Data Science and Engineering (2019) 4:323–335
https://doi.org/10.1007/s41019-019-00109-w

Leveraging Domain Context for Question Answering Over Knowledge Graph

Peihao Tong[1] · **Qifan Zhang**[1] · **Junjie Yao**[1]

Received: 19 August 2019 / Revised: 22 October 2019 / Accepted: 23 October 2019 / Published online: 4 November 2019

Abstract
With the growing availability of different knowledge graphs in a variety of domains, question answering over knowledge graph (KG-QA) becomes a prevalent information retrieval approach. Current KG-QA methods usually resort to semantic parsing, search or neural matching models. However, they cannot well tackle increasingly long input questions and complex information needs. In this work, we propose a new KG-QA approach, leveraging the rich domain context in the knowledge graph. We incorporate the new approach with question and answer domain context descriptions. Specifically, for questions, we enrich them with users' subsequent input questions within a session and expand the input question representation. For the candidate answers, we equip them with surrounding context structures, i.e., meta-paths within the targeting knowledge graph. On top of these, we design a cross-attention mechanism to improve the question and answer matching performance. An experimental study on real datasets verifies these improvements. The new approach is especially beneficial for specific knowledge graphs with complex questions.

Keywords Question answering · Knowledge graph · Meta-path

1 Introduction

Recent years have witnessed an information access paradigm shift, from a proactive search to voice/question-oriented automatic answering. A lot of personal assistants, e.g., Siri, Alexa Echo, and Google Home, are emerging. In these question answering services, we submit questions and get answers or suggestions. Under the hood, the crucial ingredient is the structured knowledge graph, including a full range of necessary information, constructed from related data sources.

Question answering over knowledge graph (KG-QA) has attracted many attentions [2, 6, 25, 26]. It accepts freestyle questions from users and responds with a direct answer, which is generated or retrieved from the underlying knowledge graph [9, 13]. Common ways of KG-QA include semantic parsing and retrieval based. The semantic parsing methods formalize the input question into the logical forms and locate the entities in the target knowledge graph. The retrieval methods conduct IR-based metrics to rank the candidate entities from the knowledge graph. With the recent improvement in deep learning algorithms, some embedding representation and generated models [6] are also introduced to tackle the problem of KG-QA as the matching between the input question and answers. Different from the previous parsing or retrieval ones, deep learning-based methods usually represent both questions and answers in the form of a semantic vector. First, the question in natural language is analyzed and transformed into the latent representation. Next, candidate answers are collected from the knowledge graph into the candidate set. Finally, the final answers are found in the candidate set which match the question properly.

Though remarkable progress has been achieved, KG-QA is still a challenging problem. The difficulty issues lie at not only the vague question description but also a variety of entity and relationship types within the knowledge graph [23, 31]. Take the insurance product domain as our motivation scenario; we have constructed a knowledge graph for insurance products (InsKG, abbrv. later in this paper) and set up an online question–answer service on top of it. This insurance product knowledge graph has more than 200k triples, consisting of insurance companies, categories, diseases, attributes, terms, etc. With intuitive and

✉ Junjie Yao
junjiey@gmail.com

[1] East China Normal University, Shanghai, China

comprehensive answers, the question–answer service on top of it has attracted almost 100k input questions from ordinary users. The returned answers of commonly used and the proposed KG-QA approaches are listed in Table 1. For factual questions (first question), i.e., related to product attributes, both current and proposed methods can return satisfying answers. However, for some general category relevant or survey questions (second, third), the current method cannot compete with the proposed approach.

By investigating the underlying knowledge graph, we find that difficult questions usually cover more nodes in the knowledge graph, and the corresponding nodes have more connections between them. Due to the complexity of KG-QA and the generality of current methods, they cannot uncover or utilize the context information within the knowledge graph. In contrast to the general ones, domain knowledge graphs have particular schema patterns, i.e., specific node and relationship types.

Motivated by the rich features of the domain knowledge graph, this paper introduces a new KG-QA model, incorporated with more domain context. Figure 1 shows the novel process of input question example. We equip the question representation with more corpus features and the generated answers with more meta-path features. Specifically, in the feature extraction stage, we extract the meta-path patterns and push the embedded graph features into the meta-path levels [10]. We also discuss the semantic parsing of input questions with the tree model [24] for better identifying the users' information needs. Besides, as an integrated neural generation KG-QA model, we fit the new approach into a cross-attention framework [13], which can model the interactive question–answer matching process.

