1-2019

Knowledge base question answering with a matching-aggregation model and question-specific contextual relations

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

Knowledge Base Question Answering with a Matching-Aggregation Model and Question-Specific Contextual Relations

Yunshi Lan, Shuohang Wang, and Jing Jiang

***Abstract*—Making use of knowledge bases (KBs) to answer questions (KBQA) is a key direction in question answering systems. Researchers have developed a diverse range of methods to address this problem, but there are still some limitations with existing methods. Specifically, existing neural network-based methods for KBQA have not taken advantage of the recent "matching-aggregation" framework for sequence matching, and when representing a candidate answer entity, they may not choose the most useful context of the candidate for matching. In this paper, we explore the use of a "matching-aggregation" framework to match candidate answers with questions. We further make use of question-specific contextual relations to enhance the representations of candidate answer entities. Our complete method is able to achieve state-of-the-art performance on two benchmark datasets: WebQuestions and SimpleQuestions.**

***Index Terms*—Artificial intelligence, natural language processing, knowledge base question answering.**

I. Introduction

With the development of large-scale knowledge bases such as Freebase [1], DBpedia [2] and YAGO [3], Knowledge Base Question Answering (KBQA) has become an important task and gained much attention in recent years [4]–[7]. KBQA aims to automatically find answers to factoid questions from a Knowledge Base (KB), where answers are usually entities in the KB. Figure 1 shows a small subset of a KB and an example question that can be answered from the KB.

Early work on KBQA often uses semantic parsing to transform a question into a KB-specific structure that can be used to directly query or match the KB [8]–[12]. However, this approach depends heavily on a suitable and accurate semantic parser, which may not be easy to build. More recently, a number of neural network-based methods have been proposed for KBQA and achieved good results on benchmark datasets [6], [7], [13]. Typically, these methods start by identifying entities mentioned in the question, which are referred to as *topic entities*. From these topic entities and by following the relation paths in the knowledge graph, candidate answer entities can be located, which are usually one or two-hops away from a topic entity. Neural network models are then used to encode both questions and candidate answers as vectors, which are then matched against each other for candidate ranking. Differences between the various models proposed lie in the types of KB information they use to represent candidate answers as well as the way they encode and match questions and candidate answers.

All three authors are with the School of Information Systems, Singapore Management University, 80 Stamford Road, Singapore 178902. Yunshi Lan is the corresponding author. E-mail: yslan.2015@phdis.smu.edu.sg, shwang.2014@phdis.smu.edu.sg and jingjiang@smu.edu.sg.

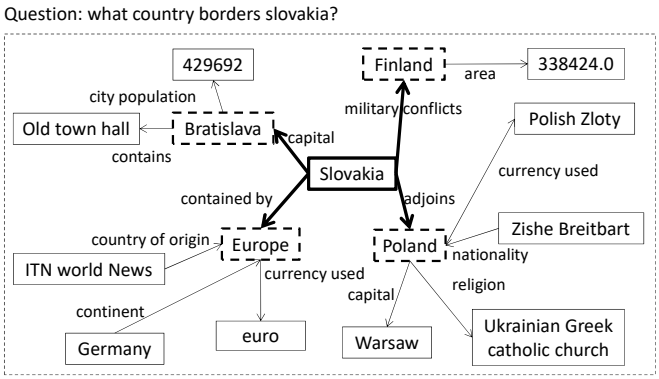


Figure 1: An example question and a subgraph of a KB. The correct answer to the question is *Poland*. The entity shown in the thick solid rectangle is a *topic entity*, and the four entities shown in dashed rectangles are *candidate answer entities*.

Although existing neural network-based methods have explored various candidate answer representations and matching models, there are at least two limitations of existing work. The first is about the matching model. KBQA can be regarded as a sequence matching problem where one sequence is the question and the other sequence is the relation path in the KB linking a topic entity to a candidate answer entity. For natural language sequence matching, several previous studies have shown that a "matching-aggregation" framework is preferred [14]–[16], in which two sequences are matched at word-level and the matching results are aggregated for a final decision. However, to the best of our knowledge, most existing matching models for KBQA are not based on this "matching-aggregation" framework, and thus may not achieve optimal matching results between questions and candidate answers.

Second, to represent candidate answers, existing methods usually consider only the entities and relations along the path from a topic entity to a candidate answer entity. However, other relations linked to a candidate answer may also contain very useful information about the candidate and should be considered. Take the example shown in Figure 1. We can see that the relations "capital" and "nationality" linked to the candidate "Poland" strongly indicate that this candidate is a country. If this information is considered during matching, we can increase the chance of this candidate being ranked high.

In this paper, we address the two limitations above by proposing two ideas for KBQA. First, we apply a "matching-aggregation" model to measure the similarity between a question and a candidate answer, which allows us to exploit word-level interactions through bi-directional attention mechanisms. Second, we incorporate question-specific contextual relations connected to a candidate to enhance its representation, where we use attention mechanism to weigh the relations based on their relevance to the question. We evaluate our proposed method on two data sets: WebQuestions [9] and SimpleQuestions [17]. The empirical results verified our hypotheses that a "matching-aggregation" framework works better for finding correct answers in KBQA, and the question-specific contextual relations incorporated into the candidate representations can further improve KBQA performance. We also find that our overall method can outperform the reported state-of-the-art performance on both WebQuesions and SimpleQuestions datasets by 4.2 percentage points and 0.7 percentage points, respectively.[1]

The contributions of our work are twofold: 1) We demonstrate that a "matching-aggregation" framework for sequence matching works significantly better than a standard sequence matching model for KBQA. 2) We propose to use question-specific contextual relations connected to candidate answers to help candidate ranking, which can also significantly improve KBQA performance.

II. Related Work

A. Semantic Parsing Methods for KBQA

The idea of semantic parsing-based methods is to transform a natural language question into a specific query language form so that we can use it to directly retrieve answers from the given knowledge base.

Early work along this line predefined some templates of queries. Entities and relations detected in the questions were then used to populate the query templates so that such filled templates could be executed via a programmer and predicted answers were returned [8], [18]–[20]. But predefined templates are difficult to capture the diverse expressions of natural language questions. Cai *et al.* [10], Kwiatkowski *et al.* [21] and Berant *et al.* [9] proposed several methods to transform natural language questions into general logic forms via semantic parsing and adapted to KB-specific forms by extending the lexicon. More recent work directly mapped questions to KB-specific forms, which could be generalized to large-scale datasets [11], [22], [23]. With the rapid development of reinforcement learning, Yih *et al.* [24] and Liang *et al.* [12] attempted to leverage reinforcement learning techniques to construct queries for knowledge bases by extending the queries step by step. These methods have gained promising results on several datasets. However, this category of methods mostly faces the challenge of building accurate and domain-specific parsers.

[1]For fair comparison, we do not consider models using external resources such as Wikipedia.

B. Neural Network Methods for KBQA

Neural network-based sequence matching methods treat the KBQA problem as matching two sequences using low-dimensional dense vectors as representations and neural networks for matching. Specifically, they match questions with candidate answers in a sequential manner.

