Learning When Not to Answer: A Ternary Reward Structure for Reinforcement Learning based Question Answering

Fréderic Godin
Ghent University
Ghent, Belgium
frederic.godin@ugent.be

Anjishnu Kumar
Amazon Research
Cambridge, United Kingdom
anjikum@amazon.com

Arpit Mittal
Amazon Research
Cambridge, United Kingdom
mitarpit@amazon.co.uk

Abstract

In this paper, we investigate the challenges of using reinforcement learning agents for question-answering over knowledge graphs for real-world applications. We examine the performance metrics used by state-of-the-art systems and determine that they are inadequate for such settings. More specifically, they do not evaluate the systems correctly for situations when there is no answer available and thus agents optimized for these metrics are poor at modeling confidence. We introduce a simple new performance metric for evaluating question-answering agents that is more representative of practical usage conditions, and optimize for this metric by extending the binary reward structure used in prior work to a ternary reward structure which also rewards an agent for not answering a question rather than giving an incorrect answer. We show that this can drastically improve the precision of answered questions while only not answering a limited number of previously correctly answered questions. Employing a supervised learning strategy using depth-first-search paths to bootstrap the reinforcement learning algorithm further improves performance.

1 Introduction

A number of approaches for question answering have been proposed recently that use reinforcement learning to reason over a knowledge graph (Das et al., 2018; Lin et al., 2018; Chen et al., 2018; Zhang et al., 2018). In these methods the input question is first parsed into a constituent question entity and relation. The answer entity is then identified by sequentially taking a number of steps (or 'hops') over the knowledge graph (KG) starting from the question entity. The agent receives a positive reward if it arrives at the correct answer entity and a negative reward for an incorrect answer entity. For example, for the question "What is the capital of France?", the question entity is $(France)$ and the goal is to find a path in the KG which connects it to $(Paris)$. The relation between the answer entity and question entity in this example is $(Capital\ of)$ which is missing from the KG and has to be inferred via alternative paths. This is illustrated in Figure 1. A possible two-hop path to find the answer is to use the fact that $(Macron)$ is the president of $(France)$ and that he lives in $(Paris)$. However, there are many paths that lead to the entity $(Paris)$ but also to other entities which makes finding the correct answer a non-trivial task.

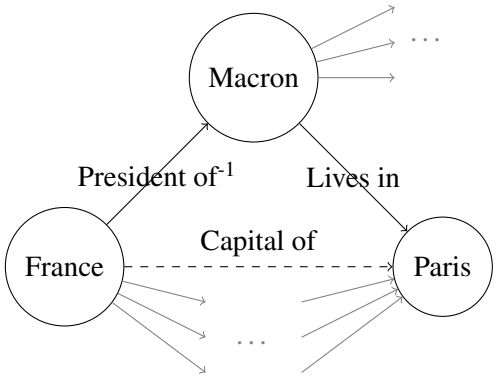


Figure 1: Fictional graph for the the question "What's the capital of France?". The relation $(Capital\ of)$ does not exist in the graph and thus an alternative path needs to be used that leads to the correct answer.

The standard evaluation metrics used for these systems are metrics developed for web search such as Mean Reciprocal Rank (MRR) and hits@k, where k ranges from 1 to 20. We argue that this is not a correct evaluation mechanism for a practical question-answering system (such as Alexa, Cortana, Siri, etc.) where the goal is to return a single answer for each question. Moreover it is assumed that there is always an answer entity that could be

Proceedings of NAACL-HLT 2019, pages 122–129
Minneapolis, Minnesota, June 2 - June 7, 2019.

reached from the question entity in limited number of steps. However this cannot be guaranteed in a large-scale commercial setting and for all KGs. For example, in our proprietary dataset used for the experimentation, for 15.60% of questions the answer entity cannot be reached within the limit of number of steps used by the agent. Hence, we propose a new evaluation criterion, allowing systems to return 'no answer' as a response when no answer is available.

