11-2019

Multi-hop knowledge base question answering with an iterative sequence matching model

Yunshi LAN
Singapore Management University, yslan.2015@phdis.smu.edu.sg

Shuohang WANG
Singapore Management University, shwang.2014@phdis.smu.edu.sg

Jing JIANG
Singapore Management University, jingjiang@smu.edu.sg

Multi-hop Knowledge Base Question Answering with an Iterative Sequence Matching Model

Yunshi Lan
School of Information System
Singapore Management University
Singapore, Singapore
yslan.2015@phdis.smu.edu.sg

Shuohang Wang
School of Information System
Singapore Management University
Singapore, Singapore
shwang.2014@phdis.smu.edu.sg

Jing Jiang
School of Information System
Singapore Management University
Singapore, Singapore
jingjiang@phdis.smu.edu.sg

***Abstract*—Knowledge Base Question Answering (KBQA) has attracted much attention and recently there has been more interest in multi-hop KBQA. In this paper, we propose a novel iterative sequence matching model to address several limitations of previous methods for multi-hop KBQA. Our method iteratively grows the candidate relation paths that may lead to answer entities. The method prunes away less relevant branches and incrementally assigns matching scores to the paths. Empirical results demonstrate that our method can significantly outperform existing methods on three different benchmark datasets.**

***Index Terms*—Knowledge base question answering, Sequence matching model, Multi-hop question answering**

I. Introduction

Answering questions in natural language using information stored in a structured knowledge base such as Freebase [1], YAGO [2], DBpedia [3] has been attracting much attention in recent years. The task is often known as KBQA. Here a KB (knowledge base) is usually represented as a graph where nodes are entities and edges are relations. Figure 1 shows an example of a question and part of a knowledge base from which the answer to the question can be found. KBQA systems can be potentially used in many domains and useful for building virtual assistants such as Google Duplex and Amazon Alexa.

Previous work on KBQA largely focused on simple questions which can be answered from a single relation connecting two entities in the KB [4, 5, 6, 7]. For example, the question "who is Sylvia Brett's other half" can be answered solely from the triplet $(sylvia\ brett, spouse, charles\ vyner\ brooke)$ in the KB. However, questions in real applications can be more complex and require multiple hops of relations to answer. The question shown in Figure 1, for example, requires a relation path of 3 hops in the KB, i.e., $(spouse \rightarrow parent \rightarrow place\ of\ birth)$, in order to reach a correct answer. Clearly such questions are much harder to handle, because the correct answers are multiple hops away in the KB from the entity appearing in the question, leading to a much larger search space.

We refer to this kind of KBQA problem where the answer entities are multiple hops away in the KB from the entities in the questions the *multi-hop KBQA* problem. Recently, there have been a few attempts to tackle the multi-hop KBQA problem, together with a few benchmark datasets released. For example, Zhang et al. (2017) proposed a variational reasoning method that recursively traverses the entities in a KB to predict their probabilities as correct answers. Zhou et al. (2018) proposed an interpretable reasoning network for multi-hop KBQA.

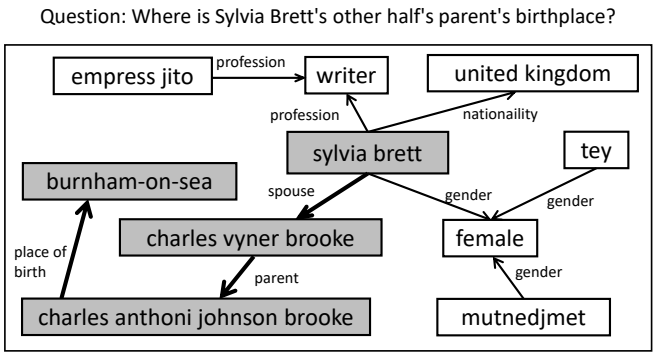


Fig. 1: An example question and a subset of a KB that contains the answer to the question. The topic entity from the question is "sylvia brett". The shaded boxes show the path of entities and relations that leads to the correct answer.

Although these studies proposed novel ideas to address multi-hop KBQA and showed promising results, they have a number of limitations. In this paper, we propose an iterative sequence matching model to address these limitations and empirically show that our method can outperform the previous methods. First, the two existing studies [8, 9] consider a large number of candidate answers or candidate relation paths (albeit in a probabilistic way). This not only makes the methods less efficient but also makes it harder to rank the candidates, because the model has to learn to separate the correct answers from many competing wrong answers. In contrast, we propose to iteratively grow the candidate relation paths that will eventually lead to the candidate answers, and at each iteration we prune away branches that are unlikely to lead to a correct answer. This allows our model to focus on differentiating the correct answers from only those competing candidates that are the most confusing. Our experiments show that indeed this iterative pruning approach works better than

a baseline that considers all candidates.

Second, when it comes to matching a question with a relation path that leads to a candidate answer, Zhang et al. (2017) and Zhou et al. (2018) encoded both the question and the candidate answer as a single embedding vector, and then the two vectors' dot product is computed to measure their similarity. Several previous studies have shown that this kind of sequence matching is not as effective as a match-aggregate sequence matching framework [10, 11, 12]. In our method, we adopt a match-aggregate framework to match the question with a candidate answer's sequence representation. In addition, because we iteratively grow the relation paths leading to candidate answers, we propose a novel incremental sequence matching model to efficiently compute the sequence matching scores without having to revisit the earlier relations in a relation path. Our experiments show that indeed this incremental sequence matching mechanism works better than standard sequence matching.

Third, the two previous studies made some strong assumptions about the problem setup. Zhang et al. (2017) assumed that the number of hops needed to answer a question is known in advance, which seriously limits the applicability of the method. In contrast, we propose a mechanism to automatically determine the number of hops needed. Zhou et al. (2018) proposed two versions of their method, and for the better performing version, it is assumed that the sequences of relations leading to the correct answers are known during training time. In contrast, our method does not require such information for training and yet we can outperform their method.

Specifically, our method consists of three modules: iterative path growth, incremental sequence matching, and termination check. The three modules work together to iteratively consider relation paths of 1 hop, 2 hops, etc. and iteratively match these paths with the question in order to rank these paths. The method automatically detects when to terminate the iterations. We conduct experiments on three recently released multi-hop KBQA datasets and find that our method performs significantly better than existing methods. We also conduct ablation studies to analyze the contributions of the different components of our method, and we find that both the idea of iteratively pruning the relation paths and the idea of incrementally computing the matching scores are important for our performance gain.

