TOWARD UNDERSTANDING TRANSFORMER BASED SELF SUPERVISED MODEL

Yizhou Zhang
zhangyiz@usc.edu

November 12, 2019

1 Introduction

Knowledge Base Question Answering (KB-QA) refers to benefit Question Answering (QA) systems by making use of information from a knowledge base, which contains facts that are usually summarized as triplets manually or automatically. This task is confronted of two major challenges: encoding or representing information from triplets in a knowledge base, and the reasoning based on the extracted information. For the former challenge, recent works try to handle it via new neural network architectures for the representation on graphs, like Gated Graph Neural Network. For the latter challenge, recent works proposed different frameworks. One of them is considering reasoning on a Knowledge Base as sequential decision and trained a model to handle it with reinforcement learning algorithm. Another framework is to use complicated bidirectional attention mechanism and enhancing module to integrate the reasoning process into the representation learning model.

2 Task Formulation and Datasets

2.1 Knowledge Base

Let $\mathcal{E}$ denote a set of entities and $\mathcal{R}$ denote a set of binary relations. Typical Knowledge Bases, like Freebase KB [1], are usually collections of facts stored as triplets (e_1, r, e_2) where $e_1, e_2 \in \mathcal{E}$ and $r \in \mathcal{R}$. e_1 is also known as triplet head while e_2 is known as triplet tail. Note that a Knowledge Base can be represented as a knowledge graph $\mathcal{G} = (V, E, \mathcal{R})$, where $V = \mathcal{E}$ and E are the vertices and directed edges of different types in the graph and $\mathcal{R}$ is the type space of edges.

Besides, some Knowledge Bases can also contain ternary relations, even though they store facts through triplets. For example, in Wikidata, the head entity of a triplet can be another triplet, like $((e_1, r_1, e_2), r_2, e_3)$.

Currently, Freebase, WordNet and Wikidata are the most popular Knowledge Base in many different downstream tasks. Based on the data from them, a lot of benchmark of different scale are constructed, like FB15K and WN18 for knowledge graph embedding.

2.2 Question Answering on Knowledge Base

Generally, on a Knowledge Base, there are two kinds of QA: Query Answering and Question Answering. In Query Answering, the query to be answered are usually structured and can be easily transformed to be a reasoning problem on a Knowledge Graph. This task is very meaningful because most of Knowledge Bases suffer from link missing. Figure 1 shows an example of Query Answering on Knowledge Base, where the relation between 'David Beckham' and 'Cruz Beckham' is missing. To answer such a query related to missing direct relation, a model must have reasoning ability on Knowledge Base, while it does not have to understand natural language as queries are usually organized structurally.

On the contrary, the Question Answering on Knowledge Base usually requires both reasoning ability and natural language understanding ability, because the question in it are usually unstructured sentence of natural language. Figure 2 shows a toy example of Question Answering. As natural language question may contain multiple constraint to the final answer as shown in the example, it's challenging to represent them and map them to the corresponding triplets in