We demonstrate that existing state-of-the-art methods are not suited for a practical question-answering setting and perform poorly in our evaluation setup. The root-cause of poor performance is the reward structure which does not provide any incentive to learn not to answer. The modified reward structure we present allows agents to learn not to answer in a principled way. Rather than having only two rewards, a positive and a negative reward, we introduce a ternary reward structure that also rewards agents for not answering a question. A higher reward is given to the agent for correctly answering a question compared to not answering a question. In this setup the agent learns to make a trade-off between these three possibilities to obtain the highest total reward over all questions.

Additionally, because the search space of possible paths exponentially grows with the number of hops, we also investigate using Depth-First-Search (DFS) algorithm to collect paths that lead to the correct answer. We use these paths as a supervised signal for training the neural network before the reinforcement learning algorithm is applied. We show that this improves overall performance.

2 Related work

The closest works to ours are the works by Lin et al. (2018), Zhang et al. (2018) and Das et al. (2018), which consider the question answering task in a reinforcement learning setting in which the agent always chooses to answer.[1] Other approaches consider this as a link prediction problem in which multi-hop reasoning can be used to learn relational paths that link two entities. One line of work focuses on composing embeddings (Neelakantan et al., 2015; Guu et al., 2015; Toutanova et al., 2016) initially introduced for link prediction, e.g., TransE (Bordes et al., 2013), ComplexE (Trouillon et al., 2016) or ConvE (Dettmers et al., 2018). Another line of work focuses on logical rule learning such as neural logical programming (Yang et al., 2017) and neural theorem proving (Rocktäschel and Riedel, 2017). Here, we focus on question answering rather than link prediction or rule mining and use reinforcement learning to circumvent that we do not have ground truth paths leading to the answer entity.

Recently, popular textual QA datasets have been extended with not-answerable questions (Trischler et al., 2017; Rajpurkar et al., 2018). Questions that cannot be answered are labeled with 'no answer' option which allows for supervised training. This is different from our setup in which there are no ground truth 'no answer' labels.

3 Background: Reinforcement learning

We base our work on the recent reinforcement learning approaches introduced in Das et al. (2018) and Lin et al. (2018). We denote the knowledge graph as $\mathcal{G}$, the set of entities as $\mathcal{E}$, the set of relations as $\mathcal{R}$ and the set of directed edges $\mathcal{L}$ between entities of the form $l = (e_1, r, e_2)$ with $e_1, e_2 \in \mathcal{E}$ and $r \in \mathcal{R}$. The goal is to find an answer entity e_a given a question entity e_q and the question relation r_q, when (e_q, r_q, e_a) is not part of graph $\mathcal{G}$.

We formulate this problem as a Markov Decision Problem (MDP) (Sutton and Barto, 1998) with the following states, actions, transition function and rewards:

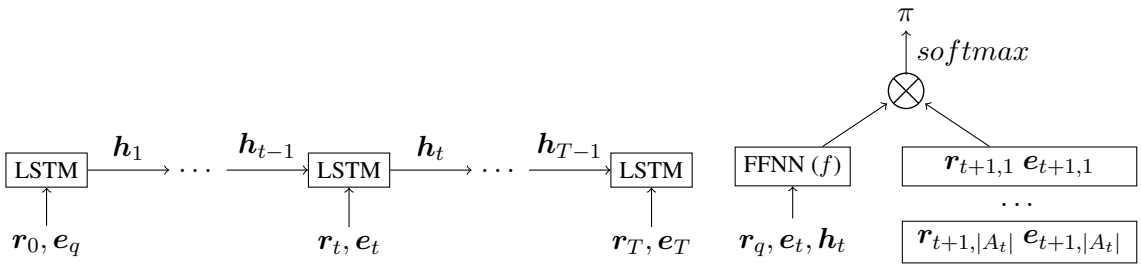
States. At every timestep t, the state s_t is defined by the current entity e_t, the question entity e_q and relation r_q, for which $e_t, e_q \in \mathcal{E}$ and $r_q \in \mathcal{R}$. More formally, $s_t = (e_t, e_q, r_q)$.