II. Related Work

KBQA is a task that has been well studied. There are generally two lines of work on KBQA: semantic parsing-based and embedding-based. Semantic parsing-based methods [13, 14, 15, 16, 17, 18] first map questions to logical forms, such as λ-DCS [19, 20], excutable SPARQL or SQL queries [15, 21, 22], graph-based queries [4, 23]. Then a programmer is designed to execute the queries within the KB. Once the question has been transformed into the logic form, the answer could be immediently retrieved from the KB. While they are theoretically sound, these methods require manually annotated logical forms for training and are domain-specific. It could be easily limited within small datasets. Recently embedding-based methods have attracted much attention. This method first identifies a topic entity. The entities which are 1 hop or 2 hops away from the topic entity are retrieved from the KB as the candidate answers. Then the challenge becomes how to rank these candidate answers. A number of methods are proposed to solve this problem. Bordes et al. (2014) took the subgraphs that surround the candidate answers as features to rank. Dong and Lapata (2016) proposed Multi-Column Convolutional Neural Networks, which convolutionally matches the question with different parts of the subgraph. Yu et al. (2017) developed a HR-BiLSTM model to match the question with relation paths of candidate answers at word level and token level. The more recent work [26] defined a special slot which contains the information of grammatical structure of the question and took advantage of it to improve the ranking accuracy. Other methods could be found from these papers [5, 25, 27, 28].

Most of early work focuses on single-relation questions, but in practice, we may encounter more complex questions. The multi-hop questions are one kind of complex questions. Recently, some reading comprehension datasets involve the multi-hop questions which could be answered using multiple ducuments [29, 30]. Several advanced methods [31, 32] have been proposed to solve the multi-hop reading comprehension task. The multi-hop questions are also worth to be explored in KBQA task. So far, there has been only limited work on multi-hop KBQA. Weston et al. (2015) applied Memory Network to KBQA, where relation triplets are saved in memory and queries are updated at every hop by interacting with the memory. The Key-Value Memory Network method [34] improves Memory Network by dividing the memory into key and value parts. Zhang et al. (2017) proposed a variational reasoning network that integrates entity linking and relation prediction together and performs reasoning by traversing within the KB. Zhou et al. (2018) proposed an interpretable reasoning network, which performs relation matching hop-by-hop and has shown good performance on some datasets. However, as we discussed earlier, these methods have several limitations. We propose a new method in this paper to address these limitations. A recently published work [35] proposed a general framework to address the multi-hop question answering task, which achieved outstanding results for evaluation. The difference between their method and ours is that they re-visit the ranked paths during the expansion of the relation paths while we propose a more complex mechanism to remember the visited paths.

III. Our Method

A. Problem Definition

We assume that a knowledge base (or knowledge graph) is defined over a set of entities $\mathcal{E}$ and a set of relations $\mathcal{R}$. The knowledge base is a set of triplets, which we use $KB = \{(e, r, e')\}$ to represent, where $e, e' \in \mathcal{E}$ and $r \in \mathcal{R}$. We also

assume that each entity $e \in \mathcal{E}$ or relation $r \in \mathcal{R}$ has a sequence of words as its textual representation.

A question q is a sequence of words. We assume that entity detection and linking has been done by an entity linking tool and a *topic entity* $e_0 \in \mathcal{E}$ has been identified inside q. Our goal is to find an entity $a \in \mathcal{E}$ that answers the question, and generally speaking, we expect that this answer a is linked to e_0 in the knowledge graph through one or more hops of relations, and these relations between e_0 and a correspond to what is expressed in q. For example, given the question in Figure 1, the topic entity is *sylvia brett*. The sequence of relations between *sylvia brett* and the correct answer entity *burnham-on-sea* is $(\textit{spouse}, \textit{parent}, \textit{place of birth})$, and we can see that these relations collectively correspond to what is expressed in the question.

For training, we assume that we have a set of (q, a) pairs but we do not know which path of relations in the KB leads to the answer a from the topic entity e_0 in q.

B. Method Overview

The general idea behind our method is to find answer entities that are linked to the topic entity through one or more hops of relations in the knowledge graph. For single-hop KBQA, previous methods typically exhaustively enumerate all the relation paths originating from the topic entity and match them with the question in order to find the best answer. For multi-hop KBQA, exhaustively enumerating all relation paths would lead to too many candidates, which would not only affect efficiency but also making the candidate ranking task harder because of the many competing wrong candidates. Our method iteratively grows the candidate paths and prunes away those paths that are unlikely to lead to the correct answers at each iteration. We also design a novel incremental sequence matching method to score the candidate paths as well as to help check when to terminate the iterations.

Our method consists of three modules: (1) an *iterative path growth* module, (2) an *incremental sequence matching* module and (3) a *termination check* module.

C. Iterative Path Growth

The iterative path growth module grows the candidate paths up to T hops, one hop at each iteration. At the end of each iteration, it keeps only the top-K candidate paths that best match the question.

Let us first define some terms and notation to facilitate the discussion.

Candidate path: Formally, given a question q, we first detect its topic entity e_0. A *candidate path* is the sequence of entities and relations along a path that starts from e_0 in the knowledge graph. Let us use $p = (e_0, r_1, e_1, r_2, e_2, \ldots, r_t, e_t)$ to represent a candidate path with t hops of relations. Here, for all $1 \leq l \leq t$, $(e_{l-1}, r_l, e_l) \in \textit{KB}$.

Candidate path set: We define the *candidate path set after the t-th iteration* to be the set of the top-K candidate paths we keep at the end of the t-th iteration of our method. We use $\mathcal{P}^{(t)}$ to represent this candidate path set. Note that each $|\mathcal{P}^{(t)}| = K$ and each $p \in \mathcal{P}^{(t)}$ has t hops of relations.

Algorithm 1 Iterative Path Growth

Input: *KB*, question q, topic entity e_0, number of hops T
Output: $\mathcal{P}^{(T)}$
Initialize: $\mathcal{P}^{(0)} \leftarrow \{(e_0)\}$
for $t = 1, 2, \ldots, T$ **do**
  $\tilde{\mathcal{P}}^{(t)} \leftarrow \emptyset$
  for each $p \in \mathcal{P}^{(t-1)}$ **do**
    $e_{t-1} \leftarrow \textit{tail}(p)$
    for each $(e, r, e') \in \textit{KB}$ such that $e = e_{t-1}$ **do**
      $p' \leftarrow p \oplus (r, e')$ ▷ sequence concatenation
      $\tilde{\mathcal{P}}^{(t)} \leftarrow \tilde{\mathcal{P}}^{(t)} \cup \{p'\}$
  score and rank elements in $\tilde{\mathcal{P}}^{(t)}$
  $\mathcal{P}^{(t)} \leftarrow$ top-K elements in $\tilde{\mathcal{P}}^{(t)}$

Tail entity: The tail entity of a candidate path, denoted as $\textit{tail}(p)$, is the last entity in the candidate path p.

Our iterative path growth module works as follows. In the beginning, starting from e_0, we identify all the relations linked to e_0 in the knowledge graph. This gives us the initial candidate path set $\mathcal{P}^{(1)}$. Subsequently, at the t-th iteration, for each $p \in \mathcal{P}^{(t-1)}$, we identify all the relations linked to $\textit{tail}(p)$ in the knowledge graph and use them to grow p by one hop of relation. This gives us multiple new candidate paths, each with t hops of relations. Call this set of candidate paths $\tilde{\mathcal{P}}^{(t)}$. We then use the sequence matching module (presented in the next section) to score and rank the paths in $\tilde{\mathcal{P}}^{(t)}$, and keep the top-$K$ paths to form $\mathcal{P}^{(t)}$.