State-of-the-art models usually initialize the embedding matrix for words or adopt some pre-trained word vectors. The vocabularies of the general pre-trained word embedding lack these proper nouns. Here, we improve the question representation with existing question corpus and novel question intent extraction.

In the representation of the answer, existing methods over open-domain knowledge graph usually adopt the translation model, i.e., TransE [5]. These methods embed the entire knowledge graph and obtain the vector representation for entities and relations in the knowledge graph. One advantage of the specific-domain knowledge graph is that the relation types are fixed, and it is suitable for

Table 1 QA cases of the current and proposed approaches

Question	Truth	State of the art	Proposed
What is the release time of this product ?	12/9/2015	12/9/2015	12/9/2015
What types of insurance products does China Life sell?	12, including disease, medicine, accident insurances, etc.	8, including disease, medicine, accident insurances, etc.	Same as ground truth
What categories does CPIC's critical disease insurance belong to?	Cancer, disease, investment	Cancer, disease	Cancer, disease, investment

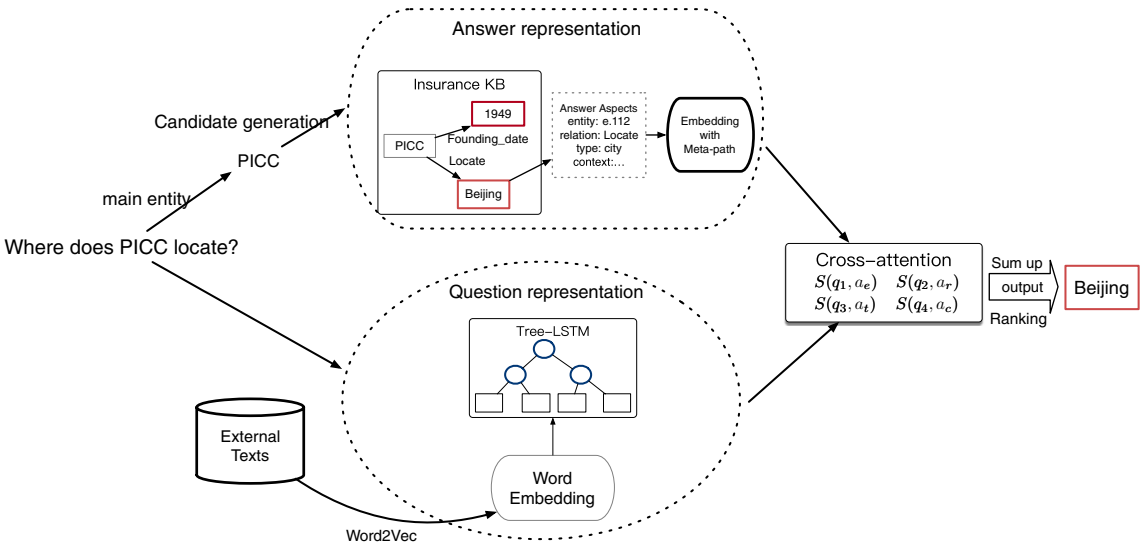

Fig. 1 KG-QA with the help of domain features

context pattern extraction. In this work, we resort to meta-path-based answer type extraction.

The contributions of this paper can be listed as follows:

1. We exploit the domain context in KG-QA, improving the QA performance with more KG patterns as the representation features.
2. We utilize a cross-attention model to match the question and answers in the KG-QA task, taking the rich connections between the comprehensive representation of questions and answers.
3. The experimental results demonstrate the effectiveness of our proposed approach, especially in domain knowledge graphs.

The remaining of this paper is organized as follows. We briefly review the related literature in Sect. 2. We then illustrate the framework and design of our new approach in Sect. 3. The experimental results are reported in Sect. 4.

2 Related Work

We briefly review the related research directions of KG-QA. Existing KG-QA models can be categorized into two lines, i.e., general and neural network-based methods.