Actions. For a given entity e_t, the set of possible actions is defined by the outgoing edges from e_t. Thus $A_t = \{(r', e') | (e_t, r', e') \in \mathcal{G}\}$.

Transition function. The transition function δ maps s_t to a new state s_{t+1} based on the action taken by the agent. Consequently, $s_{t+1} = \delta(s_t, A_t) = \delta(e_t, e_q, r_q, A_t)$.

Rewards. The agent is rewarded based on the final state. For example, in Das et al. (2018) and Lin et al. (2018) the agent obtains a reward of 1 if the correct answer entity is reached as the final state and 0 otherwise (i.e., $R(s_T) = \mathbb{I}\{e_T = e_a\}$).

[1] An initial version of this paper has been presented at the Relational Representation Learning Workshop at NeurIPS 2018 as Godin et al. (2018).



(a) History LSTM

(b) Policy network at timestep t

Figure 2: Figure 2a illustrates the LSTM which encodes history of the path taken. The output at timestep t is used as input to the policy network, illustrated in Figure 2b, to determine which action to take next.

3.1 Training

We train a policy network π using the REINFORCE algorithm of Williams (1992) which maximizes the expected reward:

$$J(\boldsymbol{\theta}) = \mathbb{E}_{(e_q,r_q,e_a)\in\mathcal{G}}\mathbb{E}_{a_1,\ldots,a_T\sim\pi}[R(s_T|e_q,r_q)] \quad (1)$$

in which a_t is the action selected at timestep t following the policy π, and $\boldsymbol{\theta}$ are the parameters of the network.

The policy network consists of two parts: a Long Short-Term Memory (LSTM) network which encodes the history of the traversed path, and a feed-forward neural network to select an action (a_t) out of all possible actions. Each entity and relation have a corresponding vector $\boldsymbol{e}_t, \boldsymbol{r}_t \in \mathbb{R}^d$. The action $a_t \in A_t$ is represented by the vectors of the relation and entity as $\boldsymbol{a}_t = [\boldsymbol{r}_{t+1}; \boldsymbol{e}_{t+1}] \in \mathbb{R}^{2d}$. The LSTM encodes the history of the traversed path and updates its hidden state each timestep, based on the selected action:

$$\boldsymbol{h}_t = LSTM(\boldsymbol{h}_{t-1}, \boldsymbol{a}_{t-1}) \quad (2)$$

This is illustrated in Figure 2a.

Finally, the feed-forward neural network (f) combines the history $\boldsymbol{h}_t$, the current entity representation $\boldsymbol{e}_t$ and the query relation $\boldsymbol{r}_q$. Using softmax, we compute the probability for each action by calculating the dot product between the output of f and each action vector $\boldsymbol{a}_t$:

$$\pi(a_t|s_t) = softmax(\boldsymbol{A}_t f(\boldsymbol{h}_t, \boldsymbol{e}_t, \boldsymbol{r}_q)) \quad (3)$$

in which $\boldsymbol{A}_t \in \mathbb{R}^{|A_t|\times 2d}$ is a matrix consisting of rows of action vectors $\boldsymbol{a}_t$. This is illustrated in Figure 2b. During training, we sample over this probability distribution to select the action a_t, whereas during inference, we use beam search to select the most probable path.

4 Evaluation

User-facing question answering systems inherently face a trade-off between presenting an answer to a user that could potentially be incorrect, and choosing not to answer. However, prior work in knowledge graph question-answering (QA) only considers cases in which the answering agent always produces an answer. This setup originates from the link prediction and knowledge base completion tasks in which the evaluation criteria are hits@k and Mean Reciprocal Rank (MRR), where k ranges from 1 to 20. However, these metrics are not an accurate representation of practical question-answering systems in which the goal is to return a single correct answer or not answer at all. Moreover, using these metrics result in the problem of the model learning 'spurious' paths since the metrics encourage the models to make wild guesses even if the path is unlikely to lead to the correct answer.