For example, given the topic entity *sylvia brett* in Figure 1, we first construct $\mathcal{P}^{(1)}$ that contains the following candidate paths: (*sylvia brett*, *profession*, *writer*), (*sylvia brett*, *nationality*, *united kingdom*), (*sylvia brett*, *gender*, *female*), (*sylvia brett*, *spouse*, *charles vyner brooke*). Next, during the second iteration, we grow each of these candidate paths to construction $\tilde{\mathcal{P}}^{(2)}$. Given the subset of the KB shown in Figure 1, $\tilde{\mathcal{P}}^{(2)}$ contains the following candidate paths: (*sylvia brett*, *profession*, *writer*, *profession*$^{-1}$, *empress jito*),[1] (*sylvia brett*, *gender*, *female*, *gender*$^{-1}$, *tey*), (*sylvia brett*, *gender*, *female*, *gender*$^{-1}$, *mutnedjmet*), (*sylvia brett*, *spouse*, *charles vyner brooke*, *parent*, *charles anthoni johnson brooke*). But after pruning away the less relevant branches, $\mathcal{P}^{(2)}$ may contain only (*sylvia brett*, *spouse*, *charles vyner brooke*, *parent*, *charles anthoni johnson brooke*).

The algorithm is formally defined in Algorithm 1.

D. Incremental Sequence Matching

The objective of the incremental sequence matching module is to assign a score to each candidate path p such that we can rank the paths in $\tilde{\mathcal{P}}^{(t)}$. The score should reflect how well a path p matches the question q.

Since both the question and a candidate path are sequences, normally we could employ a standard sequence matching

[1] We use a single symbol "$^{-1}$" to denote the reverse of a relation.

model such as the ones commonly used in natural language inference [10], paraphrase detection [12] and machine comprehension [36]. However, because our candidate paths grow iteratively, at the t-th iteration, when we need to match a candidate path $p \in \tilde{\mathcal{P}}^{(t)}$ with the question, the prefix of p up to the $(t-1)$-th relation has already been matched with the question in previous iterations. Therefore, it makes sense for us to only match the last relation of p with the question and aggregate the matching score at this iteration with the matching scores from previous iterations. Therefore, our sequence matching mechanism is *incremental*.

We first further introduce some notation to facilitate our discussion.

- To enable sequence matching, we represent question q as $\mathbf{Q} = (\mathbf{q}_1, \mathbf{q}_2, \ldots, \mathbf{q}_m)$, where $\mathbf{q}_i$ is the embedding vector of the i-th word in q.
- For a candidate path $p = (e_0, r_1, e_1, \ldots, r_t, e_t)$, we represent it as $\mathbf{P}^{(t)} = (\mathbf{w}_{r_t,1}, \mathbf{w}_{r_t,2}, \ldots, \mathbf{w}_{r_t,n})$, where $\mathbf{w}_{r_t,j}$ is the embedding vector of the j-th word in the textual representation of r_t.[2] Note that here we only consider r_t and ignore the other relations in p because our matching mechanism is incremental, i.e., at the current iteration, we should focus on the matching between r_t and q. Also note that we ignore the entities $e_1, e_2, \ldots$ along the path because we do not expect these entities to be mentioned in the question. Our preliminary experiments also confirmed that including these entities was not useful.
- For $p \in \tilde{\mathcal{P}}^{(t)}$, let $p_{(t-1)}$ denote the prefix of p up to the $(t-1)$-th relation. That is, if $p = (e_0, r_1, e_1, \ldots, r_{t-1}, e_{t-1}, r_t, e_t)$, then $p_{(t-1)}$ is defined as $(e_0, r_1, e_1, \ldots, r_{t-1}, e_{t-1})$.

We now describe the incremental sequence matching module. At the t-th iteration, let p be a candidate path from $\tilde{\mathcal{P}}^{(t)}$ (the set of candidate paths that need to be scored and ranked), and let $\mathbf{P}^{(t)}$ be the representation of p. We first use a standard BiLSTM to process $\mathbf{Q}$ and obtain $\overline{\mathbf{Q}} = (\overline{\mathbf{q}}_1, \overline{\mathbf{q}}_2, \ldots)$. Essentially $\overline{\mathbf{q}}_i$ represents the i-th word in the question together with its contextual information. We can similarly obtain $\overline{\mathbf{P}}^{(t)} = (\overline{\mathbf{w}}_{r_t,1}, \overline{\mathbf{w}}_{r_t,2}, \ldots)$ using BiLSTM.

Next, we compute two sets of attention weights, one normalized across $\mathbf{Q}$ and the other normalized across $\mathbf{P}^{(t)}$, as shown below:

$$e_{i,j} = F(\overline{\mathbf{q}}_i)^\intercal F(\overline{\mathbf{w}}_{r_t,j}), \quad (1)$$

$$\alpha_{i,j} = \frac{\exp(e_{i,j})}{\sum_{j'=1}^{n} \exp(e_{i,j'})}, \quad (2)$$

$$\beta_{i,j} = \frac{\exp(e_{i,j})}{\sum_{i'=1}^{m} \exp(e_{i',j})}, \quad (3)$$

where $F(\cdot)$ is a single non-linear layer with ReLU as its activation function.

Now to measure how well $\mathbf{P}^{(t)}$ matches $\mathbf{Q}$, for each word $\mathbf{q}_i$ in $\mathbf{Q}$, we derive an attention-weighted sum of $\overline{\mathbf{P}}^{(t)}$ as follows:

[2]For $\mathbf{P}^{(1)}$, we include the words in the textual representations of both e_0 and r_1.

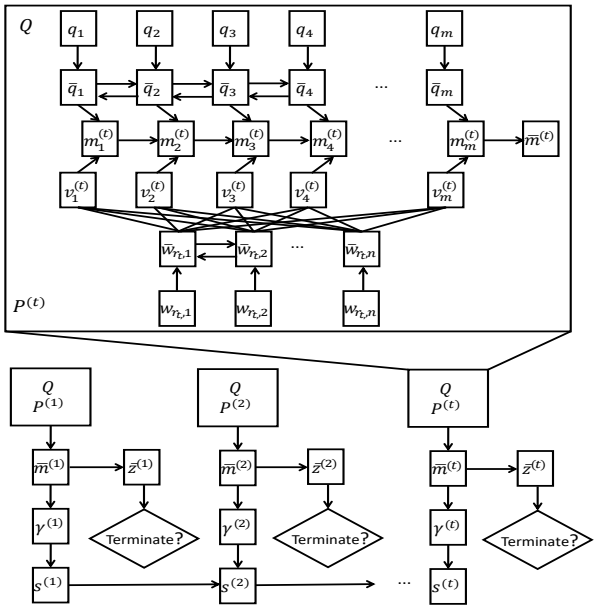


Fig. 2: The incremental sequence matching model.

$$\mathbf{v}_i^{(t)} = \sum_{j=1}^{n} \alpha_{i,j} \cdot \overline{\mathbf{w}}_{r_t,j}.$$

Intuitively, we can compare $\overline{\mathbf{q}}_i$ with $\mathbf{v}_i^{(t)}$ to measure how well the i-th word in the question is matched in the current iteration by relation r_t. However, recall that we are doing incremental matching. $\overline{\mathbf{q}}_i$ may have been previously matched with some other words in $p_{(t-1)}$, and if so, it may not be so critical to match $\overline{\mathbf{q}}_i$ with r_t in the current iteration.