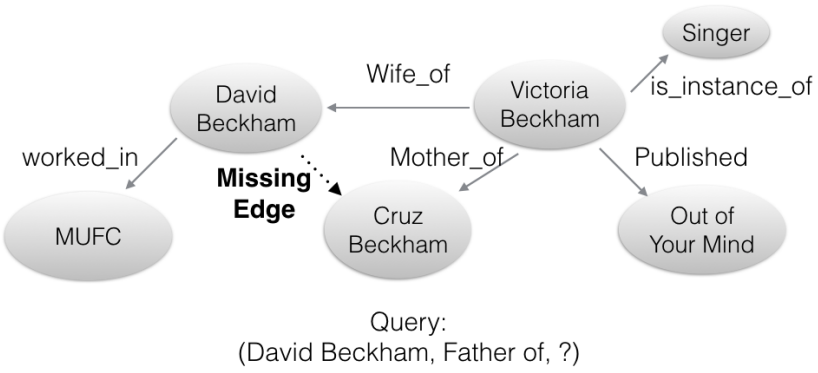

Figure 1: A toy example of query answering on a knowledge graph

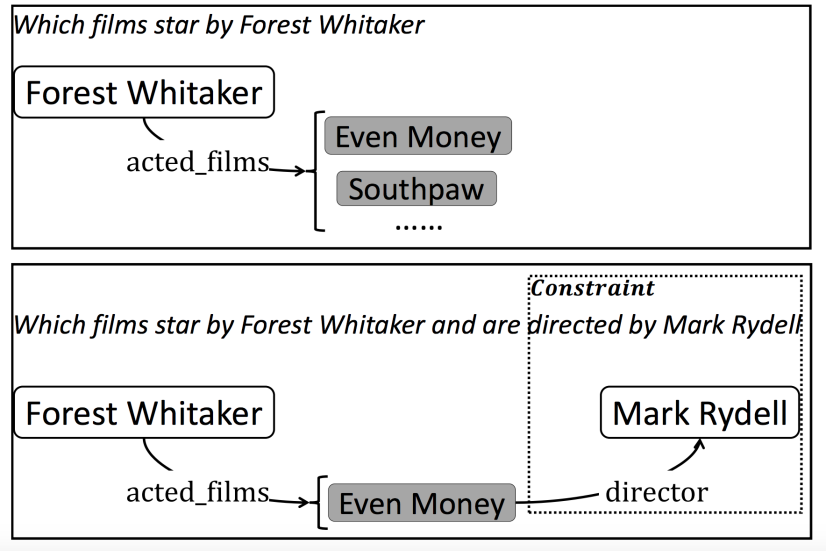

Figure 2: A toy example of question answering on a knowledge graph

Knowledge Base. An intuitive solution is to transform the question to structured query. However, transforming natural language to query is challenging due to a lot of problems like word and entity ambiguation.

Above two tasks have different benchmarks. The benchmarks for Query Answering are usually some tail prediction datasets on Knowledge Base, like FB15K and WN18. And the benchmarks for Question Answering are some natural language questions. Among them, `WebQuestions`[2][1] and its subsets are very popular. Besides, researcher also use some special datasets to evaluate the model, like `ComplexQuestions` which consists of a lot of questions with complicated constraint.

3 Knowledge Representation

The representation of Knowledge Base is especially challenging for Question Answering. Because most of popular Knowledge Graph representation learning models, like TransE, DistMult and RotatE are for general purpose and deal with the whole graph without specific attention to the content of question, their results are not applicable for Question Answering due to serious noise. Therefore, dedicated representation model for this task is required.

[1] nlp.stanford.edu/software/sempre

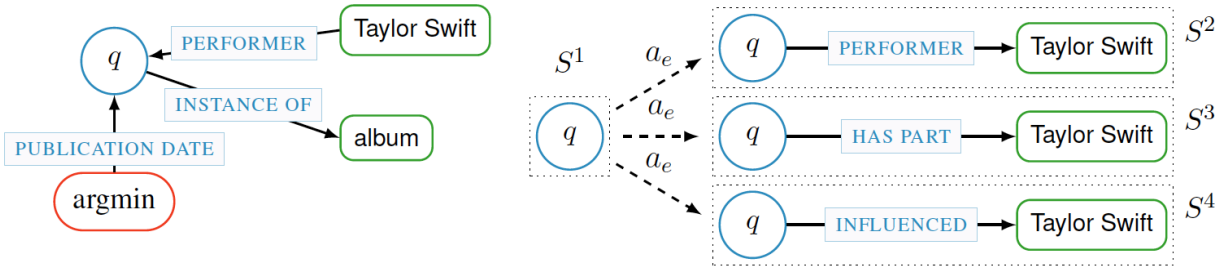

Figure 3: A toy example of semantic graph. The left figure is the semantic graph of a question: "What is the first album of Taylor Swift?" And the right figure shows three candidate semantic graphs of the question, where all candidate answers q are related to "Taylor Swift" with different relation.