We therefore propose to measure the fraction of questions the system answers (Answer Rate) and the number of correct answers out of all answers (Precision) to measure the system performance. We combine these two metrics by taking the harmonic mean and call this the QA Score. This can be viewed as a variant of the popular F-Score metric, with answer rate used as an analogue to recall in the original metric.

5 Proposed method

In this section, we will first introduce the supervised learning technique we used to pretrain the neural network before applying the reinforcement learning algorithm. Next we will describe the ternary reward structure.

5.1 Supervised learning

Typically in reinforcement learning, the search space of possible actions and paths grows exponentially with the path length. Our problem is no exception to this. Hence an imitation learning approach could be beneficial here where we provide a number of expert paths to the learning algorithm to bootstrap the learning process. This idea has been explored previously in the context of link and fact prediction in knowledge graphs where Xiong et al. (2017) proposed to use a Breadth-First-Search (BFS) between the entity pairs to select a set of plausible paths. However BFS favours identification of shorter paths which could bias the learner. We therefore use Depth-First-Search (DFS) to identify paths between question and answer entities and sample up to 100 paths to be used for the supervised training. If no path can be found between the entity pair we return a 'no answer' label. Following this, we train the network using reinforcement learning algorithm which refines it further. Note that it is not guaranteed that the set of paths found using DFS are all most efficient. However as we show in our experiments, bootstrapping with these paths provide good initialization for the reinforcement learning algorithm.

5.2 Ternary reward structure

As mentioned previously, we encounter situations when the answer entity cannot be reached in the limited number of steps taken by an agent. In such cases, the system should return a special answer 'no answer' as the response. We can achieve this by adding a synthetic 'no answer' action that leads to a special entity $e_{NOANSWER}$. This is illustrated in Figure 3. In the framework of Das et al. (2018) a binary reward is used which rewards the learner for the answer being wrong or correct. Following a similar protocol, we could award a score of 1 to return 'no answer' when there is no answer available in the KG. However, we cannot achieve reasonable training with such reward structure. This is because there is no specific pattern for 'no answer' that could be directly learned. Hence, if we reward a system equally for correct or no answer, it learns to always predict 'no answer'. We therefore propose a ternary reward structure in which a positive reward is given to a correct answer, a neutral reward when $e_{NOANSWER}$ is selected as an answer, and a negative reward for an incorrect answer. More formally:

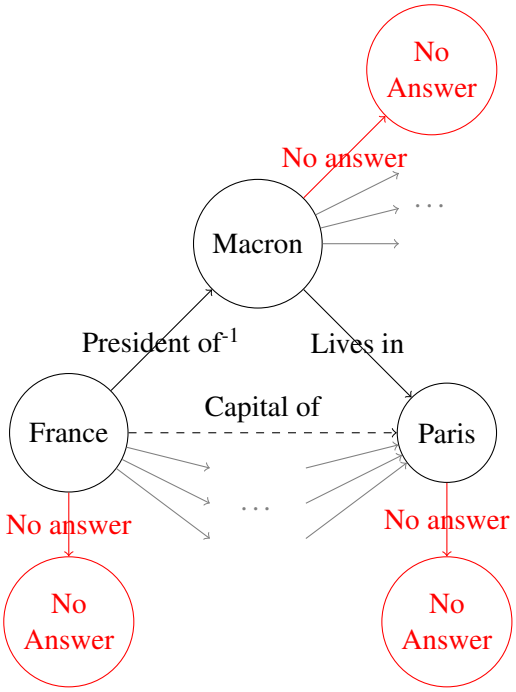


Figure 3: Fictional graph for the the question "What's the capital of France?". The relation $(Capital\ of)$ does not exist in the graph and thus an alternative path needs to be used that leads to the correct answer. To avoid that the agent returns an incorrect answer when not finding the correct answer, a 'no answer' relation is added between every entity node and a special 'no answer' node, to be able to return 'no answer'.