To capture this intuition, we borrow an idea from neural machine translation [37, 38], where it is important not to re-translate a word in the source sentence when sequentially generating the target sentence. We define a scalar value $a_i^{(t)}$ to remember how well $\overline{\mathbf{q}}_i$ has been matched in path p up to the t-th iteration. Specifically, we set $a_i^{(0)} = 0$. We then define

$$a_i^{(t)} = a_i^{(t-1)} + \sum_{j=1}^{n} \beta_{i,j}.$$

Note that $\beta_{i,j}$ is as defined above in Eqn. (3), based on matching the question with r_t in the t-th iteration. Intuitively, $\sum_{j=1}^{n} \beta_{i,j}$ represents how well $\mathbf{q}_i$ has been matched by all the words in r_t, as compared with other words in $\mathbf{Q}$.

Now with $a_i^{(t)}$ clearly defined, let us define the following matching vector:

$$\mathbf{m}_i^{(t)} = \begin{bmatrix} a_i^{(t-1)} \\ \overline{\mathbf{q}}_i \odot \mathbf{v}_i^{(t)} \\ (\overline{\mathbf{q}}_i - \mathbf{v}_i^{(t)}) \odot (\overline{\mathbf{q}}_i - \mathbf{v}_i^{(t)}) \end{bmatrix}, \quad (4)$$

where $\odot$ represents element-wise multiplication of two vectors. Note that using $\mathbf{v}_1 \odot \mathbf{v}_2$ and $(\mathbf{v}_1 - \mathbf{v}_2) \odot (\mathbf{v}_1 - \mathbf{v}_2)$

to represent how well vectors $\mathbf{v}_1$ and $\mathbf{v}_2$ match has been commonly used in previous work [11, 12]. Here we add $a_i^{(t-1)}$ in the matching vector in order to take previous matching results into consideration. Our preliminary experiments also show that not including $a_i^{(t-1)}$ here would markedly affect the results.

Given the sequence $(\mathbf{m}_1^{(t)}, \mathbf{m}_2^{(t)}, \ldots, \mathbf{m}_m^{(t)})$, we use an LSTM to process the sequence followed by maxpooling to derive a single vector $\overline{\mathbf{m}}^{(t)}$:

$$\overline{\mathbf{m}}^{(t)} = \text{Max-Pool}(\text{LSTM}(\mathbf{m}_1^{(t)}, \mathbf{m}_2^{(t)}, \ldots, \mathbf{m}_m^{(t)})). \quad (5)$$

We then use this vector to derive a matching score between $\mathbf{Q}$ and $\mathbf{P}^{(t)}$ as follows:

$$\gamma^{(t)} = \mathbf{w}^\intercal \overline{\mathbf{m}}^{(t)} + b, \quad (6)$$

where the vector $\mathbf{w}$ and scalar b are parameters to be learned.

Note that although $\gamma^{(t)}$ has implicitly encoded the matching results of previous relations in p with the question through $a_i^{(t-1)}$ in Eqn. (4), it does not tell us whether each relation in p is critical. When we score the entire path p, we would want to promote those paths in which each segment is highly relevant to the question. To do so, we define the final score for p to be the product of $\gamma^{(t)}$ with $\gamma^{(t-1)}$, $\gamma^{(t-2)}$ and so on. Let $s^{(t)}(p)$ denote the complete matching score between candidate path p and the question. $s^{(t)}(p)$ can be recursively defined as follows: $s^{(0)}(\cdot) = 1$. For p with t relations,

$$s^{(t)}(p) = s^{(t-1)}(p_{(t-1)}) \cdot \gamma^{(t)}. \quad (7)$$

The scoring function $s^{(t)}(\cdot)$ is then used to rank the candidate paths in $\tilde{\mathcal{P}}^{(t)}$ in order to derive $\mathcal{P}^{(t)}$.

Figure 2 illustrates how our incremental sequence matching model works.

E. Termination Check

We now describe how our method determines when to terminate the iterations. Intuitively, if a candidate path p has matched the entire question q well, then we can terminate the iterations. Based on the matching method described above, the vector $\overline{\mathbf{m}}^{(t)}$ encodes how well p matches q. To turn it into a single value to facilitate our termination checking, we first define the following score $z(p) \in [0, 1]$ for each candidate path p at the t-th iteration:

$$z^{(t)}(p) = \sigma(\mathbf{v}^\intercal \overline{\mathbf{m}}^{(t)} + c),$$

where the vector $\mathbf{v}$ and scalar c are parameters to be learned. Then we take the maximum z among all the paths in $\mathcal{P}^{(t)}$. This implicitly leverages the $z^{(t)}(p)$ of the best-matched candidate path.

$$\overline{z}^{(t)} = \max_{p \in \mathcal{P}^{(t)}} z^{(t)}(p).$$

During training we learn the parameters $\mathbf{v}$ and c for z. During prediction time, we compare $\overline{z}^{(t)}$ with a threshold τ to determine when to stop the iterations.

F. Loss Function

We now describe the loss function we use during training. The parameters of our model that need to be learned include the following: the parameters of the various LSTMs we use, the parameters of the function $F(\cdot)$ used in Eqn. (1), the parameters $\mathbf{w}$ and b in Eqn. (6) and $\mathbf{v}$ and c in Eqn. (7). We train these parameters in an end-to-end fashion.

For the loss function, we consider two factors. First, we would like the candidate path that leads to the correct answer entity or entities to be ranked higher than the other paths in the candidate path set. Second, we want the value $\overline{z}^{(t)}$ to be close to 1 if we should terminate at the t-hop and close to 0 if we still need to continue to grow the paths.

Specifically, consider the candidate path set $\mathcal{P}^{(t)}$. For each $p \in \mathcal{P}^{(t)}$, by comparing the tail entity (or tail entities if the sequence of relations in p leads to more than one tail entities) with the ground truth answer entity or entities, we can calculate the F1 score of this path p. We then normalize these scores using the following formula:

$$\hat{g}(p) = \frac{\text{F1}(p)}{\sum_{p' \in \mathcal{P}^{(t)}} \text{F1}(p')}.$$

We can think of $\hat{g}(p)$ as the empirical probability for us to choose p among all candidate paths in $\mathcal{P}^{(t)}$.

On the other hand, we derive the probability for p from our model:

$$g(p) = \frac{\exp(s^{(t)}(p))}{\sum_{p' \in \mathcal{P}^{(t)}} \exp(s^{(t)}(p'))},$$

where $s^{(t)}(\cdot)$ is as defined in Eqn. (7).