Many studies represented a question and a candidate answer as two low-dimensional vectors and computed their similarity using dot product, but these methods ignored word order information and importance of different words [4], [13], [17], [25], [26]. To better capture features of candidate answers, recent work developed more expressive models [27], [28]. Yu *et al.* [6] represented questions and relations of candidate answers via different levels of abstraction and sorted relations via a neural network in their staged framework. Their method makes full use of the questions and relations but overlooks other important information in the KB. Hao *et al.* [7] collected multiple aspects of the information about a candidate answer as the representation of the candidate. Then a cross-attention mechanism was proposed to match it with the question. In addition, they used the TransE method to train representations for candidate answers using global KB knowledge. Their work is the most similar to ours, but there are still some differences: Their candidate answers are not represented as sequences whereas we use sequence representation for candidates. They use TransE to incorporate additional knowledge about candidates, but TransE requires the entire KB to train. In contrast, we only need to use the local contexts of candidates in the form of question-specific relations linked to the candidates.

There has been some work incorporating rules or external resources such as the ClueWeb corpus [29], the PPDB paraphrase dataset [30] and Wikipedia to help boost the performance of KBQA [11], [13], [24], [31]–[33]. In our work, we do not make use of any external knowledge. Therefore, we do not compare with such methods.

C. "Matching-aggregation" Framework for Sequence Matching

The "matching-aggregation" framework is a paradigm of neural network-based sequence matching models that *matches* vectorized representations of tokens in two sequences and then *aggregates* the matching results to arrive at a final matching score. Compared with the early approach that compresses each sequence into a single vector first [34], [35], the "matching-aggregation" framework first aligns words between the two sequences and then aggregates the alignment results to form a single vector to measure sequence-level similarity. This "matching-aggregation" framework has the advantage of fully capturing substructure interactions. It is widely recognized that the "matching-aggregation" framework performs well on diverse tasks. Several papers have shown the promising results on natural language inference [14]–[16], [36].

III. Method

A. Task Definition and Setup

We assume that there is a KB from which questions are to be answered. The KB contains a set of entities $\mathcal{E}$, where

each entity $e \in \mathcal{E}$ has a textual representation, which is a sequence of words. For example, $(isthmus, of, panama)$ is the textual representation of the entity *Isthmus of Panama*. The KB also has a set of relations defined, denoted by $\mathcal{R}$, where each relation $r \in \mathcal{R}$ also has a textual representation in the form of a word sequence (e.g., $(contained, by)$). We assume that all relations in the KB are directed, binary relations.[2] Facts in this KB are represented as triplets. For example, (e, r, e') indicates that the relation $r \in \mathcal{R}$ holds between the entities $e \in \mathcal{E}$ and $e' \in \mathcal{E}$, where e is called the *head entity* and e' the *tail entity*. The KB can also be represented as a graph, as we can see in Figure 1.

A question is represented as a sequence of words $Q = (q_1, q_2, \ldots, q_m)$ (e.g., $(what, country, borders, slovakia)$). Our goal is to return an entity from $\mathcal{E}$ as the answer to a given question. We assume that there is a set of question-answer pairs used as training data.

B. Method Overview

Before presenting the details of our method, we first give an overview. Similar to most previous work, our method begins by identifying *topic entities*, which are entities mentioned in a given question. We then use these topic entities to identify *candidate answer entities* (or *candidates* for short). In this work, candidates are those entities that are either directly linked to a topic entity through a single relation or two-hop away from a topic entity through two relations in the KB. In the example shown in Figure 1, *Slovakia* is a topic entity, and four candidates can be derived from the topic entity if we consider only one hop of relation: *Europe*, *Bratislava*, *Finland* and *Poland*. For each candidate, we use the entities and relations along the path from the original topic entity to the candidate to construct a sequence, which we refer to as a *candidate sequence*. Details of candidate sequences will be presented in Section III-C and Section III-D. Finally, using a neural network-based sequence matching model, we match all candidate sequences with the question sequence in order to rank the candidates and select the top one as the answer.

Although the overall framework of our method is similar to previous work, we propose two novel ideas to address the limitations we pointed out earlier in order to improve KBQA performance. (1) To match the question sequence with a candidate sequence, we adopt a "matching-aggregation" framework, which has been shown to be more effective than a "Siamese" matching model for various NLP tasks [14]–[16], [36]. Note that although this "matching-aggregation" framework is not new, to the best of our knowledge, it has not been applied to KBQA. (2) Based on our observation that relations connected to the candidates are potentially useful, we include them in the candidate representations. We also note that these relations are not equally important and therefore we propose to use an attention mechanism to carefully weigh these relations in order to optimize their effect.

We now present our method in detail.

[2]For n-ary relations, we convert them to binary relations following the practice in [26].

C. Base Candidate Sequences

In order to identify a set of entities from the KB as candidate answers, we first identify a set of *topic entities* from the given question. For example, given the question "*where is isthmus of panama located*," we would identify *Isthmus of Panama* and *Panama* as topic entities. Note that this step follows the practice of several previous studies [6], [7], [17], [18], [24], [32]. We use external tools to identify the topic entities. Let us use $\mathcal{E}^t_Q \subset \mathcal{E}$ to denote the set of topic entities found in question Q.

Next, for each topic entity $e^t \in \mathcal{E}^t_Q$, by following its connections in the KB, we can identify all the entities that are either one-hop or two-hop away from e^t. We combine all these entities that are one or two-hop away from any topic entity and consider them to be our candidate answer entities. Let us use $\mathcal{E}^c_Q \subset \mathcal{E}$ to refer to this candidate set.

We first introduce our *base candidate sequences*. To construct the base candidate sequence for a candidate $e^c \in \mathcal{E}^c_Q$, we use the entities and relations along the path connecting this candidate to the corresponding topic entity. Recall that each entity or relation has a textual representation in the KB. We concatenate the word sequences representing the entities and relations along the path to form the base candidate sequence. Note that we exclude the candidate entity itself in the base candidate sequence representation. This is because eventually we will match the candidate sequence with the question sequence, but we do not expect the candidate itself to appear in the question. For example, the candidate *Poland* does not appear in the question *"what country borders slovakia"* although this candidate is the correct answer.[3]

To illustrate base candidate sequences, we show four candidates and their corresponding base candidate sequences in Table I for the question "*what country borders slovakia*" that corresponds to the example shown in Figure 1. Note that X^{-1} represents the inverse relation of X.

D. Enhanced Candidate Sequences

To enhance the base candidate sequence for a candidate e^c, we further look for other relations that are linked to e^c, where e^c could be either a head entity or a tail entity. Let $\mathcal{R}_{e^c} \subset \mathcal{R}$ denote the set of relations connected to e^c, excluding those that link e^c to a topic entity. This set of relations $\mathcal{R}_{e^c}$ provides some *contextual information* about the candidate entity and thus can be potentially useful. For example, for the candidate *Poland*, the relation *nationality* connected to it could potentially help better match this candidate with the question, which asks for a country. We refer to this set $\mathcal{R}_{e^c}$ as the *contextual relations* of candidate e^c. See Table I for these contextual relations of the four candidates in our example. In Section III-E we will explain how we use attention mechanism to further give those question-specific contextual relations higher weights.

