $\mathcal{G}$. Most of the previous work extends this by concatenating the structure vector representation with the feature vector representation to perform further tasks like classification.

In Graph Convolutional networks (GCNs) (Defferrard *et al.*, 2016), the function at each layer is convolution operation over the feature vectors of neighbours of the input

node. Here the function between at each layer can be written as:

$$f(h_l, A) = \sigma(D^{-1/2}AD^{-1/2}h_l W_l),$$

where A is adjacency matrix of $\mathcal{G}$ and D is diagonal node matrix of A.

Inherently GCNs are very slow with the amount of computation involved. Various methods have been proposed to speed up GCNs. GraphSAGE (Hamilton *et al.*, 2017) speedups by introducing sampling of neighbours. By sampling neighbours the amount of computation is restricted.

Also, recently methods have been introduced to improve GCNs. GAT (Velickovic *et al.*, 2018) introduces attention mechanism to assign importance to each of the neighbours using attention mechanism. They use the simple self-attention, *i.e.*, neighbours with features similar to the current node's features are given relatively higher weight.

These methods have also been extended to knowledge graphs (Lin *et al.*, 2015).

CHAPTER 3

Incorporating Graph Structure for reasoning in Knowledge Graphs

In this chapter we propose to improve Deeppath by incorporating graph structure information in the network.

3.1 Motivation

Deeppath performs traversal in the knowledge graph for reasoning in Knowledge graphs. If the network has additional information of the edges that are adjacent to the current node it will help the agent in selecting an edge among the edges that are adjacent. So, we propose to pass an attended structure embedding that contains weighted sum of the embeddings of the edges. The weights are found using attention from the target embedding.

TransE embedding of an Knowledge graph are optimised such that, $\mathbf{e} + \mathbf{r} = \mathbf{t}$. So, by giving more weight is given to the edge which has its embedding $\mathbf{r}$ more similar to $\mathbf{t} - \mathbf{e}$ we are giving the network additional information of the best possible action to take. So the embedding becomes:

$$\frac{\sum_{p \in E(e)}((\mathbf{t} - \mathbf{e}) \cdot \mathbf{p})\mathbf{p}}{\sum_{p \in E(e)}((\mathbf{t} - \mathbf{e}) \cdot \mathbf{p})}$$

3.2 Task

The task here is multi-hop reasoning in knowledge graph. For a query relation r_q we must find good representative paths and use these paths to predict presence of edge (e_1, r_q, e_2) for the entity pair e_1 and e_2.

While training, the model must utilise the pairs e_1 and e_2 present in the knowledge

graph for which the edge (e_1, r_q, e_2) is also present in the knowledge graph to find representative paths present between them.

At test time, the model must be able to predict presence or absence of missing edges with predicate type e_q.

3.3 Background

Here, we describe Deeppath for multi-hop reasoning in knowledge graph in more detail. The main task is to suggest good paths between two pair of nodes in the knowledge graph. This is formulated as a sequential decision making problem and is solved using reinforcement learning. The agent learns to pick paths that have high probability of being representative and often with semantic interpretation of the relation.

Actions:

To traverse from a start entity to target entity, the agent must select an edge at each step to go to the next entity. So they define the action space as the total types of predicate relations. Here, it is assumed that all the relations in the knowledge graph are one-to-one or many-to-one. In case of one-to-many relations the action space no longer has one-to-one mapping to an edge. In Deeppath, different models are trained for different predicate types.

States:

As the number of entities in a knowledge graph can be very high they propose to use TransE embeddings. The state vector at entity $\mathbf{s}$ and with target entity $\mathbf{t}$ is given by:

$$(\mathbf{e}_s, \mathbf{e}_t - \mathbf{e}_s)$$

where $\mathbf{e}_p$ denotes the TransE embedding of entity p.

Rewards:

The total reward is weighted summation of the below awards.

1. *Global accuracy:* This reward is $+1$ if the agent successfully reaches the target entity. If the agent fails to reach a target entity the reward is -1.

2. *Path efficiency:* Shorter paths are preferred as they provide more reliable reasoning as per observation. So additional reward of inverse of length of path is awarded if the agent reaches the final state.