$$R(s_T) = \begin{cases} r_{pos} & \text{if } e_T = e_a, \\ 0 & \text{if } e_T = e_{NOANSWER}, \\ r_{neg} & \text{if } e_T \notin \{e_a, e_{NOANSWER}\} \end{cases} \quad (4)$$

with $r_{pos}>0$ and $r_{neg}<0$. The idea is that the agent receives a larger reward for a correct answer compared to not answering the question, and a negative reward for incorrectly answering a question compared to not answering the question. In the experimental section, we show that this mechanism provides better performance.

6 Experimental setup

We evaluate our proposed approach on a publicly available dataset, FB15k-237 (Toutanova and Chen, 2015) which is based on the Freebase knowledge graph and a proprietary dataset Alexa69k-378 which is a sample of Alexa's proprietary knowledge graph. Both the public dataset and the proprietary dataset are good examples of

Model	Hits@1	Hits@10	MRR	Precision	Answer Rate	QA Score
(Das et al., 2018)	0.217	0.456	0.293	0.217	**1**	0.357
(Lin et al., 2018)	0.329	0.544	0.393	0.329	**1**	0.495
RL	0.2475	0.4032	0.2983	0.2475	**1**	0.3968
Supervised	0.2474	0.4929	0.3276	0.2474	**1**	0.3967
Supervised + RL	0.2736	0.5015	0.3469	0.2736	**1**	0.4296
No Answer RL	0.2345	0.3845	0.2831	0.4011	0.5847	0.4758
All	0.2738	0.4412	0.3286	**0.4835**	0.5663	**0.5216**

Table 1: Results on FB15k-237 dataset.

Model	Hits@1	Hits@10	MRR	Precision	Answer Rate	QA Score
(Das et al., 2018)	0.1790	0.2772	0.2123	0.1790	**1**	0.3036
(Lin et al., 2018)	0.1915	0.3184	0.2358	0.1915	**1**	0.3214
RL	0.1677	0.2716	0.2031	0.1677	**1**	0.2872
Supervised	0.1471	0.3142	0.203	0.1471	**1**	0.2565
Supervised + RL	0.1937	0.3045	0.2312	0.1937	**1**	0.3245
No Answer RL	0.1564	0.2442	0.1858	**0.3892**	0.4019	0.3955
All	0.1865	0.294	0.2229	0.3454	0.5401	**0.4213**

Table 2: Results on Alexa69k-378 dataset.

			#queries	
#ent	#rel	#facts	valid	test
FB15k-237				
14,505	237	272,115	17,535	20,466
Alexa69k-378				
69,098	378	442,591	55,186	55,474

Table 3: Statistics of the datasets.

real-world general-purpose knowledge graphs that can be used for question answering. FB15k-237 contains 14,505 different entities and 237 different relations resulting in 272,115 facts. Alexa69k-378 contains 69,098 different entities and 378 different relations resulting in 442,591 facts. We follow the setup of Das et al. (2018), using the same train/val/test splits for FB15k-237. For Alexa69k-378 we use 10% of the full dataset for validation and test. For both datasets, we add the reverse relations of all relations in the training set in order to facilitate backward navigation following the approach of previous work. Similarly, a 'no op' relation is added for each entity between the entity and itself, which allows the agent to loop/reason multiple consecutive steps over the same entity. An overview of both datasets can be found in Table 3.