[3]We have tried to incorporate candidate answer entity itself into the candidate sequence. However, based on our preliminary experiments, we find that indeed such additional information does not improve the performance.

TABLE I: EXAMPLES OF CANDIDATES AND THEIR CANDIDATE SEQUENCES.

Candidate	Base candidate sequence	Contextual relations
Europe	(*slovakia*, *contained*, *by*)	{*#continent*$^{-1}$, *#currency_used*, *#country_of_origin*$^{-1}$}
Poland	(*slovakia*, *adjoins*),	{*#nationality*$^{-1}$, *#religion*, *#capital*, *#currency_used*}
Bratislava	(*slovakia*, *capital*),	{*#contains*, *#city_population*}
Finland	(*slovakia*, *military*, *conflicts*),	{*#area*}

E. Sequence Matching

We are now ready to use a sequence matching model to measure how close a candidate sequence is to a question sequence. Neural network-based sequence matching has been well studied in many NLP problems such as natural language inference and machine comprehension. Early work on neural network-based sequence matching first transforms each sequence into a single vector representation using models such as CNN and LSTM, and then the two vectors representing the two sequences are combined either through dot product or another neural network layer to give a matching score [34], [35], [37]. A limitation with this approach is that it does not consider token-level alignment between the two sequences.. Later, a number of models following a general "matching-aggregation" framework were proposed for different tasks and achieved better performance [14]–[16], [36]. In these models, tokens represented as vectors from the two sequences are first matched, where the matching results are represented as vectors. Then these "matching vectors" are further aggregated. In this paper, we believe that this "matching-aggregation" framework may also work well for KBQA.

First of all, for the question sequence and the base candidate sequence, we associate each word with a word embedding vector (which will be initialized using existing word embeddings but updated during training). Then for the set of contextual relations in the enhanced candidate sequence, we associate each contextual relation with a relation embedding vector (which will be randomly initialized and updated during training).

Let $\mathbf{Q} = (\mathbf{q}_1, \mathbf{q}_2, \ldots, \mathbf{q}_m)$ denote the sequence of word embeddings of the question. Let $\mathbf{C} = (\mathbf{c}_1, \mathbf{c}_2, \ldots, \mathbf{c}_n)$ denote the sequence of embedding vectors of the enhanced candidate sequence for candidate e^c. Here $(\mathbf{c}_1, \mathbf{c}_2, \ldots, \mathbf{c}_{n-1})$ are the word embeddings of the words in the base candidate sequence for e^c, and the last embedding $\mathbf{c}_n$ is defined as a combination of the relation embeddings of the relations inside $\mathcal{R}_{e^c}$, i.e., the contextual relations linked to e^c. We will explain how $\mathbf{c}_n$ is derived from the relations in $\mathcal{R}_{e^c}$ later.

Given the two sequences $\mathbf{Q}$ and $\mathbf{C}$, we try to derive a matching score between them.

First, we try to match $\mathbf{q}_i$ with $\mathbf{c}_j$ as follows:

$$e_{ij} \quad = \quad F(\mathbf{q}_i)^T F(\mathbf{c}_j),$$

where $F(\cdot)$ is a single non-linear layer with ReLU as its activation function. e_{ij} essentially encodes the degree of matching between $\mathbf{q}_i$ and $\mathbf{c}_j$, the same as what was used in [16].

We then use e_{ij} defined above to derive the following normalized attention weights:

$$a_{ij} \quad = \quad \frac{\exp(e_{ij})}{\sum_{i'=1}^{m} \exp(e_{i'j})}.$$

Here a_{ij} represents the importance of matching $\mathbf{q}_i$ to $\mathbf{c}_j$, compared with other tokens $\mathbf{q}_{i'}$ in the question. Similarly, we also compute another set of attention weights in the other direction:

$$b_{ij} \quad = \quad \frac{\exp(e_{ij})}{\sum_{j'=1}^{n} \exp(e_{ij'})}.$$

Here b_{ij} represents the importance of matching $\mathbf{c}_j$ to $\mathbf{q}_i$, compared with other tokens $\mathbf{c}_{j'}$ in the candidate sequence.

We then define the following weighted versions of $\mathbf{q}$ and $\mathbf{c}$:

$$\tilde{\mathbf{q}}_j \quad = \quad \sum_{i=1}^{m} a_{ij} \cdot \mathbf{q}_i,$$
$$\tilde{\mathbf{c}}_i \quad = \quad \sum_{j=1}^{n} b_{ij} \cdot \mathbf{c}_j.$$

We can see that here $\tilde{\mathbf{q}}_j$ is a weighted sum of all the $\mathbf{q}_i$ in the question sequence in order to match $\mathbf{c}_j$ in the candidate sequence. It follows the standard attention mechanism in most previous work. The same idea applies to $\tilde{\mathbf{c}}_i$.

Next, we match $\mathbf{q}_i$ with $\tilde{\mathbf{c}}_i$ and $\mathbf{c}_j$ with $\tilde{\mathbf{q}}_j$ by defining the following two vectors:

$$\mathbf{v}_{1,i} \quad = \quad G\left(\begin{bmatrix} \mathbf{q}_i \odot \tilde{\mathbf{c}}_i \\ (\mathbf{q}_i - \tilde{\mathbf{c}}_i) \odot (\mathbf{q}_i - \tilde{\mathbf{c}}_i) \end{bmatrix}\right),$$
$$\mathbf{v}_{2,j} \quad = \quad G\left(\begin{bmatrix} \mathbf{c}_j \odot \tilde{\mathbf{q}}_j \\ (\mathbf{c}_j - \tilde{\mathbf{q}}_j) \odot (\mathbf{c}_j - \tilde{\mathbf{q}}_j) \end{bmatrix}\right),$$

where $\odot$ denotes element-wise multiplication and $G(\cdot)$ is another feed-forward neural network with ReLU activation. $\mathbf{v}_{1,i}$ measures the similarity between the question and its weighted version at the i-th position. The same principle applies to $\mathbf{v}_{2,j}$. Note that our design of these two vectors are inspired by [14].

Next, we aggregate the sequences of $\mathbf{v}_{1,i}$ and of $\mathbf{v}_{2,j}$ using LSTM [38] and then extract two values from the resulting vectors through max pooling:

$$\tilde{\mathbf{V}}_1 = \text{LSTM}([\mathbf{v}_{1,1}, \mathbf{v}_{1,2}, \ldots, \mathbf{v}_{1,m}]), \tilde{\mathbf{v}}_1 = \max_i \tilde{\mathbf{V}}_{1,i},$$
$$\tilde{\mathbf{V}}_2 = \text{LSTM}([\mathbf{v}_{2,1}, \mathbf{v}_{2,2}, \ldots, \mathbf{v}_{2,n}]), \tilde{\mathbf{v}}_2 = \max_j \tilde{\mathbf{V}}_{2,j}.$$

Note that other work has used other ways of aggregation such as summation and CNN. We chose LSTM because it worked well in [15].