3.1 Candidate Graph

A recent method of generating knowledge representation in light of question is to extract a set of candidate graphs from the Knowledge Base for the question. Figure 3 shows an example, which is a kind of candidate graph, named semantic graph, proposed by [3]. There are also other kinds of candidate graphs, like query graph and its extension[4, 5]. The center node q in a semantic graph is an answer candidate, while others correspond to the entities appearing in the question (sometimes entity linking technique is applied to help construct semantic graph) and some constraint to the answer (like for 'first' or 'firstly', the related time or date must be minimized). Each edge correspond the relation between the two nodes and its label indicates the relation type. Such semantic graphs store the relation between known entity and a candidate answer. By generating a representation of the semantic graph, model can calculate the ranking score for different candidate answers and select the best one. As some models consider two-hop relations, nodes that do not appears in the question but are related with both candidate answer and appearing nodes are also added into the graph.

3.2 Candidate Graph Representation

Recent works usually learn representation for candidate graphs via graph neural network (GNN). In [3], the authors propose to encode the question via a CNN and apply Gated Graph Neural Network (GGNN) to learn representation for semantic graphs. GGNN is a kind of graph neural network with recurrent architecture, which updates the representation of nodes based on information aggregated from their neighbors in each iteration. Original GGNN takes the graph structure of the candidate graph and the feature vector of nodes as input. However, as the semantic graphs contain different relations, the authors also assign each relation an embedding vector as input feature. Both of the node feature and relation feature are calculated based on their labels in Wikidata, denoted as l. For an entity or a relation, the authors tokenize its label as a set of word and calculate the sum of the word embedding vectors (in this paper, they use GloVe), denoted as w_{sum}. Then, a fully connected layer transform the word embedding to a single feature vector:

$$h_l = \tanh(W_l w_{sum} + b_l) \tag{1}$$

Then, the hidden states of nodes are initialized with their feature vectors $h_v^0 = h_{l_v}$. And relation features are further transformed to two vectors representing incoming edges and outgoing edges respectively:$h_r^{in} = W_{in} h_{l_r}$,$h_r^{out} = W_{out} h_{l_r}$.

In an iteration t, the information from neighbor nodes and incoming and outgoing edges are first aggregated as a vector a_v^t:

$$a_v^t = \sum_{j \in \mathcal{N}(v)} h_j^{t-1} + \sum_{e \in E_{in}(v)} h_{r_e}^{in} + \sum_{e \in E_{out}(v)} h_{r_e}^{out} \tag{2}$$

where $E_{in}(v)$ and $E_{out}(v)$ are the sets of incoming and outgoing edges of node v respectively and r_e is the relation type of e. Then the hidden states of node v is updated based on following equations with similar form as Gated Recurrent Unit (GRU):