We use the KL-divergence between $\hat{g}(\cdot)$ and $g(\cdot)$ as the first part of our loss function:

$$L_1 = \sum_{p \in \mathcal{P}^{(t)}} g(p) \ln \frac{g(p)}{\hat{g}(p)}.$$

A second goal of our loss function is to help termination check. Based on the ground truth answers, if at iteration t one of the candidate paths in $\mathcal{P}^{(t)}$ can give a F1 score of 1, then we consider t to be the last iteration. We also cap the number of iterations at a constant T. So if t reaches T before we see any F1 score of 1, we also consider this to be the last iteration. Let $\hat{t}$ represent the ground truth number of iterations. We can then define the second part of our loss function as follows:

$$L_2 = -\left(\log(\overline{z}^{(\hat{t})}) + \sum_{t=1}^{\hat{t}-1} \log(1 - \overline{z}^{(t)})\right).$$

Our final loss funtion is

$$L = L_1 + L_2.$$

IV. EXPERIMENTS

A. Data Sets

To evaluate the method we have proposed, we conduct experiments using three recently released datasets designed for multi-hop KBQA.

	MetaQA	PathQuestion	WC2014
#Train	100k	5688	6416
#Dev	50k	722	758
#Test	10k	696	780
#Pattern	47	128	12
#Entities	40k	2256	1127
#Relations	18	26	12
#Triplets	134k	4050	3977
% 1-hop	25.3	0	80.5
% 2-hop	38.0	25.4	19.5
% 3-hop	36.7	75.6	0

TABLE I: Some statistics of the three datasets. The first section shows the number of the questions in different splits and the question patterns. The second section shows the number of the entities, relations and triplets in the associated KB. The third section shows the percentage of the different hop questions.

MetaQA (MoviE Text Audio QA): This dataset was introduced by Zhang et al. (2017).[3] It contains more than 400K questions in the movie domain which require either 1-hop, 2-hop or 3-hop reasoning. While the original dataset separates questions of different hops, we mix all questions together for our evaluation. We take the vanilla text version of the dataset.
PathQuestion: This dataset was used in [9].[4] The questions were created by first identifying the relation paths between pairs of entities in a KB followed by generating natural language questions based on these paths using templates. Some further post-processing was done to vary the questions to make them more real. We mix the 2-hop and 3-hop questions in PathQuestions for our evaluation.
WC2014 (WorldCup2014): This dataset was created by Zhang et al. (2016).[5] It contains a mixture of 1-hop and 2-hop questions related to 2014 World Cup.

Some statistics of the datasets are shown in Table I.

In all the original KBs, relation triplets are directional. To simplify our path growth module, we add new edges to the KBs by reversing the direction of each triplet and adding the suffix *INV* to the relation description. For example, from $(\textit{sylvia brett}, \textit{profession}, \textit{writer})$, we create $(\textit{writer}, \textit{profession INV}, \textit{sylvia brett})$ and add it the the KB.

B. Experiment Setup

Although the three datasets already have topic entities annotated, here for fair comparison we perform our own entity linking based on simple string matching. Due to the simple pattern of the question, we have achieved near-perfect entity linking accuracy on the three datasets. We do not provide more details here because entity linking is not the focus of this paper.

It is worth noting that when we train on the PathQuestion dataset, since there are many 3-hop questions where the answer entity is also directly connected to the topic entity in the KB but the relation between them is not relevant to the question, our method would mistakenly treat these 1-hop connections as correct candidate paths. For example, "The place of birth of parent of Henri Victor Regnault's offspring", the golden relation path "*Henri Victor Regnault* $\xrightarrow{children}$ *Henri Regnault* $\xrightarrow{parent}$ *Henri Victor Regnault* $\xrightarrow{place\ of\ birth}$ *Aachen*" and fake golden relation path "*Henri Victor Regnault* $\xrightarrow{place\ of\ birth}$ *Aachen*" both give us correct answers, if we follow the same criteria to supervise our model, the fake relation path cheat the model to obtain an early stop signal. Such scenarios happen a lot in PathQuestion dataset. To avoid this problem, for PathQuestion, we also make use of the ground truth hop numbers to supervise our model. Note that even with this setting on PathQuestion, we are not using any more information for training than previous methods by Zhang et al. (2018) and by Zhou et al. (2018),

We use the Adagrad optimizer [40] with an initial learning rate of 0.01. We use 300 as the dimension of all word embedding vectors. Word embeddings are initialized via GloVe [41]. All hyper-parameters are tuned on the development data. All hidden dimensions in the model are set to 200 after tuning it among $\{100, 150, 200, 250\}$. The dropout ratio is set to 0 after tuning it among $\{0, 0.1, 0.2, 0.3, 0.4\}$. For our iterative sequence matching method, the threshold τ is set to 0.5 after tuning it among $\{0.3, 0.5, 0.8\}$. Finally, we set $T = 3$ and $K = 3$.

We use accuracy of the top-1 predicted answer entity as our evaluation metric, where a predicted answer is considered correct if it is one of the ground truth answers. This metric is the same as the one used by Zhou et al. (2018) and essentially the same as % hits@1 used by Zhang et al. (2017). Note that in the case when our top-ranked candidate path in the last iteration leads to more than one answer entities, we randomly pick one of these answer entity as the top-1 predicted answer entity.

C. Main Results

We first show the comparison of the following methods on the three datasets:
VRN: This is the Variational Reasoning Network method proposed by Zhang et al. (2017). Since their code is not publicly available, we take their reported performance on MetaQA. Note however that their method assumes that the correct number of hops to answer a question is known, which we do not assume we have except for PathQuestion.
IRN: This is the Interpretable Reasoning Network proposed by Zhou et al. (2018). We re-implemented this method that uses strong supervision, i.e., using the ground truth relation path for supervision.
IRN-Cons: When re-implementing the IRN method, we realized that this method does not restrict the relation paths to

[3]https://github.com/yuyuz/MetaQA
[4]https://github.com/zmtkeke/IRN
[5]https://github.com/zmtkeke/IRN

	MetaQA	PathQuestion	WC2014
VRN	59.6/-	-/-	-/-
IRN	17.0/8.8	86.9/80.2	90.7/68.4
IRN-Cons	21.8/9.2	89.8/82.9	92.6/70.5
MemNN	12.0/6.4	87.1/55.6	90.7/46.6
KVMemNN	16.6/6.5	88.0/56.3	90.5/47.1
Ours	**98.6/98.1***	**96.7/96.0***	**99.9/99.9***

TABLE II: %Hits@1/F1 scores of various methods on the three datasets. Note that for VRN, we took the reported performance in [8], and we do not have its performance on the other datasets. $*$ indicates that the result is statistically significantly better than the best baseline for that dataset at 0.05 significance value based on McNemar test.

	1-hop	2-hop	3-hop
VRN	82.0	75.6	38.3
IRN	9.0	8.3	31.2
IRN-Cons	14.6	10.7	38.2
MemNN	7.0	11.3	16.0
KVMemNN	6.2	12.6	27.9
Ours	96.3	99.1	99.6

TABLE III: %Hits@1 performance on different questions in MetaQA.