Finally, we concatenate and feed $\tilde{\mathbf{v}}_1$ and $\tilde{\mathbf{v}}_2$ to H, which is a feed forward network followed by a linear layer. It gives us the matching score between question sequence $\mathbf{Q}$ and candidate sequence $\mathbf{C}$:

$$s(\mathbf{Q}, \mathbf{C}) \quad = \quad H([\tilde{\mathbf{v}}_1; \tilde{\mathbf{v}}_2]).$$

Using a softmax layer over the matching scores of all candidates, we can then derive a distribution over the candidates.

During the training stage, we use the KL divergence between the true distribution and predicted distribution as the objective function to learn the various model parameters. For prediction, we select the candidate with the highest probability as the predicted answer.

F. Combining Contextual Relations with Attention

We now describe how $\mathbf{c}_n$ is derived from the contextual relations $\mathcal{R}_{e^c}$ for candidate e^c.

A naive way is to take the average of all the relation embeddings, which we refer to as **Avg**:

$$\mathbf{c}_n = \frac{1}{|\mathcal{R}_{e^c}|} \sum_{r \in \mathcal{R}_{e^c}} \mathbf{r},$$

where $\mathbf{r}$ is the embedding vector of relation r.

However, this naive method has its weakness because not all contextual relations are equally relevant to the question. To better capture the relevant contextual relations, we use attentions to weigh the different contextual relations such that the question-specific contextual relations can be weighted higher.

We first encode the question sequence $\mathbf{Q}$ into a single vector $\bar{\mathbf{q}}$ by taking the average.

$$\bar{\mathbf{q}} = \frac{1}{m} \sum_{i=1}^{m} \mathbf{q}_i.$$

After that we apply an attention network to decide which relation is important for the question. We define β_r as follows:

$$\beta_r = \frac{\exp(\mathbf{w}^T[\mathbf{r}; \bar{\mathbf{q}}] + b)}{\sum_{r' \in \mathcal{R}_{e^c}} \exp(\mathbf{w}^T[\mathbf{r}'; \bar{\mathbf{q}}] + b)}, \quad (1)$$

where $\mathbf{w}$ and b_r are parameters to be learned, and $\mathbf{r}$ is the embedding for relation r. β_r essentially represents the importance of relation r with respect to the question. Then $\mathbf{c}_n$ is obtained as follows:

$$\mathbf{c}_n = \sum_{r \in \mathcal{R}_{e^c}} \beta_r \mathbf{r}.$$

We refer to this combination method as **RelAtt**.

Still, for a question $\mathbf{Q}$, oftentimes only some aspects of the question are more important, such as "country" in the example question we have seen. So we also explore a self-attention mechanism on the question to decide which parts of the question should be highlighted to match a candidate. We first use LSTM to transform a question sequence into a single vector:

$$\tilde{\mathbf{Q}} = \text{LSTM}([\mathbf{q}_1, \mathbf{q}_2, \ldots, \mathbf{q}_m]), \qquad \tilde{\mathbf{q}} = \tilde{\mathbf{Q}}_m.$$

Next, we define

$$\alpha_i = \frac{\exp(\mathbf{u}^T[\tilde{\mathbf{q}}; \mathbf{q}_i] + d)}{\sum_{i'=1}^{m} \exp(\mathbf{u}^T[\tilde{\mathbf{q}}; \mathbf{q}_{i'}] + d)},$$

where $\mathbf{u}$ and d are parameters to be learned. This α_i represents how important the token $\mathbf{q}_i$ is inside the question. This is the *self-attention* mechanism that has been widely used [39].

Then we define

$$\bar{\mathbf{q}}' = \sum_{i=1}^{m} \alpha_i \mathbf{q}_i.$$

This $\bar{\mathbf{q}}'$ captures the more important aspects of the question. Then we can use $\bar{\mathbf{q}}'$ instead of $\bar{\mathbf{q}}$ in Equation (1) to obtain $\mathbf{c}_n$. We denote this method as **SelfAtt**.

G. Constraint Detection

It is worth noting that some questions may give strict constraints of the answer type. For example, for the question *"what state is Harvard College located"*, the candidate answers include *Cambridge*, *United States of America* and *Massachusetts*. If we directly have the entity type information of these candidates, we can easily find that *Massachusetts* is the best answer. Although our method using contextual relations could possibly also encode such knowledge, we expect that using explicit entity type or entity description would possibly be supplementary because they can encode more fine-grained entity type information. Thus, we include a post-processing step with the following heuristic. If we find some exact word match between the question and either a candidate entity's textual description or the entity type of the candidate, we adjust the probability of the candidate by the following formula:

$$p(e^c)' = \gamma + (1 - \gamma) \times p(e^c), \quad (2)$$

where $p(e^c)$ is the probability for candidate e^c as computed by the sequence matching model, and γ is a hyper-parameter manually set.

Note that we detect and apply such constraints in all versions of our model that are being compared in Section IV.

IV. Experiments

A. Setup

We evaluate our proposed method on two commonly used benchmark datasets: WebQuestions and SimpleQuestions.

WebQuestions (WQ)[4]: This dataset was introduced by Berant *et al.* [9]. The dataset contains 5,810 question-answer pairs with 3,778 training pairs and 2,032 test pairs. We randomly split the training data into 3,000 training pairs and 778 development pairs. In order to obtain topic entities, we use the entity linking output generated by YodaQA[5]. The KB we use is the latest dump of Freebase[6] and we process it the same way as [26].

Because multiple correct answers for each question are annotated as the ground truth, our method also returns multiple answers based on a tuned threshold. We evaluate our method using the official evaluation script provided by Berant *et al.* [9]. The standard evaluation metric is average F1 over all test questions.

SimpleQuestions (SQ)[7]: The SimpleQuestions dataset was introduced by Bordes *et al.* [17]. It contains 108,442 question-answer pairs, with 75,910, 10,845 and 21,687 pairs for training, development and testing, respectively. In order to make fair comparison with previous work, we use FB2M as our KB,

[4]https://nlp.stanford.edu/software/sempre/
[5]https://github.com/brmson/dataset-factoid-webquestions
[6]https://developers.google.com/freebase/
[7]https://research.fb.com/downloads/babi/