3. *Path diversity:* To encourage more diverse paths agent receives additional reward if the path found is different

Supervised learning as pretraining:

As the action space is large the agent with many relationship types, learning is very slow. To address this problem the authors propose to first do imitation learning. For this random walks are performed on the Knowledge graph. These random walks are used as expert trajectories to learn the structure of knowledge graph and about the feasible actions. The network is then updated using Monte-carlo policy gradient.

$$\begin{aligned} J(\theta) &= E_{a\sim\pi(a|s;\theta)}(\sum_t R_{s_t,a_t}) \\ &= \sum_t \sum_{a\in\mathcal{A}} \pi(a|s_t;\theta) R_{s_t,a_t} \end{aligned}$$

For a path p found using random walk from entity 1 to entity 2 with intermediate relations $\{r_t\}$, the network is updated with the below gradient.

$$\begin{aligned} \nabla_\theta J(\theta) &= \sum_t \sum_{a\in\mathcal{A}} \pi(a|s_t;\theta) \nabla_\theta \log \pi(a|s_t;\theta) \\ &\approx \nabla_\theta \sum_t \log \pi(a = r_t|s_t;\theta) \end{aligned}$$

Retraining with rewards:

For the query $(e_{source}, r_q, e_{target})$, the agent picks relations and traverses along the edge. If the agent reaches the target state the network the network is updated with the following loss function.

$$\nabla_(\theta) = \nabla_\theta \sum_t \log \pi(a = r_t|s_t;\theta) R_{total}$$

3.4 Method

We propose to improve Deeppath by also encoding the structure information to produce better performance. We propose to attended Graph Convolutional networks to encode the feature as well as structure information.

GCNs (Kipf and Welling, 2016) are ideal for learning representations which have encoded both attribute information and structure information. We propose to prepend a representation learning layer of GCNs before calculating state embedding.

As proposed by GraphSAGE Hamilton *et al.* (2017) we use sampling to limit the computation. For each node we consider maximum of 25 edges adjacent to the node at any step. We perform So the set of edges to be considered is sampled from set of all the edges. $\mathcal{N}(s)$ is a set of neighbours for entity node s. It contains tuples of predicate and entity.

$$\mathcal{N}(s) = \{(r, y)|(s, r, y) \in \mathcal{E}\}$$

Let n_s be the sampled set of neighbours adjacent at entity node s from its set of neighbours $\mathcal{N}(s)$.

We propose to use attention to assign higher weightage to more relevant edges. Using attention edges with predicate embeddings more similar to $\mathbf{e}_t - \mathbf{e}_s$ are assigned more weights.

$$a_i = (W^T(\mathbf{e}_t - \mathbf{e}_s)) \cdot \mathbf{r}_i$$

where r_i is the predicate embedding of the predicate in the i-th edge.

The state vector at current entity node s and target entity node t can now be written as:

$$(\mathbf{e}_s, \mathbf{e}_t - \mathbf{e}_s, \sum_{i \in n_s} a_p \mathbf{r}_i)$$

where r_i is the predicate embedding of the predicate in the i-th edge. The rest of the pipline and training procedure is same as in Deeppath.

3.5 Preliminary experiments

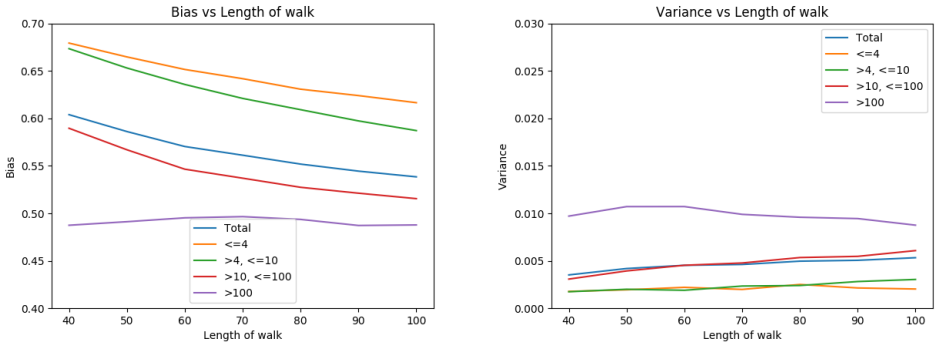
These experiments were performed alongside literature survey to get familiar with the problem statement, the tasks, difficulties involved and state-of-the-art methods.

Bias-variance decomposition analysis of DeepWalk:

To look at the bias-variance decomposition of the error and how they vary with the change in the parameters, we deviate from measure as accuracy to mean squared error. Experiments were performed on BlogCatelog3 network, with node Label classification task. The task was multi-label, multi-class classification, i.e. each node may belong to multiple classes.

Bias calculation Bias was calculated by taking mean squared error of the mean of the output probability distribution from Logistic Regression on the embeddings of these 10 models to that of the expected output probability distribution. This task was multi-class multi-label so for mean squared error the probability mass was equally distributed among the positive classes. Bias thus calculated is an approximate of the actual bias as we can not deduct the underling function.

Variance calculation Variance was calculated by taking Mean squared error of the mean output probability distribution with each of the 10 output probability distributions. As the variance of due to negative classes was high, both bias and variance were calculated on the positive classes only.



(a) Bias vs length of random walk (b) Variance vs length of random walk

Figure 3.1: BlogCatelog - Bias and variance trends for different category of nodes

Task For node label classification, embeddings of 90% of the nodes was taken as training data and rest 10% was taken as test data. Logistic regression was used for

classification. It was found that the model had high bias and low variance. It was also observed that nodes with smaller degree have more bias and less variance than nodes with higher degree. One possible reason for this could be due to their less number of occurrences in random walks.