Parsing and retrieval-based KG-QA In general KG-QA methods, semantic parsing and retrieval-based methods are popular ones. Semantic parsing-based methods compile the input natural language questions into logical forms, with the help of a combinatory grammar [7] or dependency-based compositional semantics [18]. The answers are then returned through SPARQL queries or other well-designed query processing mechanism. These methods are strict and usually take many efforts in the annotation and preprocessing stage [11, 16].

The retrieval-based methods turn to analyze the dependency of the words in the question and then use the target information extracted from the question to locate candidate answers from the knowledge graph. Final answers are selected from the candidate set by further quality or relevance evaluation [26].

Neural network-based KG-QA In recent years, with the rapid growth of the deep learning techniques, there are more and more tasks in the NLP field, including KG-QA task, benefiting from the neural network-based method. Recent deep learning methods usually transform questions and answers into the form of semantic vectors [4, 6] and then conduct the matching and ranking operations. Attention models received a lot of attentions, with its versatile ability to model the matching stage. Yin et al. [29] designed an attentive max-pooling layer for CNN to answer factual questions. Hao et al. [13] discussed attention-based end-to-end model with the global knowledge graph features for question answering. Li et al. [17] introduced the interactive model for question answering. We proceed to design a mutual attention model, combining both question and knowledge graph features, from the local and global perspectives.

Domain features in KG-QA Existing KG-QA methods are always designed for the open-domain knowledge graph. However, in reality, the application of a specific-domain knowledge graph is also widespread. Although the existing method mentioned above can be directly applied to a specific-domain knowledge graph QA task, several unique characteristics of the specific-domain knowledge graph are neglected. Yih et al. [27] developed a semantic parsing framework utilizing convolutional neural networks to process single-relation questions. Yang et al. [25] proposed a model to map the natural language questions into logical forms by leveraging the semantic associations between lexical representations and KG properties in the latent space.

However, sequential recurrent neural network-based models ignore the constituency and recursion information in natural language. Yin et al. [28] utilized recursive neural networks to understand the question, and the experiment demonstrated that recursive neural networks are more appealing to characterize the semantics of a query in the QA task. With the benefits for relation inference, knowledge graph completion, even collective classification, and relation detection in a knowledge graph are becoming more and more important. Several detection methods are proposed and demonstrated. Fan et al. [12, 14] discussed the reasoning framework and translation-based methods for relation detection in knowledge graph. Miyanishi et al. [21, Zhang et al. 31] presented type selection approach. Yu et al. [30] discussed relation detection for precise question match with edge descriptions.

The proposed approach in this paper follows fusion paradigm. We extract the question and answer representation in a more general way and provide more comprehensive question–answer matching model. Recently, some combined or fused methods are proposed, and at the same time, some domain features are investigated in the models. Deng et al. [8] utilized a multitask learning framework to solve answer selection and relation detection simultaneously. Qu et al. [22] proposed a similarity matrix-based CNN model to improve the detection performance. This paper differs from these recent works, and we focus on answer relation detection and composition mutually. We further profile the question representation in a tree structure and the knowledge graph representation with meta-path. The new approach presents a more general context feature extraction approach.

3 Proposed Approach

In this section, we first present the general framework for the proposed KG-QA approach, and then we reveal our improvement designs, especially domain context feature extraction. KG-QA is designed to extract the answer set A from the knowledge graph when the input question q is given. KG-QA process involves the input question and the underlying knowledge graph, respectively. Input questions are in the form of natural language and consisting of a set of words. The knowledge graph is regarded as a set of entities and the associated edges, i.e., fact triples, in the form of (*subject*, *predicate*, *object*).

3.1 Framework

We introduce the design of the proposed KG-QA framework proposed in Fig. 2. In the domain context feature extraction and representation layer, the proposed approach processes input question with two modules, i.e., question representation and answer representation. For the specific-domain knowledge graph, we make some modifications to the general framework. As mentioned in Sect. 1, we carefully design the novel domain context features. Both the answer- and question-related contexts are utilized.

The main steps of this model can be divided into three steps. First, the question representation is extracted. Second, according to the question, we select the candidate set from the knowledge graph and generate the answer representation. Finally, to find the answer which matches the input question more accurately, a similarity score function S is designed to measure the correlation between the question and the candidate answers. The answer with the highest score will be selected as the final answer.