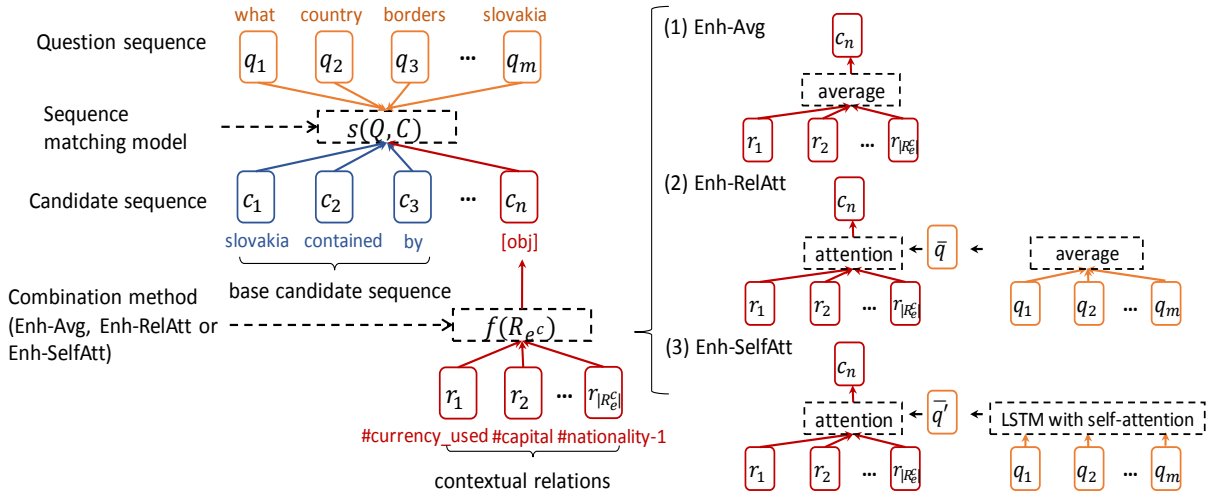


Figure 2: An illustration of our model.

which is a subset of Freebase that consists of 2M entities and 6K relations. For topic entities, we start from the entity linking results from [6][8]. For this dataset, the standard evaluation metric is accuracy, which means we count one prediction as correct if our topic entity and relation match the ground truth.

Through the empirical evaluation we aim to test (1) whether the "matching-aggregation" model works better than a standard sequence matching model for KBQA, and (2) whether our enhanced candidate sequences with the contextual relations are better than the base candidate sequences. Therefore, we compare the following methods:

Previous Work: We list the performance of previous work built on end-to-end neural networks [6], [7], [13], [17], [25], [27], [28] or semantic parsers [40]–[43] on WebQestions or SimpleQuestions. We use † to indicate our re-implemented versions of previous models.

Baseline: This is a baseline method implemented by ourselves, where we use the base candidate sequences and a standard sequence matching model that does not follow the "matching-aggregation" framework. Specifically, we use a BiLSTM model [44] to process both the question sequence and the candidate sequences first. We then use max pooling to combine all the hidden states of a question (or a candidate sequence) into a single vector. These vectors are then used to compute cosine similarities between all candidate sequences and questions to rank the candidates.

Match-Aggr: This is our method that uses the base candidate sequences together with the "matching-aggregation" framework for sequence matching, as presented in Section III-E.

Enh-Avg: This is our method using the enhanced candidate sequence with the Avg method to combine the contextual relations.

Enh-RelAtt: This is our method using the enhanced candidate sequence with the RelAtt method to combine the contextual relations.

Enh-SelfAtt: This is our method using the enhanced candidate sequence with the SelfAtt method to combine the contextual relations.

For the sake of completeness, we also list the best reported performance on these datasets. However, it is important to note that the best performing systems [27], [33], [45] make use of external resources such as Wikipedia and WikiAnswers. Because we do not use such external resources, it is not fair to compare our results with these state-of-the-art results.

B. Implementation Details

For both datasets, we initialize our word vectors with 300-dimensional pre-trained word embeddings [46][9]. Adagrad algorithm [47] is employed to optimize our objective function. We tune the hyper-parameters on the development data in the following way: (1) The size of hidden states is chosen from $\{50, 100, 150, 200\}$. (2) Dropout ratio is chosen from $\{0, 0.1, 0.2, 0.3, 0.4\}$. (3) The hyper-parameter γ for post-processing is chosen from $\{0.1, 0.2, 0.3, 0.4, 0.5\}$.

It is possible that sometimes there may be many candidates sharing the same base candidate sequence. This is because there are often one-to-many relations in a KB. To reduce the computational costs, instead of treating these as different candidate sequences, we merge these candidates as well as their contextual relations and construct a single candidate sequence for them. We use the heuristic explained above to further rank them.

C. Results

Our results are shown in Table II. The top section contains previously reported performance on the two datasets. The middle section contains our results, where $*$ and $\star$ indicate that the result is statistically significantly better than Match-Aggr and Enh-Avg, respectively. The bottom section serves as a reference point to show the state of the art. However, the three studies in the bottom section used external resources such as Wikipedia whereas our method does not use any external resource.

As we can see from the table, the Match-Aggr method beats Our Baseline with a vast margin, and it can already reach

[8]https://github.com/Gorov/SimpleQuestions-EntityLinking

[9]https://github.com/stanfordnlp/GloVe

TABLE II: EXPERIMENT RESULTS.

Method	**WQ** Avg F1	**SQ** Avg acc
Bao *et al.* (2014) [40]	37.5	-
Xu *et al.* (2014) [41]	39.1	-
Bordes *et al.* (2014) [25]	39.2	-
Berant and Liang (2014) [42]	39.9	-
Yang *et al.* (2014) [43]	41.3	-
Dong *et al.* (2015) [13]	40.8	-
Bordes *et al.* (2015) [17]	42.2	63.9
Yin *et al.* (2016) [48]	-	76.4
Hao *et al.* (2017) [7]	42.9	-
Yu *et al.* (2017) [6]	-	78.7
Hao *et al.* (2018) [28]†	42.6	80.2
Baseline	39.5	74.1
Match-Aggr	42.9	79.2
Enh-Avg	43.8*	80.3*
Enh-RelAtt	45.2**	80.7**
Enh-SelfAtt	**47.1****	**80.9****
Yih *et al* (2015) [24]	52.5	-
Xu *et al.* (2016) [33]	53.3	-
Wang *et al.* (2018) [27]	63.4	81.5

the state-of-the-art performance among models not using any external knowledge on both datasets. Next, we can see that after we use enhanced candidate sequences with the contextual relations, even with the Avg combination method, the performance can be significantly improved. With the attention mechanisms, the performance can be further improved significantly, and specifically, SelfAtt performs better than RelAtt. Overall, with our complete method, we can improve the performance of KBQA by around 4 and 1 percentage points for WebQuestions and SimpleQuestions datasets respectively.

D. Further Analysis

In this section, we perform some further analysis to illustrate the effectiveness of our model.

TABLE III: TOP-K RESULTS ON WEBQUESTIONS AND SIMPLEQUESTIONS.

	WebQuestions	SimpleQuestions
Top-2	68.6	91.0
Top-3	78.0	92.4
Top-5	82.5	93.4
Top-10	87.2	94.5

1) Top-K Performance: We present the top-K accuracy, as defined below, in Table III. For both WebQuestions and SimpleQuestions, we retrieve the candidate answer entities that are linked to the top-K best candidate sequences. If any one of the ground truth is in the returned answers, we mark it as correct. Note this is a relaxed evaluation metric compared with our evaluation shown in Table II. As we can see, WebQuestions is harder to answer than SimpleQuestions, but most questions can be answered correctly by the top-10 answers.

2) Performance Breakdown: To see if our method works better for some types of questions and worse for others, we group the questions in two ways.