3.6 Experiments

We use Deeppath as baseline. We evaluate our model on the tasks proposed by Deeppath, *i.e.*, link prediction (predicting target entities) and fact prediction (predicting whether an unknown fact holds or not). We evaluate our model on 10 different relation types on the NELL Knowledge graph.

Dataset:

Our experiments are conducted on NELL knowledge graph subset that is used in Deeppath (Xiong *et al.*, 2017). It is a preprocessed subset of 995th iteration of NELL system. The triples with relation *generalizations* or *haswikipediaurl* were removed as they do not have any semantic representation but they occur more than 2M times in the dataset. A subgraph was extracted form the resultant knowledge graph by keeping only the triples with predicate type being one of the Top-200 most frequently occurring predicate types. Finally, inverse relations were introduced in the knowledge graph and for each tripe (s, v, o), triple (o, v^{-1}, s) was added to allow backward traversals also.

Link prediction:

Here, the task is to rank the target entities given the source entity. The evaluation metric used to evaluate the performance is MAP (Mean Average Precision) score.

Fact Prediction:

In this task, the model is evaluated on good it can rank all the positive and negative entity pair samples for the relation. The evaluation metric here also is MAP.

3.7 Results

The results on the fact prediction on NELL dataset for our method and Deeppath are shown in Table 3.1

Table 3.1: Results for fact prediction

Task	DeepPath	Ours
agentBelongsToOrganization	0.367	0.278
athleteHomeStadium	0.729	0.717
athletePlaysInLeague	0.527	0.519
athletePlaysSport	0.55	0.35
organizationHeadquarteredInCity	0.592	0.642
organizationHiredPerson	0.554	0.519
personBornInLocation	0.289	0.487
personLeadsOrganization	0.532	0.4838
teamPlaysSport	0.398	0.308
worksFor	0.457	0.387
Average	**0.499**	**0.469**

The results on the link prediction on NELL dataset for our method and Deeppath are shown in Table 3.2

Table 3.2: Results for link prediction

Task	DeepPath	Ours
agentBelongsToOrganization	0.574	0.562
athleteHomeStadium	0.89	0.829
athletePlaysInLeague	0.961	0.8272
athletePlaysSport	0.956	0.863
organizationHeadquarteredInCity	0.789	0.789
organizationHiredPerson	0.742	0.7433
personBornInLocation	0.757	0.6989
personLeadsOrganization	0.796	0.7949
teamPlaysSport	0.741	0.663
worksFor	0.711	0.696
Average	**0.791**	**0.746**

CHAPTER 4

Visual Relationship Detection

4.1 Motivation

Visual relationship detection is a difficult problem. The state-of-the-art has 16.09% of Recall@50. There are two natural approaches Dai *et al.* (2017) to tackle this problem. Earlier Sadeghi and Farhadi (2011) this was considered as a classification task with combination of object and relationship predicates as classes. With even 1000 object categories and 200 predicate classes this will scale to 2×10^8 classes. Recent approaches Dai *et al.* (2017) Xu *et al.* (2017) Newell and Deng (2017), consider it as classification task of predicates given the object categories. With this relationships of very different object types but same predicates are considered in same class. But, this increases intra-class diversity. For example, 'man near horse' and 'tree near road' are very different but considered in same class. The poor results of the these approaches show that for even deep learning type complex approaches handling such high intra-class diversity remains difficult.

Above approaches ignore semantic inter-dependencies between objects, relationships and attributes. Thus they can only consider a small set of predicates. Liang *et al.* Liang *et al.* (2017) propose to put semantic constraints on the classes by using a semantic action graph. But, they do not consider the structure information from the graph. There are may relationships and objects that tent to co-occur. A simple example would be man, shirt and relationship 'man-wearing-shirt'. To exploit such co-occurrences we propose to use recent development in network representation learning techniques to get the structure information. Due to such large search space for classes, it is possible even complex deep learning models may not be able to completely learn the structure information. So we propose to incorporate this structure information with additional data from a Knowledge graph.

4.2 Background

In this section we describe DQN(Deep Q-Network) Mnih *et al.* (2015).

Deep Learning is very good at learning good vectorial representations of very high dimensional data. So ideally with the good representations learnt by a deep neural network a reinforcement learning agent must be able to learn good policies. But it was seen that it performed worse than a linear model. This is because deep neural network overfit and training is unstable. Mnih *et al.* (2015) introduce techniques to train deep reinforcement learning agents. The techniques are:

- **Experience Replay:** Replay buffer is used to store experiences containing state, action, reward and next state. Mini-batch is sampled from this buffer and updates are done for this mini-batch. This helps in avoiding overfitting in the deep neural network. Also, past transitions avoid catastrophic forgetting.
- **Target network:** In calculation of Q-learning error the target functions unstable as it is also getting updated. Instead, target network is fixed and is updated with new weights after some transactions.
- **Clipping rewards:** Rewards greater than 1 may lead to exploding Q-values. So rewards are if greater than 1 are clipped to 1.