To represent the input question, we collect the related text (users' existing input questions) in the specific domain and train the word embedding using Word2vec. After that, we parse the input question with the help of the tree model, which profiles the users' question intents, i.e., targeted entities in the question. We adopt the tree structure LSTM to understand the question and obtain the representation of questions.

For the answers, the knowledge graph embedding method TransE [5] is usually used to obtain the representation for different answers. We adopt the meta-path random walk [10] to capture the context of the targeted entities in the knowledge graph. The extracted paths are used to represent the entities and later answers.

After the domain context feature extraction, in the matching stage, the cross-attention model [13, 17] is designed to capture the relations between input questions and selected answers. Finally, the score function is trained to measure the matching score between the representation of questions and candidate answers.

3.2 Input Question Representation

Formally, the question q consists of several words $q = (w_1, w_2, \ldots, w_n)$, where w_i means the ith word in the question. For each word, x_{w_i} denotes the latent representation of the word w_i, i.e., embedded vectors in our case.

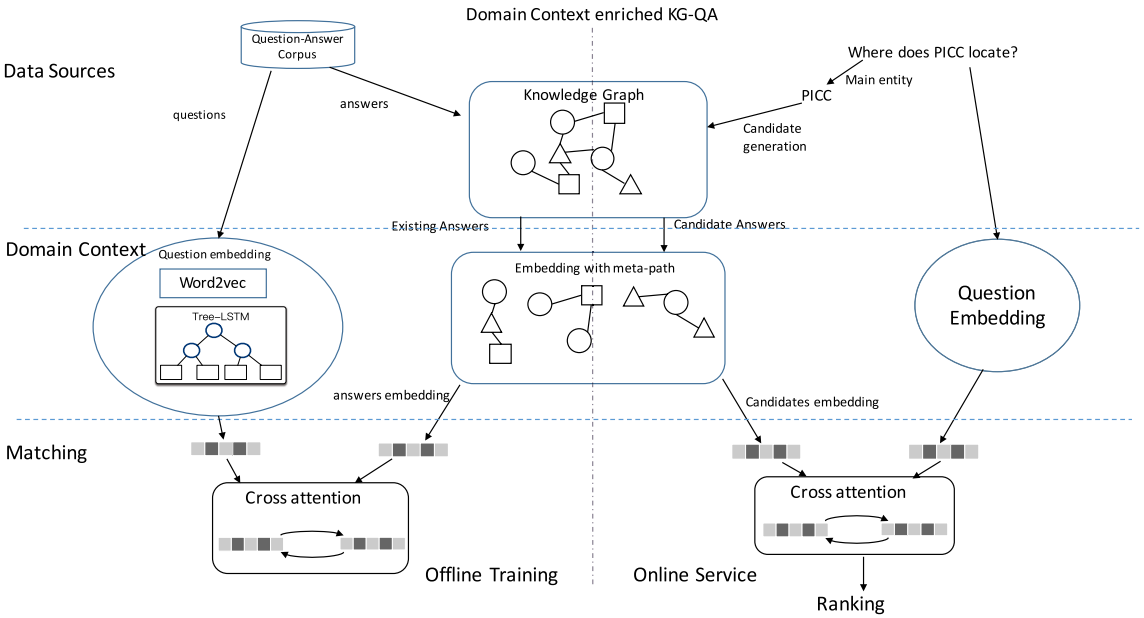

Fig. 2 Domain context for KG-QA tasks

As shown in Fig. 2, the question representation has two stages. In offline training, we collect external text resources, to embed and represent the questions. After that, we parse the input question with the help of the tree model, which parses the users' question intents, i.e., targeted entities. In the online response, the extracted embedding vectors are used to represent the input questions.

3.2.1 Question Embedding

Hence, we train the word embedding using this related text. For example, in the insurance knowledge graph, we collect the text of users' existing input questions, the company, and the product descriptions as the text corpus and utilize the Word2Vec to generate word embedding.

Speaking of the word embedding, we choose Word2Vec [19, 20] method to train the word vectors, with the help of the specific-domain text corpus. Here, we use the users' input question logs. The word embedding vectors of InsKG are generated and then used as input to the later question training and online response.