First, we group the questions based on answer types. This can be done by looking at the first word of a question, such as "when", "where." We show the performance of different types of questions on the WebQuestions dataset in Figure 3. We can see that "when" question is harder to answer than other

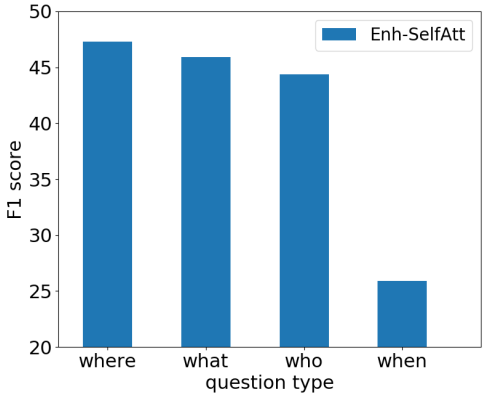


Figure 3: F1 scores over different question types on WQ.

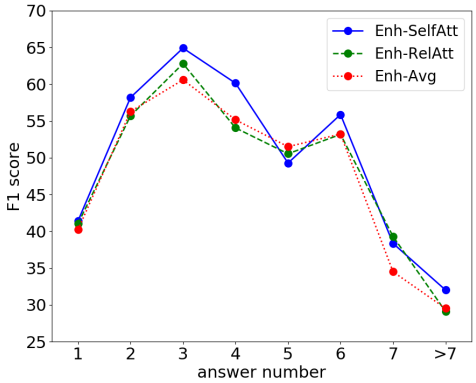


Figure 4: F1 scores over questions with different numbers of answers on WQ.

question types. For "when" questions, although we expect the answers to be a temporal type of entities, oftentimes the correct answers may not be typical temporal expressions. For example, one of the "when" questions in the WebQuestions dataset is *"when did Romney become governor."* The predicted answer is "2003-02-01", which is linked to the topic entity "Mitt Romney" through the relation *appointed_from*. The correct answer according to the ground truth, however, is "1/2/2003", which is essentially the same as "2003-02-01" but in a different format. Because the evaluation script does not consider the two different representations to be the same, the predicted answer is considered wrong although it is actually correct. We can see that because temporal expressions have different formats, the model could be penalized for returning an answer in a different format.

Because some questions have multiple answers, we also group the questions based on how many answers they have. We show the performance breakdown in terms of the numbers of answers on the WebQuestions dataset in Figure 4. We can see when the number of answers goes up, the performance drops. It is interesting to see that the best performance is achieved when the question has three answers. This is probably because when there are more than one answers, if we can capture one of them, we can already gain some points, but if a question's answer is unique, it is hard to rank the correct answer at the top.

TABLE IV: SAMPLING NUMBER AND F1 SCORE ON WEBQUESTIONS

Sample Size L	Percentage	Avg F1
50	92.8%	46.8
100	95.7%	46.9
150	97.0%	47.0
300	98.5%	47.1
no limit	100%	47.1

TABLE V: RELATION TYPE AND F1 SCORE ON WEBQUESTIONS

Relation type	Avg F1
No direction	44.4
Only left	45.9
Only right	46.5
Both	47.1
Only right (word level)	46.2

3) Effect of Relation Sampling: We find that a small percentage of candidate answer entities may have many contextual relations, sometimes up to thousands. If we always use all these contextual relations during training, the computational costs may be high. We therefore experiment with a sampling strategy where we sample up to L contextual relations for each candidate answer entity during training. Note that for prediction we still take all contextual relations into consideration. We report the performance on WebQuestions when L is chosen to be 50, 100, 150, 300. The middle column shows the percentage of answer entities whose total number of contextual relations are below L. We can see that if we sample too few (e.g., 50), the performance clearly drops, but the difference is not big. On the other hand, we can safely sample up to 300 contextual relations for each candidate answer entity without hurting the performance. We can see that if L is 300, more than 98% of the candidate answers still have all their contextual relations used.

4) Effect of Relation Direction: In our experiments, we care about the directions of those contextual relations connected to the candidate answers, and we use X and X^{-1} to denote relations with opposite directions. See Table I. To see whether it is necessary to do so, we conduct contextual experiments and show the results in Table V. "No direction" means we treat *institution*$^{-1}$ the same as *institution*. "Only left" means we only consider those relations where the candidate answer is a tail entity, like in *institution*$^{-1}$. "Only right" means we only consider those relations where the candidate answer is a head entity, like in *institution*. "Both" means we consider both left and right relations and treat them differently, which is the default strategy we use. "Only right (word level)" denotes that we only use right relations and represent them by words (by sum up the word embeddings).

We can see from the result that "Only left" performs worse than "Only Right", but both of them perform better than "No direction". Moreover, treating relations as sequences of words decreases the performance slightly. This may be because the sequences of words are not expressive enough to indicate the direction of relations.

E. Case Study

In this section, we focus on some cases and visualize some of the hidden variables to better understand our model.

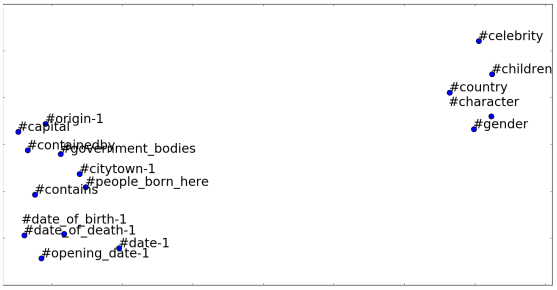


Figure 5: Learned relation embeddings in 2-D space.

1) Some Cases of Correct Predictions: Table VI demonstrates some cases which are wrongly predicted by Match-Aggr method but correctly predicted by Enh-SelfAtt. The two columns in the middel show the candidate sequence (excluding the candidate entity) ranked high by the corresponding method. The first example shows that if we do not make use of the contextual relations, the candidate sequence (*mediterranean*, *sea*, *contains*) would match the question well based on the Match-Aggr method, but actually this sequence does not lead to the correct answer. However, by considering the contextual relations, the Enh-SelfAtt method can find a better candidate sequence (*mediterranean*, *sea*, *islands*), which leads to the correct answer. Similarly, in the second example, the contextual relation "*county*$^{-1}$" helps to identify *Randolph County* (linked to *Coalton* through *containedby*) as a correct answer. For the third example, "California" in the sequence returned by Match-Aggr is a book. However, if we consider the context of the question, it should be an album. The contextual relations help detect this and correctly choose an answer which is related to album instead of book.

2) Learned Embeddings of Contextual Relations: To check whether the contextual relations can indeed encode useful knowledge about the candidates, we extract the learned relation embeddings of some of the contextual relations and map them to a 2-dimensional space. We show these relations in Figure 5. We can see that indeed relations that are close to each other tend to be associated with the same type of entities. For example, *country* and *gender* are close to each other in Figure 5, probably because both these two relations connect to entities which are people. We can also see that *opening_date*$^{-1}$ and *date_of_birth*$^{-1}$ are also close to each other, probably because they both connect to entities which are dates.