4.3 Task

To get scene graph from a image, we can assume that region proposals are given. We can also use a object detection system to get initial set of objects. For each object we need to infer the class label and bounding box center coordinates, height and width. Alternatively, we can also use a end-to-end network that also does object detection for us. The task now remains to detect relationships and assign classes to capture the interactions between the objects. We must also detect for each object in the proposal, a set of attributes. For this we have knowledge from a semantic graph about the semantic constraints on the object and predicate given the subject class.

$$\mathbf{x_{objs}} = \{x_i^{cls}, x_i^{bbox} | i = 1 \dots n$$

$$\mathbf{x_{rels}} = \{x_{i,j} | i = 1 \dots n, j = 1 \dots n, i \neq j\}$$

,where n is the number of proposal boxes, $x_i^{cls} \in C$ is the class label of the i-th proposal box, $x_i^{bbox} \in \mathrm{R}^4$ is the bounding box offsets relative to the i-th proposal box coordinates, and $x_{i,j} \in R$ is the relationship predicate between the i-th and the j-th proposal boxes. C is the set of possible object classes and R is the set of possible predicate classes. $\mathbf{x_{objs}}$ is a set of objects in the scene graph. $\mathbf{x_{rels}}$ is a set of relationships in the scene graph defined on the objects.

Let B_I be the set of object proposals obtained from region proposal network. We are also given a semantic graph S. So, we would like to get the following probability distribution:

$$p(\mathbf{x_{objs}}, \mathbf{x_{rels}} | I, S)$$

This can be broken into:

$$p(\mathbf{x_{objs}}, \mathbf{x_{rels}} | I, S) = p(B_I | I) p(\mathbf{x_{objs}} | I, B_I) p(\mathbf{x_{rels}} | I, \mathbf{x_{objs}}, S)$$

4.4 Method

We propose to use knowledge graph for relationship detection. The complete approach is described in this section.

We assume that we have a knowledge graph that provides us the world knowledge of to which objects a object can be related to and the possible set of relations that exist between them. Suppose, we are working on with 1000 object types and 500 possible predicate types between them. Thus each object can be related to 1000×500 relationships at the same time. The availability of a knowledge graph restricts the number of relationships it can have to a very tractable number. Thus, a knowledge graph can act as an oracle guiding the model to make more sensible predictions and making the classification performance better.

First, we use object detector to get set $\mathcal{S}$ of candidate object instances. We use pretrained ResNet50 model to get visual features from an image.

We model finding relationships in an image as a sequential decision process. At each step of this decision process the model must detect all relationships associated with an object in the image. In order to detect multiple relationships for all objects we perform beam search traversal on the knowledge graph. We start with the object that has maximum possible relationships among the objects present in the graph. In training time the model makes traversals with beam width 1. Beam search is used at test time to make multiple predictions. This is because with this the training time is less and by learning to make better traversals beam search performance should also improve.

We use DQN (Mnih *et al.*, 2015) to learn to do traversals in the knowledge graph.

4.4.1 Knowledge graph construction

We construct the dataset itself. The dataset contains images, ground truth objects and ground truth relationships that exist between them. We take union of all the relationships of all the images. This gives us the set of feasible relationships that can exist semantically. The set of relationships can be much more than the knowledge graph constructed. But, as most of the knowledge graphs are not complete, this seems a good starting knowledge graph. We can add edges to this knowledge graph using various methods for link prediction. For the following experiments we make closed-domain assumption for the edges on the knowledge graph.

The constructed knowledge graph contains all the object types as entities and all the predicate types as the edge types. All the scene graphs (containing all the images) for images are sub-graph of this knowledge graph.

4.4.2 Knowledge graph embeddings

We use ComplEx Knowledge graph embeddings (Trouillon *et al.*, 2016, 2017) from our constructed Knowledge graph for relationship detection. [1]

ComplEx embedding learns 2 set of embeddings for both entities and relations to address asymmetric relations. For this reason they propose to learn complex embeddings for entities and relations. For each entity or relationship, its embedding can be

[1]The implementation at https://github.com/ttrouill/complex was used.

written as $E(e) = Re(e) + Im(e)i$. The function used to predict absence or presence of a edge in the knowledge graph is as follows:

$$\begin{aligned} f(E(h), E(r), E(t)) = Real(\sum_{j=1}^{k} (Re_j(h) + Im_j(h)i) \\ * (Re_j(r) + Im_j(r)i) * (Re_j(t)Im_j(t)i) \\ = Re(< E(h), E(r), \bar{E}(t) >) \end{aligned}$$

where $\bar{E}(t)$ is complex conjugate of the embedding.