3.2.2 Question Parsing

We continue to discuss the tree-structured LSTM model, a variant of recursive neural networks [24, 28]. Different from sequential LSTM, which only allows strictly sequential information propagation, the tree-LSTM is a network that accepts tree-structured input. Moreover, the architecture of the network will be constructed according to the tree structure of the extracted concept layers from the input question. Here, we set up constituency trees for each question and use tree-LSTM to handle the constituency tree. The constituency tree is intuitive to cover different semantic modules within it, beneficial for the question representation.

Take the question "What products does PICC provide?" as an example. Its corresponding constituency tree is shown in Fig. 3. The left subtree contains the words "What" and "products," while the right subtree contains the words "does," "PICC," and "sell." In this question, "PICC" is the main entity, denoting the target, and "products" is a noun indicating the question type and "sell" is a verb, meaning the relation between the target and answers. Therefore, the question type information and relation information are implied in the left subtree and right subtree, respectively.

Intuitively, the representation of the left subtree's root is valuable in deciding answer type, and the representation of the right subtree's root is precise in deciding answer relations. As a consequence, when generating the embedding for questions, we only take the vector of the root, root's left child, and root's right child into account, respectively, h_{root}, h_{left}, and h_{right}. We propose an attention mechanism-based method to generate the final representation according to different answer aspects, which will be explained in Sect. 3.4 in detail.

The left subtree of the constituency tree denotes the noun phrase, while the right subtree corresponds to the verb phrase. Concretely, the left subtree "What products" is more likely to decide the type of the answer. In this example, the answer entity's type should be insurance products. The right subtree "does PICC sell" is more likely to indicate the relation between the main entity and the answer entity. Since this question's main entity is "PICC," the relation between PICC and answers entity should be "sell."

3.2.3 Question Output

After the generation of the constituency tree, the tree-LSTM is adopted to handle the questions. The following equations show the underlying tree-LSTM. The leaf units of the tree-LSTM receive the word representation as input, and the internal units take their left and right children's output as the input.

$$i_j = \sigma(W^{(i)}x_j + U_l^{(i)}h_{jl} + U_r^{(i)}h_{jr} + b^{(i)}) \tag{1}$$

$$f_{jk} = \sigma(W^{(f)}x_j + U_{kl}^{(f)}h_{jl} + U_{kr}^{(f)}h_{jr} + b^{(f)}) \tag{2}$$

$$o_j = \sigma(W^{(o)}x_j + U_l^{(o)}h_{jl} + U_r^{(o)}h_{jr} + b^{(o)}) \tag{3}$$

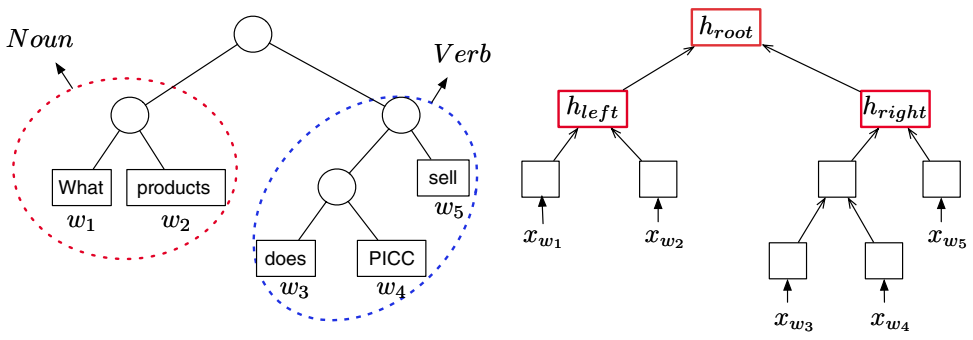

Fig. 3 Constituency tree and the corresponding tree-LSTM

(a) Constituency tree **(b)** Tree-LSTM architecture

$$u_j = tanh(W^{(u)}x_j + U_l^{(u)}h_{jl} + U_r^{(u)}h_{jr} + b^{(u)}) \quad (4)$$

$$c_j = i_j \odot u_j + f_{jl} \odot c_{jl} + f_{jr} \odot c_{jr} \quad (5)$$

$$h_j = o_j \odot tanh(c_j). \quad (6)$$

When the node is a leaf node, the input h is set to zero. On the contrary, when the node is an internal node, the input x is set to zero. Equations (1)–(3), respectively, define the input gate, forget gate, and output gate in the tree-LSTM. In Eq. (2), the k denotes a binary index variable that indicates the left or right child of the current node.