We extend the publicly available implementation of Das et al. (2018) for our experimentation. We set the size of the entity and relation representations d at 100 and the hidden state at 200. We use a single layer LSTM and train models with path length 3 (tuned using hyper-parameter search). We optimize the neural network using Adam (Kingma and Ba, 2015) with learning rate 0.001, mini-batches of size 256 with 20 rollouts per example. During the test time, we use beam search with the beam size of 100. Unlike Das et al. (2018), we also train entity embeddings after initializing them with random values. Reward values are set as $r_{pos} = 10$ and $r_{neg} = -0.1$ after performing a coarse grid search for various reward values on the validation set. For all experiments, we selected the best model with the highest QA Score on the corresponding validation set.

7 Results

The results of our experiments for FB15k-237 and Alexa69k-378 are given in Table 1 and Table 2 respectively.

Supervised learning For FB15k-237, we see that the model trained using reinforcement learning (RL) scores as well as the model trained using supervised learning. This makes supervised learning using DFS a strong baseline system for question answering over knowledge graphs, and for FB15k-237 in particular. On Alexa69k-378, models trained using supervised learning score lower on all metrics compared to RL. When combining supervised learning with RL overall performance increases.

No answer When we train RL system with our ternary reward structure (No Answer RL), the precision and QA score increase significantly on both datasets. For FB15k-237, our No Answer RL model decided not to answer over 40% of the questions, with an absolute hits@1 reduction of only 1.3% over standard RL. Moreover, of all the answered questions, 40.11% were answered correctly compared to 24.75% of the original question-answering system: an absolute improvement of over 15%. This resulted in the final QA Score of 47.58%, around 8% higher than standard RL and 12% higher than Das et al. (2018).

Similarly, 60% of the questions did not get answered on Alexa69k-378. This resulted in hits@1 decrease of roughly 1% but compared to standard RL, the precision increased from 16.77% to 38.92%: an absolute increase of more than 20%. The final QA Score also increased from 28.72% to 39.55%, and also significantly improved over Das et al. (2018) and Lin et al. (2018). The results indicate that using our method allows us to improve the precision of the question-answering system by choosing the right questions to be answered by not answering many questions that were previously answered incorrectly. This comes at the expense of not answering some questions that previously could be answered correctly.

All Finally, all methods were combined in a single method. First the model was pretrained in a supervised way. Then the model was retrained using RL algorithm with ternary reward structure. This jointly trained model obtained better QA scores than any individually trained model. On FB15k-237, a QA score of 52.16% is obtained which is an absolute improvement of 4.58% over the best individual model and 2.66% over Lin et al. (2018). Similarly, on Alexa69k-378, an absolute improvement of 2.57% over the best individual result is

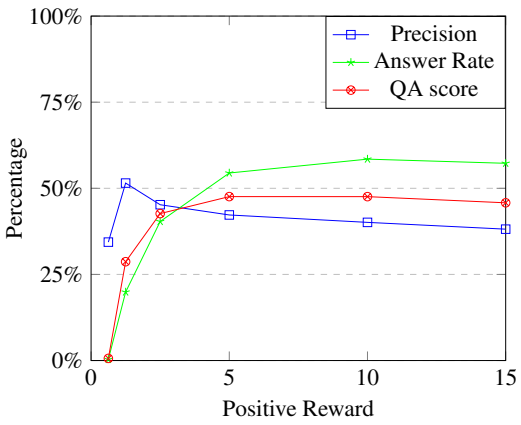


Figure 4: Influence of changing the positive reward for FB15k-237. The negative reward is fixed at $r_{neg} = -0.1$ and the neutral reward is fixed at $r_{neutral} = 0$.