Real part of triple product is used to predict the presence of edge in the knowledge graph.

By changing the sign of the tail entity ComplEx models asymmetric relations.

The choice of ComplEx embedding was based on the following reasons:

- (Trouillon *et al.*, 2017) prove that ComplEx is fully-expressive. In practice, they prove that any ground truth edgeset can be modelled in ComplEx with embeddings of length at most $|E| \cdot |R|$. Wang and Li (2018) prove that translational approach for embeddings are not fully expressive. ComplEx are also fully-expressive even if there are non-symmetric relationships present in the knowledge graph.
- ComplEx have linear time complexities and the number of their parameters grow linearly. Therefore, they scale better and are less prone to over-fitting.
- ComplEx embeddings are interpretable. Each scalar in the embedding vector of the entity can be considered as a feature. And each number in the embedding of relation represents the weightage relation assigns to the particular feature.

4.4.3 State space

Let s be the current object in the traversal on knowledge graph. The state vector is concatenation of

- 2096 dimensional Visual features from ResNet50 network.

- Concatenation of real part and imaginary part of the entity embedding.

$$Re(s) \oplus Im(s)$$

  where $Re(s)$ and $Im(s)$ are the complex knowledge graph embedding learnt for node s using ComplEx.

This representation contains visual information of the object and the global semantic knowledge contained learnt using knowledge graph.

4.4.4 State constrained action space

Each edge in the knowledge graph from node s is a valid action from state s.

$$A(s) = \{(h, t)|(h, r, t) \in \mathcal{E}\}$$

We use vectorial representation of action instead of symbolic representation as we have more than. Action embedding for action (r, t) is concatenation of

- 2096 dimensional Visual features of object o
- Complex embeddings of predicate: $Re(r) \oplus Im(r)$
- Complex embedding of tail entity $Re(t) \oplus Im(t)$
- Engineered features $Re(r) \cdot Re(t)$
- Engineered features $Re(r) \cdot Im(t)$
- Engineered features $Im(r) \cdot Re(t)$
- Engineered features $Im(r) \cdot Im(t)$

The above features were added as the function that approximates presence or absence of edge in Knowledge graph.

4.4.5 Reward

Reward is +1 if the relationship exists in ground truth. Instead of given 0 reward for non-existing relationships we give reward of 0.01 as the relationship may exist but has not been included.

4.4.6 Value network

From set of possible actions we concatenate the action embeddings to get action matrix. Let set of actions be $\mathcal{A} = \{a_1, a_2, \ldots a_n\}$. Then action matrix can be written as:

$$\mathbf{A} = \begin{bmatrix} F_a(a_1) \\ F_a(a_2) \\ \vdots \\ F_a(a_n) \end{bmatrix}$$

where, $F_a(a)$ are features of action a

The Q-value function for an action in a particular state is estimated as follows:

$$Q(s, a) = f_{obj}(F_o(s)) \cdot f_{act}(F_a(a))$$

where,

- f_{obj} and f_{act} are fully connected neural networks
- $F_o(\cdot)$ and $F_a(\cdot)$ represent the feature vectors for object and action respectively

4.4.7 Training

The network is trained using the Q-learning update. The tuples of type $(\mathbf{v}, \mathbf{v}', \mathbf{a}, R, \mathbf{A}')$ are stored in Relpay buffer, where,

- $\mathbf{v}$ is the feature vector of current state,
- $\mathbf{v}'$ is the feature vector of next state,

- $\mathbf{a}$ is the action embedding,
- R is the reward,
- $\mathbf{A}'$ is a matrix containing embeddings of all the possible actions in the next state.

For a minibatch sampled from replay buffer containing tuples of type $(\mathbf{v}, \mathbf{v}', \mathbf{a}, R, \mathbf{A}')$, the update equation is given by

$$\Delta\theta = \alpha(\nabla_\theta(f_{obj}(v) \cdot f_{act}(a)))(R + \gamma \max f'_{obj}(v') \cdot f'_{act}(A') - f_{obj}(v) \cdot f_{act}(a))$$

where, f'_{act} and f'_{obj} are the corresponding target networks, γ is discount factor, α is learning rate

Exploration while training:

As the aim is to learn to predict as many relationships and not take a action with maximum possible reward this becomes a problem of Multiple Choice Learning Guzman-Rivera *et al.* (2012). So with 0.9 probability we select from the Top B actions with probability of each action being taken being proportional to its predicted Q-value.

Supervised learning:

The learning is very slow initially so we perform imitation-learning so that the agent learns to follow the expert trajectories. We do 1 supervised learning update for each 8 Q-learning updates. For this, expert trajectories are obtained by performing random walk on the ground truth scene graphs.