3.3 Answer Representation

3.3.1 Answer Aspects

In this paper, we categorize four different kinds of answer aspects, i.e., answer entity a_e, answer relation a_r, answer type a_t, and answer context a_c. The context set of an answer is $C = \{e_{c_1}, e_{c_2}, \ldots, e_{c_n}\}$, and the final answer context representation is calculated as $e_c = \frac{1}{n}\sum_i^n e_{c_i}$.

Specifically, the answer entity denotes the entity identification in the knowledge graph. Answer relation indicates the relation path from the main entity to answer entity. For each answer in the training or online response stages, we generate latent representation e_e, e_r, e_t, and e_c for each aspect. If the relation path only includes one relation, we can directly use the relation representation as e_r. If there is more than one relation in the path, the relation path representation should be calculated in $e_r = \frac{1}{n}\sum_i^n e_{r_i}$. For answer type, in InsKG, there is a special relation called "*entity_type*" denoting the entity's type, and we use it as the answer type in our model. For answer context, we first take all entities on the path from the main entity to answer entity into consideration and generate a context set. Moreover, all the neighbors connected to the set members will be added to the set too.

For the example question "Where does PICC locate?", the answer comes from the triple (*PICC*, *locate*, *Beijing*). Here, the entity identification for entity "Beijing" is "e.112" in the knowledge graph; thus, "e.112" is selected as an answer entity. As to answer relation, "locate" should be chosen. For answer type, we find the triple (*Beijing*, *entity_type*, *city*), and we take the "city" as the answer type.

3.3.2 Answer Extraction

To fully utilize the structure information of the knowledge graph, we employ the knowledge graph embedding method TransE [5]. TransE takes the pairwise training as the training strategy; thus, for every fact triple (h, r, t), we should sample negative triple (h', r, t'). The basic idea of TransE is trying to make the distance between $h + r$ and t shorter, while $h' + r$ and t' should be far away. TransE usually utilizes the $L1$-norm or $L2$-norm to define the distance $d(h + r, t)$. In this paper, we use the $L2$-norm as a distance function.

For every fact triple (h, r, t), negative triple (h', r, t') is randomly sampled. The basic idea of TransE is trying to make $h + r \approx t$, while $h' + r$ should be far away from t'. Meanwhile, in the training step, the loss function is given in Eq. (7):

$$\mathcal{L} = \sum_{(h,r,t)\in S}\sum_{(h',r,t')\in S'} [\gamma_k + d(h + r, t) - d(h' + t, t')]_+ \quad (7)$$

Then, we should minimize the margin-based loss function over the training set. The $[]_+$ here means the function $max(0, z)$. The γ_k here denotes the margin, and the S' is the set of negative triples. However, there are millions of entities and relations in Freebase, which costs too much if we run the TransE on the whole knowledge graph. Moreover, to avoid costly computation, we only reserve part of the entities in the knowledge graph, which may be used in the QA task and also keep the relations between these entities to reduce the scale of the knowledge graph.

As mentioned above, for the representation of answer, the translation model TransE is usually adopted to embed the entire knowledge graph, so as to obtain the vector representation for entities, relations, and types in the knowledge graph. However, TransE model always ignores the path information in the knowledge graph. In addition, the quality of the representation generated by TransE is sometimes conditioned by the initialization of the vector. The poor initialization may lead to inadequate representation. In the specific-domain knowledge graphs, the relation types are usually limited, which means that it is suitable to capture and summarize the paths. Here, we propose an approach that utilizes the meta-path to model the path context in the knowledge graph.

Take the insurance domain as the motivating example; we define a meta-path scheme $p_i = (Company \rightarrow Product \rightarrow Type \rightarrow Product \rightarrow Company)$.

Under this scheme, we sample several sequences of nodes in the knowledge graph. As the example shown in Fig. 4, the path $(Pingan \rightarrow p_1 \rightarrow term\ insurance \rightarrow p_2 \rightarrow ChinaLife)$ marked with the red solid line is one of the chosen sequence. The meta-path usually implies rich information. The marked path means that the companies *Pingan* and *ChinaLife* provide products of the same type. From this path, we find that the companies *Pingan* and *ChinaLife* are similar, and they would be embedded close in the latent representation space.