	MetaQA	PathQuestion	WC2014
ES-Siamese	95.7/93.7	89.2/88.8	94.9/90.1
ES-MatchAgg	97.0/96.3	91.5/90.9	96.3/92.1
IS-Prune	97.5/96.5*	92.7/91.8	99.7/99.7*
Ours	**98.6/98.1***	**96.7/96.0***	**99.9/99.9***

TABLE IV: Ablation experiment results. $*$ indicates that the result is statistically significantly better than ES-MatchAgg at 0.05 significance value based on McNemar test.

only those that are connected to the topic entities. We therefore implemented an improved version of IRN by imposing such a constraint, which we call IRN-Cons.

MemNN: This is the Memory Network method proposed by Weston et al. (2015) for KBQA. Following work [6], the memory contains all relevant triplets of topic entities. In our implementation, we include triplets up to 3 hops away from the topic entities in the memory.

KVMemNN: This is the Key-Value Memory Network method [34]. It improves MemNN by splitting memory into two parts: key and value. The key stores the subject entity and relation, and the value stores the object entity. When we train this model and MemNN, we set hop number as 3.

Ours: This is our overall method that iteratively searches the space of candidate paths while pruning away branches with low scores. It also uses our incremental sequence matching to score the candidate paths and our termination check mechanism to determine when to stop the iterations.

Table II shows the comparison between these methods on the three datasets. From the results, we can observe the following: (1) First of all, our method clearly outperforms all the baseline methods on all three datasets consistently. This demonstrates the effectiveness of our method. (2) Our method in general achieves very high accuracy values on all datasets. This is probably because these datasets were to a large extent generated from templates. Using a neural network model with enough complexity and sufficient training data, it is possible for the model to capture the patterns of these templates. (3) We can see that IRN, IRN-Cons, MemNN and KVMemNN perform poorly on MetaQA although they can perform well on PathQuestion and WC2014. We think there are a few reasons for this. First, the MetaQA dataset has a much larger KB (see Table I) and thus a larger search space. This shows that these methods cannot easily scale with large KB. Second, both the PathQuestion and WC2014 datasets have a large bias to questions with certain hop number (see Table I) while we sample questions of MetaQA evenly. This shows instead of memorizing the major hop number, our method could detect the hop number accurately. Figure 3a verifies that our termination check mechanism performs well on these datasets even with a shrunken searching space.

Next, we show the %hits@1 of these different methods on MetaQA when we group questions by the number of hops needed. The results are shown in Table III. We can see that for VRN, the performance is high on 1-hop questions and gradually drops for 2-hop and 3-hop questions. This is reasonable as longer questions are generally harder to answer. For IRN, IRN-Cons, MemNN and KVMemNN, their performance increases as the number of hops increases. This is because these methods cannot automatically detect the correct number of hops needed, and in our implementation we force them to always take $T = 3$ hops. Therefore, they end up performing better for 3-hop questions. In contrast, our method automatically detects when to stop the iterations and thus performs consistently well for 1-hop, 2-hop and 3-hop questions.

D. Ablation Studies

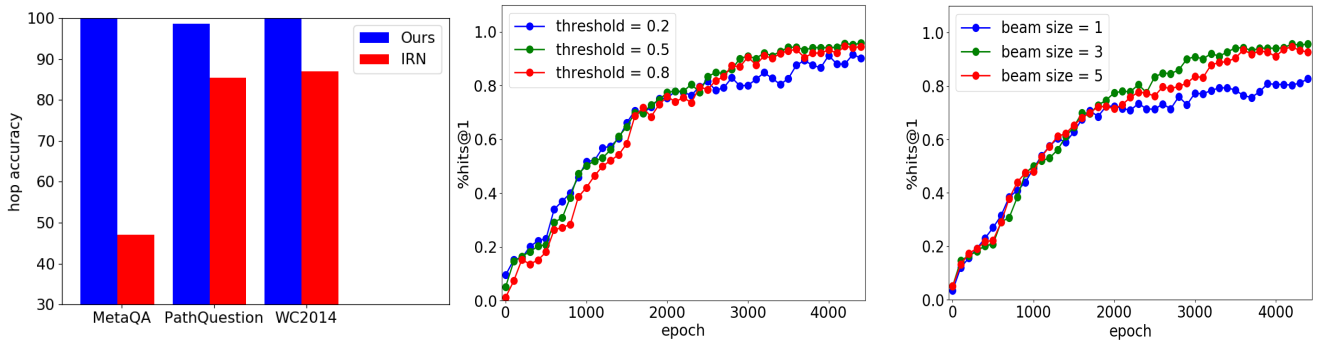
Next, we conduct some ablation studies to test whether each component of our method is necessary. Specifically, we compare with the following variants of our method:

ES-Siamese: This is a basic method that exhaustively searches all candidate paths up to T hops for the best answer entity, i.e., there is no pruning as in our method. When ranking the different paths, a traditional Siamese architecture is used where both the question and the candidate path are each separately encoded into a single vector before the two vectors are matched.

ES-MatchAgg: This method also performs the exhaustive search as ES-Siamese but uses a match-aggregate framework for sequence matching.

Is-Prune: This method uses our iterative path growth mechanism together with the path pruning mechanism. However, it does not use the incremental sequence matching model. Instead, it uses a standard match-aggregate matching method as in ES-MatchAgg. The purpose of this baseline is to measure the effect of pruning.

Table IV shows the ablation experiment results. We can observe the following. (1) ES-MatchAgg is better than ES-



(a) (b) (c)

Fig. 3: (a) Hop number accuracy on three datasets of IRN and Ours methods. (b) Epoch and %hits@1 on test set of MetaQA dataset with thresholds $\tau = 0.2$, $\tau = 0.5$, $\tau = 0.8$ respectively. (c) Epoch and %hits@1 on test set of MetaQA dataset with beam sizes $K = 1$, $K = 3$, $K = 5$ respectively.

Siamese, which shows that match-aggregate sequence matching is useful for our problem. (2) IS-Prune is better than ES-MatchAgg. It is worth noting that pruning is not only to reduce the search space but also to improve the model by restricting the negative candidate paths within the ones that are very similar to the correct candidate paths and the most confusing incorrect answers. Thus our iterative and pruning-based path growth mechanism has the benefit of training a more accurate matching function. (3) Our overall method is better than IS-Prune, showing that the incremental sequence matching mechanism is also effective. It's worth noting that even ES-Siamese baseline could outperform the IRN, IRN-Cons, MemNN and KVMemNN a lot. We suspect the main reason is that above comparable methods simply represent the question as the unordered bag-of-words and represent the relation or entity by a single embedding without any pre-training. Such methods are not strong enough to select the best-matched relation paths from a large scale of candidates while ES-Siamese baseline can do it using the word-level embedding and expressive encoding method.