4.4.8 Inference

Beam search is used at inference. Beam search with width d is performed to get many good relationships for a object.

4.5 Preliminary experiments

These experiments were performed alongside literature survey to get familiar with the problem statement, the tasks, difficulties involved and state-of-the-art methods.

Implementation of Px2graph paper (Newell and Deng, 2017) With the aim of penalising the network if it makes semantically unlikely relationship predictions. This can be done by adding additional loss function to bring object embedding produced close to the Knowledge graph embedding. But with potential better approach this was then scraped.

Table 4.1: Convergence of Losses

	Heatmap	Obj class.	Obj reg.	Rel class.	Pull	Push
Initial	2482.52	11394	4.3e+07	6823.8	12596	21382
5 epochs	1907.47	791.51	2247.9	130.5	7958.1	1.8

4.6 Experiments

4.6.1 Dataset

We perform experiments on Visual Genome dataset (Krishna *et al.*, 2017).The dataset contains over 107k images, ground truth objects, relationships between objects and object attributes. 95k images are used for training, 5k for validation and 5k for test. We train on subset of the annotated data. We consider all the object and predicate types that occur at least 200 times in the dataset. This subset has 1293 object types, 303 predicate types, 278,984 unique relationship types. On an average each image contains 30.7 annotated objects and 23.2 annotated images.

4.6.2 Implementation details

We train a deep Q-network for 1 epoch with RMSProp optimizer. We use a mini-batch size of 8 images. Discount factor is set to 0.9. Learning rate is 2e-5.

4.6.3 Evaluation

Recall@k is used as evaluation metric. It calculates the number of ground truth predictions present in Top k confident predictions.

4.7 Results

4.7.1 Comparison with baseline models

We compare with VRL (Liang *et al.*, 2017). All other methods consider only a small set object types and relationship types.

Due to large training time of baseline we could not train baseline on our dataset. We achieve performance similar to VRL. As the data is different the the two models can not be compared but our model gives decent performance.

Table 4.2: Results

Model	# Obj types	# Pred types	# Rel types	# Avg objs in image	#Avg rels in image	metric	Score
Ours	1293	303	278984	30.7	23.2	Recall@200	12.93 %
VRL	1750	347	13,894	-	8.05	Recall@50	14.36%

A few of the predicted relationships for 2 example images.

Figure 4.1: An example

Table 4.3: Predicted relationships

Head	Relation	Tail
man	has	Jacket
Man	wearing	Jacket
Man	behind	pole
man	behind	pole

Figure 4.2: An example

Table 4.4: Predicted relationships

Head	Relation	Tail
woman	has	neck
man	wearing	Jacket
Man	wearning	shirt
woman	has	hair

CHAPTER 5

Conclusion

We study sequential decision making in graph-structured environments using action-constrained reinforcement learning. Our first task is multi-hop reasoning in Knowledge graphs. We propose to improve the current state-of-the-art method by adding graph structure information using Graph Convolution networks. But, we do not see improvement in the performance. Another task that we look at is relationship detection by performing traversal in the knowledge graph. We propose a novel framework for this task. We run on dataset containing over 20 times number of classes as compared to baseline with almost similar performance on evaluation metric.

REFERENCES

1. **Abadi, M.**, **A. Agarwal**, **P. Barham**, **E. Brevdo**, **Z. Chen**, **C. Citro**, **G. S. Corrado**, **A. Davis**, **J. Dean**, **M. Devin**, **S. Ghemawat**, **I. Goodfellow**, **A. Harp**, **G. Irving**, **M. Isard**, **Y. Jia**, **R. Jozefowicz**, **L. Kaiser**, **M. Kudlur**, **J. Levenberg**, **D. Mané**, **R. Monga**, **S. Moore**, **D. Murray**, **C. Olah**, **M. Schuster**, **J. Shlens**, **B. Steiner**, **I. Sutskever**, **K. Talwar**, **P. Tucker**, **V. Vanhoucke**, **V. Vasudevan**, **F. Viégas**, **O. Vinyals**, **P. Warden**, **M. Wattenberg**, **M. Wicke**, **Y. Yu**, and **X. Zheng** (2015). TensorFlow: Large-scale machine learning on heterogeneous systems. URL https://www.tensorflow.org/. Software available from tensorflow.org.

2. **Bordes, A.**, **N. Usunier**, **A. Garcia-Duran**, **J. Weston**, and **O. Yakhnenko**, Translating embeddings for modeling multi-relational data. *In Advances in neural information processing systems*. 2013.

3. **Chang, A. X.**, **M. Savva**, and **C. D. Manning**, Learning spatial knowledge for text to 3d scene generation. *In Empirical Methods in Natural Language Processing (EMNLP)*. 2014.