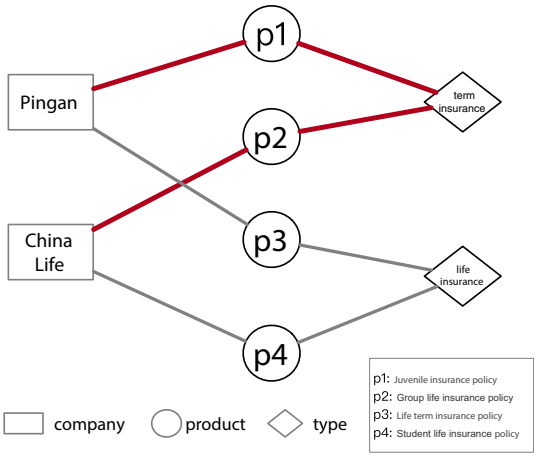

Fig. 4 Meta-path examples

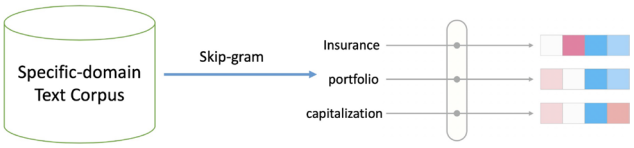

Fig. 5 Skip-gram examples

3.3.3 Answer Output

We choose several meta-path schemes to form a scheme set as P, based on prior knowledge or expert advice. The meta-path schemes in knowledge graph can be denoted in the form of $(e_1 \rightarrow e_2 \rightarrow \cdots \rightarrow e_n)$. After the selection of meta-path schemes, we utilize them to sample sequences in the knowledge graph. Then, these sequences are fed into the Skip-gram model as the input [20], which shows in Fig. 5 a natural language processing model, and are widely applied in the embedding field.

3.4 Cross-attention

Usually, in the question representation generation, there are several strategies to generate the final question representation. However, these approaches cannot fully utilize the abundant information of the constituency tree. We propose a cross-attention mechanism to improve the representation. In the question–answer matching stage of KG-QA tasks, the objective is to choose correct answers from a candidate set according to the given question. The overview of cross-attention mechanism [15] used in this work is shown in Fig. 6.

It consists of two attention models, i.e., answer-towards-question and question-towards-answer attention.

1. Given one answer's type, we should re-read the given question to target the corresponding part related to the answer type. As one answer usually has several aspects, we can re-read the question several scans, i.e., answer-towards-question attention.
2. On the other hand, when we concatenate different question parts together, we could re-read the answer to identify more important parts. It is the effect of question-towards-answer attention.

It is vivid that the cross-attention model can be interpreted as a re-reading mechanism to mutually align the question and answers. The following details will reveal the closeness matching process, with the help of a cross-attention model.

3.4.1 Answer-Towards-Question Attention

Here, we discuss how to combine the h_{root}, h_{left}, and h_{right}. The weight of attention is measured by the relevance between each tree node on question representation and answer aspect representation. Final representation should be dynamic according to the different aspects of the answer that each answer aspect should focus on different nodes in the constituency tree of the same question.