$$z_v^t = \sigma(W^z a_v^t + U^z h_v^{t-1} + b^z) \tag{3}$$

$$r_v^t = \sigma(W^r a_v^t + U^r h_v^{t-1} + b^r) \tag{4}$$

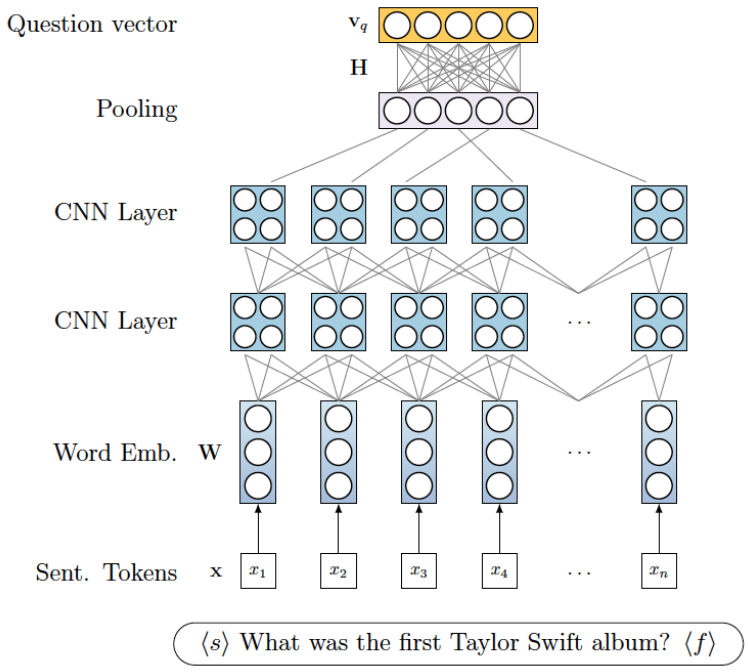

Figure 4: The process of encoding question sentence.

$$\hat{h}_v^t = \sigma(Wa_v^t + U(r_v^t \odot h_v^{t-1}) + b) \tag{5}$$

$$h_v^t = (1 - z_v^t) \odot h_v^{t-1} + z_v^t \odot h_v^t \tag{6}$$

where σ is a sigmoid function. After T iterations, the final hidden vector of center node is transformed through a fully connected layer with relu activation: $v_g = ReLU(W_g h_q^T + b_g)$. Then the question sentence is encoded to a vector v_q via a CNN as shown in Figure 4. The cosine similarity of two vectors $\gamma(v_q, v_g)$ is calculated as the ranking score. The whole model is optimized via a marginal loss function:

$$\mathcal{L} = \sum_{g \in C} max(0, m - \gamma(v_q, v_g^+) + \gamma(v_q, v_g^-)) \tag{7}$$

where C is the set of all candidate semantic graphs of a question, $\gamma(v_q, v_g^+)$ is the ranking score of golden truth and $\gamma(v_q, v_g^-)$ is the ranking score of a negative sample.

4 Reasoning on Knowledge Base

In this section, we will illustrate two advanced research of Knowledge Base (Graph) reasoning for Query Answering and Question Answering respectively.

4.1 Reasoning for Query Answering

Given a query $(e_1, r, ?)$, the Knowledge Base reasoning for query answering can be formulated as searching a path P from e_2 to another node on the corresponding Knowledge Graph and guarantee that P is equivalent to relation r. This formulation is equivalent to a sequential decision process. Consequently, reinforcement learning can be applied to train a model that can search the neighborhood of the head node via random walk efficiently and stop at the answer node[6].

4.1.1 Reinforcement Learning Environment

In the authors' design, the model will stop searching after T steps. So the environment of answer searching is a "finite horizon" (which mean the process has a terminal state), "deterministic partially observed Markov decision process that lies on the Knowledge Graph $\mathcal{G}$". Therefore, this process can be formulated as a 5-tuple $(\mathcal{S}, \mathcal{O}, \mathcal{A}, \sigma, \mathcal{R})$:

- States: The state space $\mathcal{S}$ consists of all valid combination (e_t, e_1, r_q, e_2), where e_t is the current entity node, e_1 is the head node, r is the query relation and e_2 is the node the model is searching for.
- Observations: As the answer node can not be observed by the model, the observation set $\mathcal{O}$ only consist of all valid (e_t, e_1, r), which the model make decisions based on.
- Actions: The set of possible actions $\mathcal{A}$ in a State $S = (e_t, e_1, r_q, e_2)$ consist of all outgoing edges from e_t and staying on the current node e_t. Formally $\mathcal{A}_S = \{(e_t, r, v) \in E : S = (e_t, e_1, r_q, e_2)\} \cup \{(e_t, \phi, e_t)\}$, where E is the set of outgoing edges on e_t and v is a neighbor of e_t.
- Transition: The environment transits deterministically immediately after the action of the model. As the action answer and query are both deterministic, the environment evolves as $\sigma(S, A) = (v, e_1, r_q, e_2)$, where $A = (e_t, r, v)$ and $S = (e_t, e_1, r_q, e_2)$.
- Rewards: The model only acquire reward of +1 when it is on the right answer after T steps.