4. **Chang, S.**, **W. Han**, **J. Tang**, **G.-J. Qi**, **C. C. Aggarwal**, and **T. S. Huang**, Heterogeneous network embedding via deep architectures. *In Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. ACM, 2015.

5. **Dai, B.**, **Y. Zhang**, and **D. Lin**, Detecting visual relationships with deep relational networks. *In The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2017.

6. **Das, R.**, **S. Dhuliawala**, **M. Zaheer**, **L. Vilnis**, **I. Durugkar**, **A. Krishnamurthy**, **A. Smola**, and **A. McCallum**, Go for a walk and arrive at the answer: Reasoning over paths in knowledge bases using reinforcement learning. *In Proceedings of Sixth International Conference on Learning Representations*. 2018.

7. **Defferrard, M.**, **X. Bresson**, and **P. Vandergheynst**, Convolutional neural networks on graphs with fast localized spectral filtering. *In Advances in Neural Information Processing Systems*. 2016.

8. **Dong, Y.**, **N. V. Chawla**, and **A. Swami**, Metapath2vec: Scalable representation learning for heterogeneous networks. *In Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. 2017.

9. **Gardner, M.** (2015). *Reading and Reasoning with Knowledge Graphs*. Ph.D. thesis, Language Technologies Institute, Carnegie Mellon University.

10. **Girshick, R.**, Fast r-cnn. *In Proceedings of the IEEE international conference on computer vision*. 2015.

11. **Girshick, R.**, **J. Donahue**, **T. Darrell**, and **J. Malik**, Rich feature hierarchies for accurate object detection and semantic segmentation. *In Proceedings of the IEEE conference on computer vision and pattern recognition*. 2014.

12. **Grover, A.** and **J. Leskovec**, node2vec: Scalable feature learning for networks. *In Proceedings of the 22nd ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM, 2016.

13. **Guu, K.**, **J. Miller**, and **P. Liang**, Traversing knowledge graphs in vector space. *In Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*. 2015.

14. **Guzman-Rivera, A.**, **D. Batra**, and **P. Kohli**, Multiple choice learning: Learning to produce multiple structured outputs. *In Advances in Neural Information Processing Systems*. 2012.

15. **Hamilton, W.**, **Z. Ying**, and **J. Leskovec**, Inductive representation learning on large graphs. *In Advances in Neural Information Processing Systems*. 2017.

16. **Johnson, J.**, **R. Krishna**, **M. Stark**, **L.-J. Li**, **D. Shamma**, **M. Bernstein**, and **L. Fei-Fei**, Image retrieval using scene graphs. *In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2015.

17. **Karpathy, A.** and **L. Fei-Fei**, Deep visual-semantic alignments for generating image descriptions. *In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2015.

18. **Kipf, T. N.** and **M. Welling** (2016). Semi-supervised classification with graph convolutional networks. *arXiv preprint arXiv:1609.02907*.

19. **Krishna, R.**, **Y. Zhu**, **O. Groth**, **J. Johnson**, **K. Hata**, **J. Kravitz**, **S. Chen**, **Y. Kalantidis**, **L.-J. Li**, **D. A. Shamma**, *et al.* (2017). Visual genome: Connecting language and vision using crowdsourced dense image annotations. *International Journal of Computer Vision*, **123**(1), 32–73.

20. **Lao, N.**, **T. Mitchell**, and **W. W. Cohen**, Random walk inference and learning in a large scale knowledge base. *In Proceedings of the Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, 2011.

21. **Liang, X.**, **L. Lee**, and **E. P. Xing**, Deep variation-structured reinforcement learning for visual relationship and attribute detection. *In The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2017.

22. **Lin, Y.**, **Z. Liu**, **H. Luan**, **M. Sun**, **S. Rao**, and **S. Liu**, Modeling relation paths for representation learning of knowledge bases. *In Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*. 2015.

23. **Lu, C.**, **R. Krishna**, **M. Bernstein**, and **L. Fei-Fei**, Visual relationship detection with language priors. *In European Conference on Computer Vision*. Springer, 2016.

24. **Mnih, V.**, **K. Kavukcuoglu**, **D. Silver**, **A. A. Rusu**, **J. Veness**, **M. G. Bellemare**, **A. Graves**, **M. Riedmiller**, **A. K. Fidjeland**, **G. Ostrovski**, *et al.* (2015). Human-level control through deep reinforcement learning. *Nature*, **518**(7540), 529.

25. **Neelakantan, A.**, **B. Roth**, and **A. Mc-Callum**, Compositional vector space models for knowledge base inference. *In Proceedings of 53rd Annual Meeting of the Association for Computational Linguistics*. 2015.

26. **Newell, A.** and **J. Deng**, Pixels to graphs by associative embedding. *In Advances in Neural Information Processing Systems*. 2017.

27. **Nickel, M.**, **K. Murphy**, **V. Tresp**, and **E. Gabrilovich** (2016). A review of relational machine learning for knowledge graphs. *Proceedings of the IEEE*, **104**(1), 11–33.