E. Effect of Threshold and Beam Size

Threshold τ and beam size K are two important hyper-parameters in our method. To see how they influence the results, we draw the line plot figures. These figures display the changes of %hits@1 with the increase of training epochs regarding different τ or K. Figure 3b displays the effect of the threshold. In Eqn. (6), $\gamma^{(t)}$ is a indicator which measures how much information of question is already matched by candidate sequences at the t-th iteration, when $\gamma^{(t)}$ is larger than a threshold τ, we terminate the iteration and extract the answer. So when $\tau = 0.2$, it's easier to stop the iteration and its performance is relatively good at the beginning, but it falls behind with the increase of training epochs. when $\tau = 0.8$, it's harder to stop the iteration at the start of training. Among these three values of threshold, 0.5 achieves best %hits@1 score at last.

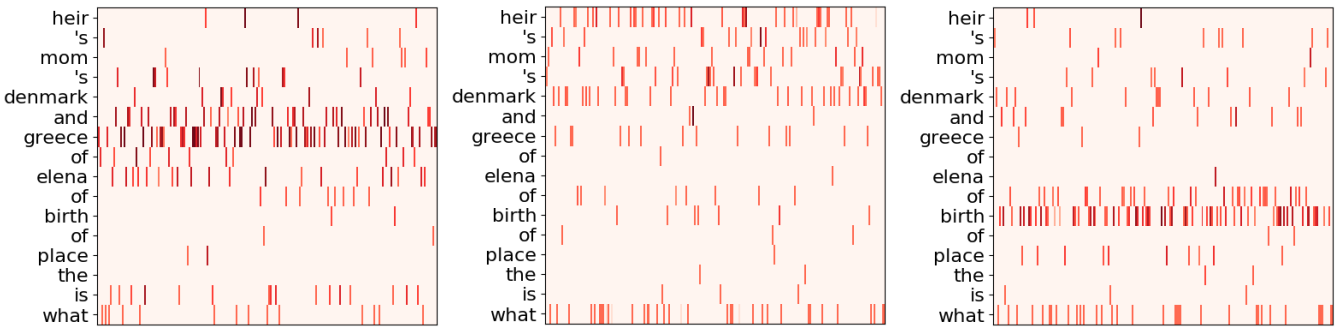
In Figure 3c, we show the effect of the beam size K. As we can see, when $K = 3$, the performance increases with relatively fast speed and it achieves the best accuracy at last. However, when $K = 1$, the accuracy improves relatively slowly and it's more likely to trap into a local optimum. But increasing K means that we need to sacrifice efficiency, and larger K doesn't always provide better accuracy (See $K = 5$). So this is a trade off for us to choose a proper K. In our experiment, we select K as 3.

F. Visualization

We visualize parts of the incremental matching model to understand its working mechanism.

In Eqn. (5), the matched sequence is processed by a LSTM. After that, the most important words are maintained and the other words are filtered by the maxpooling function. In Figure 4, we first draw the heatmaps of the output of LSTM. For the position whose value is the maximum along a dimension, we keep the value. Otherwise, we assign 0 to it. If we squeeze the heatmap vertically, we will obtain the exact vectors $\overline{\mathbf{m}}^{(1)}$, $\overline{\mathbf{m}}^{(2)}$ and $\overline{\mathbf{m}}^{(3)}$. So we denote them as expanded versions of $\overline{\mathbf{m}}^{(1)}$, $\overline{\mathbf{m}}^{(2)}$ and $\overline{\mathbf{m}}^{(3)}$. Based on such figures, we could tell which sub-question the model is handling at each iteration. Figure 4a displays that at the 1st iteration, the maximum values are mostly distributed among the phrase "elena of greece and denmark's mom". At the 2nd iteration, the maximum values are distributed among "'s heir" and the 3rd iteration has maximum values around "place of birth". This indicates that for a multi-hop question, our model tries to split them to sub-questions and solve sub-questions in the right order.

With similar intuition, we draw the Figure 5 to show the value of $a_i^{(t)}$ associated with each of its words for two hops of matching. The words in the dark color are the ones that have been matched up to that iteration. We can see that at the 1st iteration, the phrase "catherine dolgorukov's spouse" has been well matched. At the 2nd iteration, the phrase "father of catherine dolorukov's spouse" has been matched. This accumulative value indeed records how much information of the question we have matched or answered, which could provide clue to decide the termination time.



(a) (b) (c)

Fig. 4: Visualization of expanded versions of (a) $\overline{\mathbf{m}}^{(1)}$, (b) $\overline{\mathbf{m}}^{(2)}$, (c) $\overline{\mathbf{m}}^{(3)}$ respectively for example question "What is the place of birth of Elena of Greece and Denmark's mom's heir?". The darker color indicates the larger value.

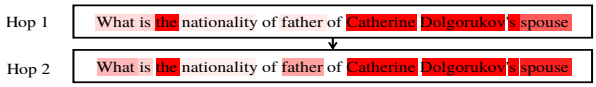


Fig. 5: Visualization of $\mathbf{a}^{(1)}$ and $\mathbf{a}^{(2)}$ for an example sentence after matching with relation *catherine dolgorukov spouse* and *parent* respectively.

G. Error Analysis

Even though our termination check mechanism works well in most cases, we noticed that there are a few wrong stops, which may come from redundant relation matching of the question. For example, in the PathQuestion dataset, the "husband" in the question "what religious belief does Rani Mangammal's husband have" has been matched twice. Other errors mainly come from incorrect relation matching. Around 74% of the errors with such mis-matching happens at the first hop for 1-hop, 2-hop or 3-hop questions, which might be due to the fact that the first hop relation requires more complex dependency analysis of the question.

V. Conclusions

In this paper, we proposed a novel iterative sequence matching model for multi-hop KBQA. Our method iteratively grows candidate relation paths, prunes away unrelated paths and also automatically detects the end of iterations, which could make the candidate relation path ranking procedure more effective and accurate. Our method achieved state-of-the-art performance on three multi-hop KBQA benchmarks. We further do some analysis to demonstrate the working mechanism of our method [6].

VI. Acknowledgment

This research is supported by the National Research Foundation, Prime Ministers Office, Singapore under its International Research Centres in Singapore Funding Initiative.

[6]We release our code at https://github.com/lanyunshi/Multi-hopQA.

References

[1] K. Bollacker, C. Evans, P. Paritosh, T. Sturge, and J. Taylor, "Freebase: A collaboratively created graph database for structuring human knowledge," in *Proceedings of the SIGMOD*, 2008, pp. 1247–1250.

[2] F. M. Suchanek, G. Kasneci, and G. Weikum, "Yago: a core of semantic knowledge," in *Proceedings of the WWW*, 2007, pp. 697–706.

[3] S. Auer, C. Bizer, G. Kobilarov, J. Lehmann, R. Cyganiak, and Z. Ives, "Dbpedia: a nucleus for a web of open data," in *Proceedings of the ISWC*, 2007, pp. 722–735.

[4] W.-t. Yih, X. He, and C. Meek, "Semantic parsing for single-relation question answering," in *Proceedings of the ACL*, 2014, pp. 643–648.

[5] A. Bordes, S. Chopra, and J. Weston, "Question answering with subgraph embeddings," in *Proceedings of the EMNLP*, 2014, pp. 615–620.