4.1.2 Policy Network

The policy network of the model encode the history information (decisions and observations) at step t as a continuous hidden state h_t via LSTM:

$$h_t = LSTM(h_{t-1}, [a_{t-1}; o_t]) \tag{8}$$

where a_{t-1} is the embedding of the outgoing relation selected by the model in step $t-1$ and o_t is the embedding of the observed e_t at step t. Then a 2-layer MLP calculate the distribution from which a discrete action is sampled:

$$d_t = softmax(\mathbf{A_t}(W2ReLU(W1[h_t : o_t; r_q]))) \tag{9}$$

$$A_t \sim Categorical(d_t) \tag{10}$$

where $\mathbf{A_t}$ is the embedding matrix of all possible actions. The i-th raw of $\mathbf{A_t}$ is the concatenation of the embedding of r_i and v (the relation and tail node of the i-th action in $\mathcal{A}_{S_t}$)

Then, the authors use policy descent algorithm train the model to find correct answers

4.2 Reasoning for Question Answering

Unlike the Query Answering with structural query, Question Answering requires the reasoning model to calculate the representation of information from Knowledge base while understanding the natural language question. Therefore, the model with good reasoning ability need interaction between knowledge representation and question encoding. In a recent paper[7], the authors propose a reasoning model with bidirectional attention module and enhancing module. The overview of the model is shown in Figure 5. The input module transform the question to a sequence of hidden states $\mathbf{H}^Q$. At the same time, the memory module encode three types of information (the type of candidate, the path from candidate answer to topical entity in the question, the neighbor entity of candidate answer in KB) from a candidate answer as three key-value vector pairs $(M_i^k t, M_i^v t)$,$(M_i^k p, M_i^v p)$,$(M_i^k e, M_i^v e)$ respectively.

Then the bidirectional attention module helps calculate the importance of candidate answer for the summary of KB in light of question and the importance of each token in the question in light of the KB. And the enhancing module updates the representation of candidate answers based on question representation while improving the encoding of question based on KB summary.

4.3 Bidirectional Attention

4.3.1 KB-aware Attention Module

The overview of the KB-aware attention module is shown in 6. A BiLSTM with self-attention operation is applied on the hidden state sequence to encode the whole sequence as a single vector q:

$$\begin{aligned} q &= BiLSTM([H^Q A^{QQ^T}, H^Q]) \\ A^{QQ} &= softmax((H^Q)^T H^Q) \end{aligned} \tag{11}$$

where A^{QQ} is the self-attention matrix. Then another attention module calculates the importance of each candidate answer to the question based on q and use the importance score to calculate the summary of the answer type m_t, answer