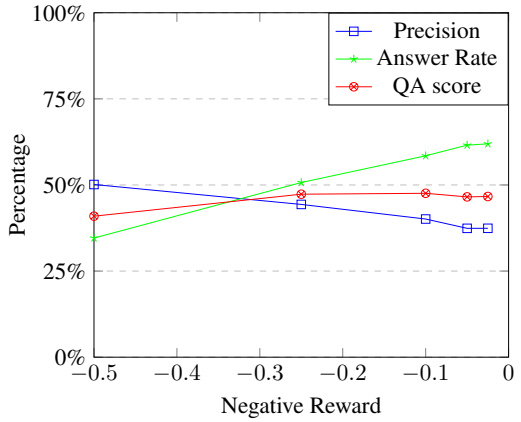


Figure 5: Influence of changing the negative reward for FB15k-237. The positive reward is fixed at $r_{pos} = 10$ and the neutral reward is fixed at $r_{neutral} = 0$.

obtained, almost 10% absolute improvement over Lin et al. (2018). Sample results from our method are given in Table 4 and Table 5.

Reward tuning An important part of increasing the QA score is to select the right combination of rewards. Therefore, we ran additional experiments where we varied either the positive or negative reward, keeping the other rewards fixed. In Figure 4, the precision, answer rate and QA score are shown when varying the positive reward and keeping the neutral and negative rewards fixed. When, the positive reward is very small ($r_{pos} = 0.625$), almost no question is answered. When the positive reward r_{pos} is 1.25, roughly 20% of the questions are answered with a 50% precision. After that, the precision starts declining and the answer rate

Question: e_q = Bruce Broughton, r_q = Profession. Answer: e_a = Music Composer
Bruce Broughton $\xrightarrow{\text{Award Nominee}}$ Oscar Best Music $\xrightarrow{\text{Award Winner}}$ Nino Rota $\xrightarrow{\text{Profession}}$ Music Composer
Question: e_q = Washington nationals, r_q = Sports Team Sport. Answer: e_a = Baseball
Washington Nationals $\xrightarrow{\text{Sports League}^{-1}}$ National League $\xrightarrow{\text{Sports League}}$ Milwaukee Braves $\xrightarrow{\text{Sports Team Sport}}$ Baseball

Table 4: Example paths of correctly answered questions on FB15k-237. Note that the fact (e_q, r_q, e_a) is not part of the KG.

Question: e_q = Sherlock holmes (movie), r_q = Story By. Answer: e_a = Conan Doyle
Sherlock holmes (movie) $\xrightarrow{\text{Film Crew Role}}$ Wardrobe Supervisor $\xrightarrow{\text{No op}}$ Wardrobe Sup. $\xrightarrow{\text{No op}}$ Wardrobe Sup.
Sherlock holmes (movie) $\xrightarrow{\text{Film Crew Role}}$ Wardrobe Supervisor $\xrightarrow{\text{No answer}}$ No answer $\xrightarrow{\text{No answer}}$ No answer

Table 5: Example question from FB15k-237, incorrectly answered by (Das et al., 2018) and not answered by our system. Note that the fact (e_q, r_q, e_a) is not part of the KG.

starts increasing, resulting in an overall increase in QA score. The QA score plateaus between 5 and 10 and then starts decreasing slowly. In Figure 5, the precision, answer rate and QA score are shown when varying the negative reward and keeping the neutral and positive rewards fixed. In this case, the highest QA score is achieved when the negative reward is between -0.25 and -0.1. As long as the negative reward is lower than zero, a wrong answer gets penalized and the QA score stays high.

8 Conclusions

In this paper, we addressed the limitations of current approaches for question answering over a knowledge graph that use reinforcement learning. Rather than only returning a correct or incorrect answer, we allowed the model to not answer a question when it is not sure about it. Our ternary reward structure gives different rewards for correctly answered, incorrectly answered and not answered questions. We also introduced a new evaluation metric which takes these three options into account. We showed that we can significantly improve the precision of answered questions compared to previous approaches, making this a promising direction for the practical usage in knowledge graph-based QA systems.

References

Antoine Bordes, Nicolas Usunier, Alberto Garcia-Duran, Jason Weston, and Oksana Yakhnenko. 2013. Translating embeddings for modeling multi-relational data. In *Advances in Neural Information Processing Systems (NIPS)*.