3) Attention Heatmap of Model: Figure 6 displays the attention weight e_{ij} introduced in Section III. As we can see, the word-level matching between the question and candidate sequences is well captured by the attention weights. "Birth" and "born", and "kateri" and "kateri" are connected with high attention values. When we look at the answer representation, it gains high attention on the question word "when", which indicates that the answer is expected to be a temporal expression. This shows that such question-specific contextual relations play an important role in answering this question.

4) Importance of Question-Specific Contextual Relations: In Figure 7, we show the values of β_r introduced in our model. From the figure, given the question "when was Blessed

TABLE VI: CASE STUDY

Examples	Match-Aggr	Enh-SelfAtt	Contextual Relations
What is an island located in the Mediterranean sea	(*mediterranean*, *sea*, *contains*)	(*mediterranean*, *sea*, *islands*)	{*#islands*$^{-1}$, *#island_group*}
What county is Coalton in	(*coalton*, *county*)	(*coalton*, *containedby*)	{*#countyplace_id*, *#county*$^{-1}$}
Who wrote the album California	(*california*, *written*, *by*)	(*california*, *artist*)	{*#album*, *#album_supporting*$^{-1}$}

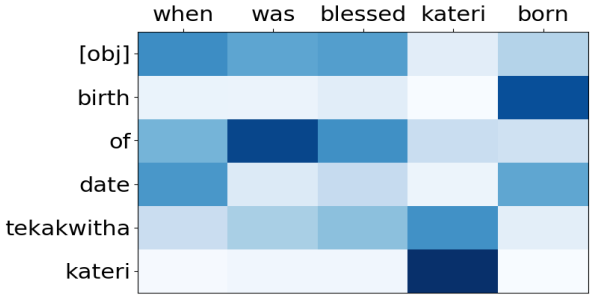


Figure 6: Heatmap of attention weights in model

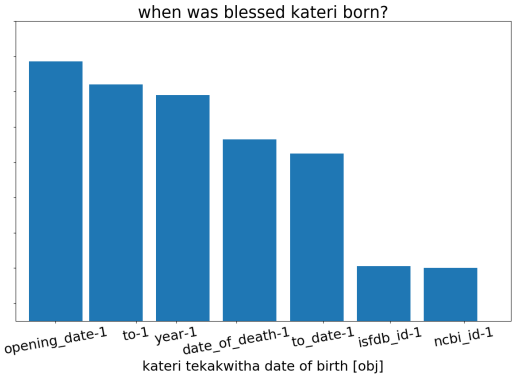


Figure 7: Visualization of importance of global relation information

Kateri born", the predicted answer *1656* is connected with many other relations such as "opening date" and "year". These relations are compressed into a single vector $[obj]$ to indicate the property of the answer. To see what kind of relations are treated as useful information for the question, we extract the top 5 important relations as well as the bottom 3 in the figure. The question-specific relations *opening_date*$^{-1}$, *to*$^{-1}$, *year*$^{-1}$ are most important, since they indicate that *1656* is a date while the other relation *ncbi_id*$^{-1}$ is a general relation connected to all entities in the knowledge base. Such meaningless relations are given lower weights, which verifies the necessity of relation weight in our model.

F. Error Analysis

We also conduct some error analysis. We sample 100 wrongly answered questions from the WebQuestions dataset randomly. We then examine them to identify the reasons for the mistakes. The following categories of errors are identified:

Ground truth incompletion (29%): There are many sampled questions whose ground truth answers are not complete. For example, for the question *"what team is Kris Humphries play for,"* besides the answer *Brooklyn Nets*, we find that there are other correct answers from the KB such as *Washington Wizards* and *Toronto Raptors*.

Question ambiguity (20%): This category contains questions which have ambiguous descriptions. As a result, multiple potential relations may match the questions correctly. For example, for the question *"who was Juan Ponce de Leon family,"* the predicted answer is *Barbara Ryan* through the *parents* relation, while the ground truth is *Elizabeth Ryan* through the *children* relation.

Complex questions (18%) : This type of errors occurs when some inference is needed to answer the question. For example, for the question *"who rules Denmark right now,"*, one needs to know the current time and compare it with the time associated with a relevant relation (e.g., *appointed_by*) in order to find the correct answer. There are also questions containing qualifiers such as *first*, *last time* and *after*, which make the questions harder to answer. For this type of questions, our method is not able to handle them.

V. CONCLUSIONS

In this paper, we proposed a sequence matching-based solution to KBQA. We constructed candidate sequences using entities and relations linking candidate answers to a question. Then an elaborately-designed "matching-aggregation" framework is leveraged to rank the candidate answers. Furthermore, we proposed to include informative relations connected to a candidate to further enhance its representation. Our experiment results showed that our method could outperform the current state of the art for two commonly used benchmark KBQA datasets. Further analysis on the KBQA questions verified the effectiveness of our model and significance of answer representation.

In the future, we plan to focus on answering complex questions, which may involve logic and reasoning.

We will release our data and code soon.[10]

ACKNOWLEDGEMENTS

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

REFERENCES

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

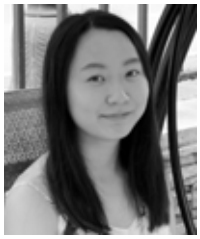
[2] S. Auer, C. Bizer, G. Kobilarov, J. Lehmann, R. Cyganiak, and Z. Ives, "Dbpedia: A nucleus for a web of open data," in *ISWC/ASWC*, 2007, pp. 722–735.

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

[4] A. Bordes, J. Weston, and N. Usunier, "Open question answering with weak supervised embedding models," in *ECML-PKDD*, 2014, pp. 165–180.