[6] A. Bordes, N. Usunier, S. Chopra, and J. Weston, "Large-scale simple question answering with memory networks," *arXiv:1506.02075*, 2015.

[7] C. Ran, W. Shen, J. Wang, and X. Zhu, "Domain-specific knowledge base enrichment using wikipedia tables," in *Proceedings of the ICDM*, 2015, pp. 349–358.

[8] Y. Zhang, H. Dai, Z. Kozareva, A. J. Smola, and L. Song, "Variational reasoning for question answering with knowledge graph," in *Proceedings of the AAAI*, 2017, pp. 6069–6076.

[9] M. Zhou, M. Huang, and X. Zhu, "An interpretable reasoning network for multi-relation question answering," in *Proceedings of the COLING*, 2018, pp. 2010–2022.

[10] A. P. Parikh, O. Täckström, D. Das, and J. Uszkoreit, "A decomposable attention model for natural language inference," in *Proceedings of the EMNLP*, 2016, pp. 2249–2255.

[11] S. Wang and J. Jiang, "A compare-aggregate model for matching text sequences," in *Proceedings of the ICLR*, 2017.

[12] Z. Wang, H. Wael, and F. Radu, "Bilateral multi-perspective matching for natural language sentences," in

Proceeding of the IJCAI, 2017, pp. 4144–4150.

[13] L. Zettlemoyer and M. Collins, "Learning context-dependent mappings from sentences to logic form," in *Proceedings of the ACL*, 2009, pp. 976–984.

[14] T. Kwiatkowski, E. Choi, Y. Artzi, and L. Zettlemoyer, "Scaling semantic parsers with on-th-fly ontology matching," in *Proceedings of the EMNLP*, 2013, pp. 1545–1556.

[15] F. Li and H. V. Jagadish, "Nalir: An interactive natural language interface for querying relational databases," in *Proceedings of the SIGMOD*, 2014, pp. 709–712.

[16] Y. Su, H. Sun, B. Sadler, M. Srivatsa, I. Gür, Z. Yan, and X. Yan, "On generating characteristic-rich question sets for qa evaluation," in *Proceedings of the EMNLP*, 2016, pp. 562–572.

[17] J. Cheng, S. Reddy, V. Saraswat, and L. Mirella, "Learning an executable neural semantic parser," vol. 45, no. 1, pp. 59–94, 2017.

[18] C. Liang, J. Berant, Q. Le, K. D. Forbus, and N. Lao, "Neural symbolic machines: learning semantic parsers on freebase with weak supervision," in *Proceedings of the ACL*, 2017, pp. 23–33.

[19] J. Berant, A. Chou, R. Frostig, and P. Liang, "Semantic parsing on Freebase from question-answer pairs," in *Proceedings of the EMNLP*, 2013, pp. 1533–1544.

[20] J. Berant and P. Liang, "Semantic parsing via paraphrasing," in *Proceedings of the ACL*, 2014, pp. 1415–1425.

[21] C. Unger, L. Buhmann, J. Lehmann, A.-C. N. Ngomo, D. Gerber, and P. Cimiano, "Template-based question answering over rdf data," in *Proceedings of the WWW*, 2012, pp. 639–648.

[22] H. Bast and E. Haussmann, "More accurate question answering on freebase," in *Proceedings of the CIKM*, 2015, pp. 1431–1440.

[23] L. Zou, R. Huang, H. Wang, J. X. Yu, W. He, and D. Zhao, "Natural language question answering over rdf: A graph data driven approach," in *Proceedings of the SIGMOD*, 2014, pp. 313–324.

[24] L. Dong and M. Lapata, "Language to logical form with neural attention," in *Proceedings of the ACL*, 2016, pp. 33–43.

[25] M. Yu, W. Yin, K. S. Hasan, C. d. Santos, B. Xiang, and B. Zhou, "Improved neural relation detection for knowledge base question answering," in *Proceedings of the ACL*, 2017, pp. 571–581.

[26] Z. Xu, H. Zheng, Z. Fu, and W. Wang, "Enhancing question understanding and representation for knowledge base relation detection," in *Proceedings of the ICDM*, 2017, pp. 571–581.

[27] A. Bordes, J. Weston, and N. Usnier, "Open question answering with weak supervised embedding models," in *Proceedings of the PKDD*, 2014, pp. 165–180.

[28] M.-C. Yang, N. Duan, M. Zhou, and H. Rim, "Joint relational embeddings for knowledge-based question answering," in *Proceedings of the EMNLP*, 2014, pp. 645–650.

[29] J. Welbl, P. Stenetorp, and S. Riedel, "Constructing datasets for multi-hop reading comprehension across documents," *TACL*, vol. 6, pp. 287–302, 2018.

[30] Z. Yang, P. Qi, S. Zhang, Y. Bengio, W. Cohen, R. Salakhutdinov, and C. D. Manning, "HotpotQA: A dataset for diverse, explainable multi-hop question answering," in *Proceedings of the EMNLP*, 2018, pp. 2369–2380.

[31] N. D. Cao, W. Aziz, and I. Titov, "Question answering by reasoning across documents with graph convolutional networks," in *Proceedings of the NAACL*, 2014, pp. 2306–2317.

[32] M. Ding, C. Zhou, Q. Chen, H. Yang, and J. Tang, "Cognitive graph for multi-hop reading comprehension at scale," in *Proceedings of the ACL*, 2019.

[33] J. Weston, S. Chopra, and A. Bordes, "Memory networks," in *Proceedings of the ICLR*, 2015.

[34] A. H. Miller, A. Fisch, J. Dodge, and A.-H. Karimi, "Key-value memory networks for directly reading documents," in *Proceedings of the EMNLP*, 2016, pp. 1400–1409.

[35] Z.-Y. Chen, C.-H. Chang, Y.-P. Chen, J. Nayak, and L.-W. Ku, "An unrestricted-hop relation extraction framework for knowledge-based question answering," in *Proceedings of the NAACL*, 2019, pp. 345–356.

[36] S. Minjoon, K. Aniruddaha, F. Ali, and H. Hananneh, "Bi-directional attention flow for machine comprehension," in *Proceedings of the ICLR*, 2016.

[37] Z. Tu, Z. Lu, Y. Liu, X. Liu, and H. Li, "Modeling coverage for neural machine translation," in *Proceedings of the ACL*, 2016, pp. 76–85.

[38] A. See, P. J. Liu, and C. D. Manning, "Get to the point: summarization with pointer-generator networks," in *Proceedings of the ACL*, 2017, pp. 1073–1083.

[39] L. Zhang, J. Winn, and T. Ryota, "Gaussian attention model and its application to knowledge base embedding and question answering," *arXiv:1611.02266*, 2016.

[40] J. Duchi, E. Hazan, and Y. Singer, "Adaptive subgradient methods for online learning and stochastic optimization," vol. 12, pp. 2121–2159, 2011.

[41] J. Pennington, R. Socher, and C. D. Manning, "Glove: global vectors for word representation," in *Proceedings of the EMNLP*, 2014, pp. 1532–1543.