28. **Paszke, A.**, **S. Gross**, **S. Chintala**, **G. Chanan**, **E. Yang**, **Z. DeVito**, **Z. Lin**, **A. Desmaison**, **L. Antiga**, and **A. Lerer**, Automatic differentiation in pytorch. *In NIPS-W*. 2017.

29. **Perozzi, B.**, **R. Al-Rfou**, and **S. Skiena**, Deepwalk: Online learning of social representations. *In Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM, 2014.

30. **Ren, S.**, **K. He**, **R. Girshick**, and **J. Sun**, Faster r-cnn: Towards real-time object detection with region proposal networks. *In Advances in neural information processing systems*. 2015.

31. **Sadeghi, M. A.** and **A. Farhadi**, Recognition using visual phrases. *In Computer Vision and Pattern Recognition (CVPR), 2011 IEEE Conference on*. IEEE, 2011.

32. **Scarselli, F.**, **M. Gori**, **A. C. Tsoi**, **M. Hagenbuchner**, and **G. Monfardini** (2009). The graph neural network model. *IEEE Transactions on Neural Networks*, **20**(1), 61–80.

33. **Shen, Y.**, **J. Chen**, **P.-S. Huang**, **Y. Guo**, and **J. Gao** (2018). Reinforcewalk: Learning to walk in graph with monte carlo tree search. *arXiv preprint arXiv:1802.04394*.

34. **Silver, D.**, **J. Schrittwieser**, **K. Simonyan**, **I. Antonoglou**, **A. Huang**, **A. Guez**, **T. Hubert**, **L. Baker**, **M. Lai**, **A. Bolton**, *et al.* (2017). Mastering the game of go without human knowledge. *Nature*, **550**(7676), 354.

35. **Teney, D.**, **L. Liu**, and **A. van den Hengel**, Graph-structured representations for visual question answering. *In The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2017.

36. **Toutanova, K.**, **D. Chen**, **P. Pantel**, **H. Poon**, **P. Choudhury**, and **M. Gamon**, Representing text for joint embedding of text and knowledge bases. *In Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*. 2015.

37. **Trouillon, T.**, **C. R. Dance**, **É. Gaussier**, **J. Welbl**, **S. Riedel**, and **G. Bouchard** (2017). Knowledge graph completion via complex tensor factorization. *Journal of Machine Learning Research*, **18**(130), 1–38.

38. **Trouillon, T.**, **J. Welbl**, **S. Riedel**, **É. Gaussier**, and **G. Bouchard**, Complex embeddings for simple link prediction. *In International Conference on Machine Learning*. 2016.

39. **Velickovic, P.**, **G. Cucurull**, **A. Casanova**, **A. Romero**, **P. Lio**, and **Y. Bengio** (2018). Graph attention networks.

40. **Wang, G. R., Yanjie** and **H. Li**, On multi-relational link prediction with bilinear models. *In AAAI*. 2018.

41. **Wang, Q.**, **Z. Mao**, **B. Wang**, and **L. Guo** (2017). Knowledge graph embedding: A survey of approaches and applications. *IEEE Transactions on Knowledge and Data Engineering*, **29**(12), 2724–2743.

42. **Wang, W. Y.** and **W. W. Cohen**, Joint information extraction and reasoning: A scalable statistical relational learning approach. *In Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, volume 1. 2015.

43. **Wenhu Chen, X. Y. W. W., Wenhan Xiong**, Variational knowledge graph reasoning. *In Proceedings of the North American Chapter of the Association for Computational Linguistics (NAACL)*. 2018.

44. **Xiong, W.**, **T. Hoang**, and **W. Y. Wang**, Deeppath: A reinforcement learning method for knowledge graph reasoning. *In Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*. 2017.

45. **Xu, D.**, **Y. Zhu**, **C. B. Choy**, and **L. Fei-Fei**, Scene graph generation by iterative message passing. *In The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2017.

46. **Yuyu Zhang, Z. K. A. S., Hanjun Dai** and **L. Song**, Variational reasoning for question answering with knowledge graph. *In Proceedings of the 2015 Conference on AAAI*. 2017.

47. **Zellers, R.**, **M. Yatskar**, **S. Thomson**, and **Y. Choi**, Neural motifs: Scene graph parsing with global context. *In Conference on Computer Vision and Pattern Recognition*. 2018.

48. **Zeng, D.**, **K. Liu**, **S. Lai**, **G. Zhou**, and **J. Zhao**, Relation classification via convolutional deep neural network. *In Proceedings of COLING 2014, the 25th International Conference on Computational Linguistics*. 2014.

49. **Zhang, H.**, **Z. Kyaw**, **S.-F. Chang**, and **T.-S. Chua**, Visual translation embedding network for visual relation detection. *In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, volume 2. 2017.