[10]https://github.com/lanyunshi/KBQA

[5] Y. Zhang, S. He, K. Liu, and J. Zhao, "A join model for question answering over multiple knowledge bases," in *AAAI*, 2016, pp. 3094–3100.
[6] 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 *ACL*, 2017, pp. 571–581.
[7] Y. Hao, Y. Zhang, K. Liu, S. He, Z. Liu, H. Wu, and J. Zhao, "An end-to-end model for question answering over knowledge base with cross-attention combining global knowledge," in *ACL*, 2017, pp. 221–231.
[8] C. Unger, L. Bühmann, J. Lehmann, A.-C. N. Ngomo, D. Gerber, and P. Cimiano, "Template-based question answering over RDF data," in *WWW*, 2012, pp. 639–648.
[9] J. Berant, A. Chou, R. Frostig, and P. Liang, "Semantic parsing on Freebase from question-answer pairs," in *EMNLP*, 2013, pp. 1533–1544.
[10] Q. Cai and A. Yates, "Large-scale semantic parsing via schema matching and lexicon extension," in *ACL*, 2013, pp. 423–433.
[11] W.-t. Yih, X. He, and C. Meek, "Semantic parsing for single-relation question answering," in *ACL*, 2014, pp. 643–648.
[12] C. Liang, J. Berant, Q. Le, K. D. Forbus, and N. Lao, "Neural symbolic machines: learning semantic parsers on freebase with weak supervision," in *ACL*, 2017, pp. 23–33.
[13] L. Dong, F. Wei, M. Zhou, and K. Xu, "Question answering over Freebase with multi-column convolutional neural networks," in *ACL*, 2015, pp. 260–269.
[14] S. Wang and J. Jiang, "A compare-aggregate model for matching text sequences," in *ICLR*, 2017.
[15] Z. Wang, W. Hamza, and R. Florian, "Bilateral multi-perspective matching for natural language sentences," in *IJCAI*, 2017, pp. 4144–4150.
[16] A. P. Parikh, O. Täckström, D. Das, and J. Uszkoreit, "A decomposable attention model for natural language inference," in *EMNLP*, 2016.
[17] A. Bordes, N. Usunier, S. Chopra, and J. Weston, "Large-scale simple question answering with memory networks," in *arXiv preprint*, 2015.
[18] W.-t. Yih, M. Richardson, C. Meek, M.-W. Chang, and J. Suh, "The value of semantic parse labeling for knowledge base question answering," in *ACL*, 2016, pp. 201–206.
[19] S. Yavuz, I. Gur, Y. Su, M. Srivatsa, and X. Yan, "Improving semantic parsing via answer type inference," in *EMNLP*, 2016, pp. 149–159.
[20] S. Hu, L. Zou, J. Yu, H. Wang, and D. Zhao, "Answering natural language questions by subgraph matching over knowledge graphs," *TKDE*, vol. 30, pp. 824–837, 2017.
[21] T. Kwiatkowski, E. Choi, Y. Artzi, and L. Zettlemoyer, "Scaling semantic parsers with on-th-fly ontology matching," in *EMNLP*, 2013, pp. 1545–1556.
[22] X. Yao and B. V. Durme, "Information extraction over structured data: question answering with freebase," in *ACL*, 2014.
[23] X. Yao, "Lean question answering over Freebase from scratch," in *NAACL*, 2015, pp. 66–70.
[24] W.-t. Yih, M.-W. Chang, X. He, and J. Gao, "Semantic parsing via staged query graph generation: question answering with knowledge base," in *ACL*, 2015, pp. 1321–1331.
[25] A. Bordes, S. Chopra, and J. Weston, "Question answering with subgraph embeddings," in *EMNLP*, 2014, pp. 615–620.
[26] S. Jain, "Question answering over knowledge base using factual memory networks," in *NAACL*, 2016, pp. 109–115.
[27] Y. Wang, R. Zhang, C. Xu, and M. Yongyi, "The APVA-TURBO approach to question answering in knowledge base," in *COLING*, 2018, pp. 1998–2009.
[28] Y. Hao, H. Liu, S. He, K. Liu, and J. Zhao, "Pattern-revising enhanced simple question answering over knowledge bases," in *COLING*, 2018, pp. 3272–3282.
[29] E. Gabrilovich, M. Ringgaard, and A. Subramanya, "FACC1: Freebase annotation of ClueWeb corpora," *Technical report*, 2013.
[30] E. Pavlick, P. Rastogi, J. Ganitkevitch, B. V. Durme, and C. Callison-Burch, "PPDB 2.0: Better paraphrase ranking, finegrained entailment relations, word embeddings, and style classification," in *ACL*, 2015, pp. 425–430.
[31] S. Narayan, S. Reddy, and S. B. Cohen, "Paraphrase generation from latent-variable PCFGs for semantic parsing," in *INLG*, 2016, pp. 153–162.
[32] K. Xu, S. Reddy, Y. Feng, S. Huang, and D. Zhao, "Question answering on Freebase via relation extraction and textual evidence," in *ACL*, 2016, pp. 2326–2336.
[33] K. Xu, Y. Feng, S. Huang, and D. Zhao, "Hybrid question answering over knowledge base and free text," in *COLING*, 2016, pp. 2397–2407.
[34] J. Mueller and A. Thyagarajan, "Siamese recurrent architectures for learning sentence similarity," in *AAAI*, 2016, pp. 2786–2792.
[35] T. Rocktäschel, E. Grefenstette, K. M. Hermann, T. Kocisky, and P. Blunsom, "Reasoning about entailment with neural attention," in *ICLR*, 2016.
[36] M. J. Seo, A. Kembhavi, A. Farhadi, and H. Hajishirzi, "Bidirectional attention flow for machine comprehension," in *ICLR*, 2016.
[37] S. R. Bowman, G. Angeli, C. Potts, and C. D. Manning, "A large annotated corpus for learning natural language inference," in *EMNLP*, 2015, pp. 632–642.
[38] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural Computation*, vol. 9, pp. 1735–1780, 1997.
[39] Z. Tan, M. Wang, J. Xie, Y. Chen, and X. Shi, "Deep semantic role labeling with self-attention," in *AAAI*, 2018.
[40] J. Bao, N. Duan, M. Zhou, and T. Zhao, "Knowledge-based question answering as machine translation," in *ACL*, 2014, pp. 967–976.
[41] K. Xu, S. Zhang, F. Yansong, and D. Zhao, "Answering natural language questions via phrasal semantic parsing," *NLPCC*, vol. 496, pp. 333–344, 2014.
[42] J. Berant and P. Liang, "Semantic parsing via paraphrasing," in *ACL*, 2014, pp. 1415–1425.
[43] M.-C. Yang, N. Duan, M. Zhou, and H.-C. Rim, "Joint relational embeddings for knowledge-based question answering," in *EMNLP*, 2014, pp. 645–650.
[44] A. Graves and J. Schmidhuber, "Framewise phoneme classification with bidirectional LSTM networks," in *IJCNN*, 2005, pp. 5–6.
[45] P. Yin, N. Duan, B. Kao, J. Bao, and M. Zhou, "Answering questions with complex semantic constraints on open knowledge bases," in *CIKM*, 2015, pp. 1301–1310.
[46] J. Pennington, R. Socher, and M. C. D., "Glove: Global vectors for word representation," in *EMNLP*, 2014, pp. 1532–1543.
[47] J. Duchi, E. Hazan, and Y. Singer, "Adaptive subgradient methods for online learning and stochastic optimization," *JMLR*, vol. 12, 2/1/2011, pp. 2121–2159, 2011.
[48] W. Yin, M. Yu, B. Xiang, B. Zhou, and H. Schütze, "Simple question answering by attentive convolutional neural network," in *COLING*, 2016, pp. 1746–1756.



Yunshi Lan received her B.S. degree from the School of Mathematics and Statistics, Southwest University, Chongqing, China in 2015. She has been a Ph.D student in the School of Information Systems at Singapore Management University since 2015. Her research interests lie in natural language processing with a focucs on knowledge base question answering.

Shuohang Wang received his M.S. degree from the School of Software, Harbin Institute of Technology, Harbin, China in 2014. Since 2014, he has been a Ph.D student in the School of Information Systems at Singapore Management University. His research interests include question answering, text entailment, deep learning and reinforcement learning.

Jing Jiang received her Ph.D degree in Computer Science from the University of Illinois at Urbana-Champaign in 2008. Currently, she is an Associate Professor of Information Systems at Singapore Management University. Her research interests include natural language processing, text mining and machine learning.