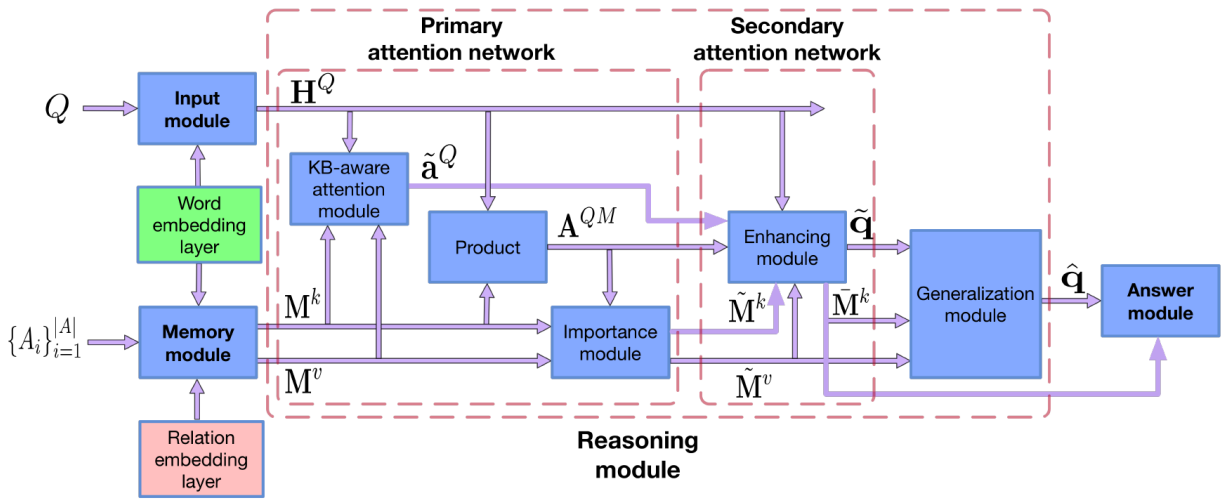

Figure 5: The overview of reasoning module for question answering.

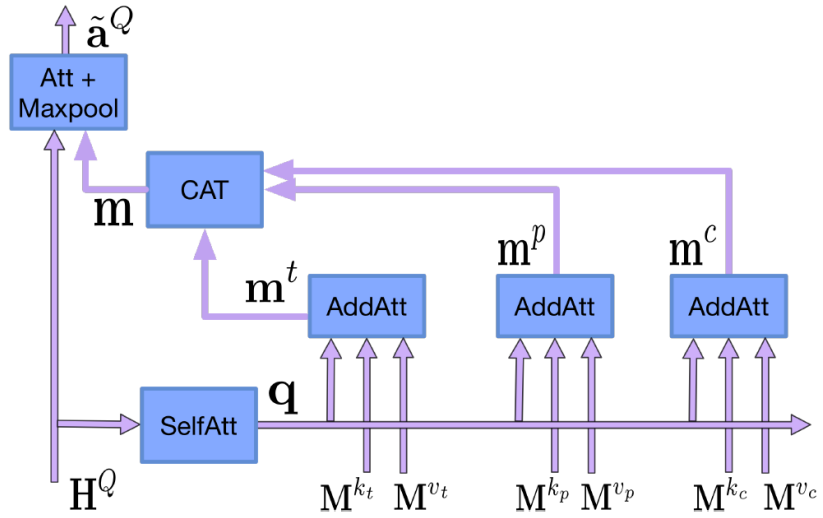

Figure 6: The overview of KB-aware Attention Module.

path m_p, answer neighbor m_e from the KB:

$$m_x = \sum_{i \in A} a_i^x \dot{M}_i^{vx}$$
$$a^x = Att_{add}(q, M^{kx}) = softmax(tanh([q^T, M^{kx}]W_1)W_2) \tag{12}$$

where A is the candidate answer set and $x \in \{t, p, e\}$. So far we have obtained a summary of the KB $m = [m_t; m_p, m_e]$. Then we can compute the attention from each word i in question to different KB aspects, formulated as $A^{Qm} = H^{Q^T} m$. Then the model applies max-pooling over the last dimension (the dimension of KB aspects, whose number is 3) and get a vector a^Q which is as long as the question sequence. The intuition of max-pooling is that each word in the question contribute to a specific purpose (like decide the type of answer, relation between answer and topical entities and so on). So, the max-pooling over the last dimension helps find the purpose of each word. Then a softmax function normalizes a^Q to $\hat{a}^Q$ and get the importance of each word to the question encoding in light of KB summary.