Wenhu Chen, Wenhan Xiong, Xifeng Yan, and William Yang Wang. 2018. Variational knowledge graph reasoning. In *Proceedings of the Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL-HTL)*.

Rajarshi Das, Shehzaad Dhuliawala, Manzil Zaheer, Luke Vilnis, Ishan Durugkar, Akshay Krishnamurthy, Alexander J. Smola, and Andrew McCallum. 2018. Go for a walk and arrive at the answer: Reasoning over paths in knowledge bases using reinforcement learning. *International Conference on Learning Representations (ICLR)*.

Tim Dettmers, Minervini Pasquale, Stenetorp Pontus, and Sebastian Riedel. 2018. Convolutional 2d knowledge graph embeddings. In *Proceedings of the 32th AAAI Conference on Artificial Intelligence (AAAI)*.

Fréderic Godin, Anjishnu Kumar, and Arpit Mittal. 2018. Using ternary rewards to reason over knowledge graphs with deep reinforcement learning. In *Workshop on Relational Representation Learning (NeurIPS)*.

Kelvin Guu, John Miller, and Percy Liang. 2015. Traversing knowledge graphs in vector space. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*.

Diederik P. Kingma and Jimmy Ba. 2015. Adam: A method for stochastic optimization. *In Proceedings of the International Conference on Representation Learning (ICLR)*.

Xi Victoria Lin, Richard Socher, and Caiming Xiong. 2018. Multi-hop knowledge graph reasoning with reward shaping. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing (EMNLP)*.

Arvind Neelakantan, Benjamin Roth, and Andrew McCallum. 2015. Compositional vector space models for knowledge base completion. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics and the International Joint Conference on Natural Language Processing (ACL-IJCNLP)*.

Pranav Rajpurkar, Robin Jia, and Percy Liang. 2018. Know what you don't know: Unanswerable questions for squad. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics (ACL)*.

Tim Rocktäschel and Sebastian Riedel. 2017. End-to-end differentiable proving. In *Advances in Neural Information Processing Systems (NIPS)*.

Richard S. Sutton and Andrew G. Barto. 1998. *Introduction to Reinforcement Learning*, 1st edition. MIT Press, Cambridge, MA, USA.

Kristina Toutanova and Danqi Chen. 2015. Observed versus latent features for knowledge base and text inference. In *3rd Workshop on Continuous Vector Space Models and Their Compositionality (ACL)*.

Kristina Toutanova, Victoria Lin, Wen-tau Yih, Hoifung Poon, and Chris Quirk. 2016. Compositional learning of embeddings for relation paths in knowledge base and text. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics (ACL)*.

Adam Trischler, Tong Wang, Xingdi Yuan, Justin Harris, Alessandro Sordoni, Philip Bachman, and Kaheer Suleman. 2017. Newsqa: A machine comprehension dataset. In *Proceedings of the 2nd Workshop on Representation Learning for NLP (ACL)*.

Théo Trouillon, Johannes Welbl, Sebastian Riedel, Éric Gaussier, and Guillaume Bouchard. 2016. Complex embeddings for simple link prediction. In *Proceedings of the International Conference on Machine Learning (ICML)*.

Ronald J. Williams. 1992. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine Learning*.

Wenhan Xiong, Thien Hoang, and William Yang Wang. 2017. Deeppath: A reinforcement learning method for knowledge graph reasoning. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing (EMNLP)*.

Fan Yang, Zhilin Yang, and William W. Cohen. 2017. Differentiable learning of logical rules for knowledge base reasoning. In *Advances in Neural Information Processing Systems (NIPS)*.

Yuyu Zhang, Hanjun Dai, Zornitsa Kozareva, Alexander J. Smola, and Le Song. 2018. Variational reasoning for question answering with knowledge graph. In *Proceedings of the AAAI Conference on Artificial Intelligence (AAAI)*.