4.3.2 Importance Module

The importance module calculate the importance of different KB aspects by measuring their relevance to the questions. The model start by computing a $|Q| \times |A| \times 3$ attention tensor A^{QM}, which indicates the strength of connection between each word-answer pair in each aspect (t,p,e):

$$A_{ijx}^{QM} = H_i^Q \dot{M}_j^{kx} \tag{13}$$

Then, for each answer-aspect pair, we select the word that maximize its strength. Then use softmax to normalize it. In this way, we can calculate the normalized importance of each aspect to an answer. Then, by applying this importance to update the representation of the answer, we acquire the answer representation in light of question

$$\begin{aligned} A_{jx} =& A_{ijx}^{QM}, i = \underset{i}{\operatorname{argmax}} A_{ijx}^{QM} \\ \hat{A}_{jx} =& softmax([A_{jt}; A_{jp}; A_{je}]) \\ \hat{M}_i^k =& \sum_{x \in \{t,p,e\}} \hat{A}_{jx} M_i^k \\ \hat{M}_i^v =& \sum_{x \in \{t,p,e\}} M_i^v \end{aligned} \tag{14}$$

4.4 Enhancing Module

To enhance the interaction between answer representation and question representation, the authors add another two-way attention. First, by applying max-pooling on the last dimension of A^{QM} (the dimension of KB aspects, whose number is 3), we preserve the aspect in which the connection of each word-answer pair is maximized and get A_M^Q. We normalize each row of A_M^Q to get the attention of each word to each answer $\hat{A}_M^Q$. Then, we update question representation sequence $\hat{H}^Q = H^Q + \hat{a}^Q \odot \hat{A}_M^Q \hat{M}^v$. The final representation of question is calculated as $\hat{q} = H^Q \hat{a}^Q$

Similarly, we can also update the question-enhanced answer representation $\overline{M}^k$. First, we transpose use softmax to normalize each row of the transposed matrix of A^{QM} and acquire A_Q^M, indicating the importance of each word to each answer. Then we calculate the answer representation $\overline{M}^k$ via

$$\begin{aligned} \hat{a}^M =& (\hat{A}_M^Q)^T \hat{a}^Q \\ \overline{M}^k =& \hat{M}^k + \hat{a}^M \odot (A_Q^M (\hat{H}^Q)^T) \end{aligned} \tag{15}$$

With above two representations, the generalization module and answer module apply some tricks like batch normalization on representation to get final result.

References

[1] Kurt Bollacker, Colin Evans, Praveen Paritosh, Tim Sturge, and Jamie Taylor. Freebase: A collaboratively created graph database for structuring human knowledge. In *Proceedings of the 2008 ACM SIGMOD International Conference on Management of Data*, SIGMOD '08, pages 1247–1250, New York, NY, USA, 2008. ACM.

[2] Jonathan Berant, Andrew Chou, Roy Frostig, and Percy Liang. Semantic parsing on Freebase from question-answer pairs. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 1533–1544, Seattle, Washington, USA, October 2013. Association for Computational Linguistics.

[3] Daniil Sorokin and Iryna Gurevych. Modeling semantics with gated graph neural networks for knowledge base question answering. In *COLING*, 2018.

[4] Wen-tau Yih, Ming-Wei Chang, Xiaodong He, and Jianfeng Gao. Semantic parsing via staged query graph generation: Question answering with knowledge base. 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)*, pages 1321–1331, Beijing, China, July 2015. Association for Computational Linguistics.

[5] Junwei Bao, Nan Duan, Zhao Yan, Ming Zhou, and Tiejun Zhao. Constraint-based question answering with knowledge graph. In *Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers*, pages 2503–2514, Osaka, Japan, December 2016. The COLING 2016 Organizing Committee.

[6] Rajarshi Das, Shehzaad Dhuliawala, Manzil Zaheer, Luke Vilnis, Ishan Durugkar, Akshay Krishnamurthy, Alex Smola, and Andrew McCallum. Go for a walk and arrive at the answer: Reasoning over paths in knowledge bases using reinforcement learning. In *ICLR*, 2018.

[7] Yu Chen, Lingfei Wu, and Mohammed J. Zaki. Bidirectional attentive memory networks for question answering over knowledge bases. In *NAACL-HLT*, 2019.