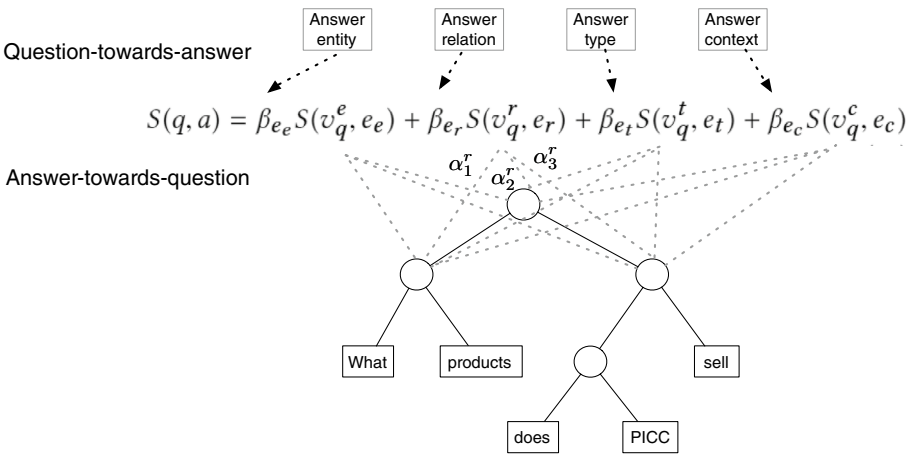

Fig. 6 Cross-attention mechanism

$$\alpha_{ij} = \frac{\exp(w_{ij})}{\sum_{k=1}^{n} \exp(w_{ik})} \tag{8}$$

$$w_{ij} = tanh(W^T[h_j;e_i] + b). \tag{9}$$

Here, α_{ij} denotes the attention weight between an answer aspect a_i and tree node representation h_j where $h_j \in \{h_{left}, h_{root}, h_{right}\}$. For each answer aspect, there is a corresponding question representation.

When a question is extracted into a constituency tree, the left subtree's root corresponds to the noun phrase, i.e., question type, while the right subtree's root corresponds to the verb phrase, as a kind of question focus. We may pay more attention to the h_{left} in the choice of the final answer type. We also pay more attention to h_{right} when choosing the answer relation. For answer type, the corresponding question representation is defined as follows:

$$v_q^t = \alpha_1^{type} h_{left} + \alpha_2^{type} h_{root} + \alpha_3^{type} h_{right}. \tag{10}$$

v_q^r, v_q^e, and v_q^c are formalized in the same way. Given a candidate answer, we derive the scores of different aspects, which are $S(v_q^e, e_e)$, $S(v_q^r, e_r)$, $S(v_q^t, e_t)$, and $S(v_q^c, e_c)$, respectively, where the scoring function $S(\cdot)$ is the inner product between question representation and aspect representation.

After we get all the aspect scores, a straightforward answer-decision strategy is to sum these scores up and select the candidate with the highest score as the final answer.

3.4.2 Question-Towards-Answer Attention

Questions should also take varying attention to different answer aspects. As we know, different questions should concentrate on different answer aspects, which leads to question-towards-answer attention. We generate question-towards-answer attention distribution β and β_{e_e}, β_{e_r}, β_{e_t}, β_{e_c}, respectively, denoting the attention weight for each answer aspect. This weight distribution reflects the importance of different answers aspects w.r.t. the given question.

$$\beta_{e_i} = \frac{\exp(w_{e_i})}{\sum_{e_k \in e_e, e_r, e_t, e_c} \exp(w_{e_k})} \tag{11}$$

$$w_{e_i} = tanh(W^T[\bar{q};e_i] + b) \tag{12}$$

$$\bar{q} = \frac{1}{3}(h_{left} + h_{root} + h_{right}). \tag{13}$$

For example, in some QA cases, the answer type may play a key role in determining the final answer. So the corresponding weight β_{e_t} will be larger than other weights. The final score of each answer is summed up by scores from different answer aspects in the question-towards-answer attention. Candidate with the highest score is selected.

$$S(q,a) = \beta_{e_e} S(v_q^e, e_e) + \beta_{e_r} S(v_q^r, e_r) + \beta_{e_t} S(v_q^t, e_t) + \beta_{e_c} S(v_q^c, e_c). \tag{14}$$

3.5 Question–Answer Match

For each QA pair, we construct a candidate answer set for the question. Then, we expand the candidate set with incorrect answers. For each question, we generate m negative samples from the entities connected to the primary entity within $k - hops$. For example, if the number of main entity's neighbors within $1 - hop$ is more than m, we will randomly choose m negative samples from the $1 - hop$ neighbors. If the number of $1 - hop$ neighbors is less than m, we will randomly select the negative samples within $2 - hop$. We will expand the hop boundaries until there are enough negative samples.

3.5.1 Offline Traning

In the model training procedure, we employ the pairwise training strategy. After generating the training set, the pairwise training loss is defined in the following:

$$\mathcal{L}_{q,a,a'} = \sum_{a \in P_q} \sum_{a' \in N_q} [\gamma + S(q,a') - S(q,a)]_+. \tag{15}$$

Here, a denotes the correct answer, a' denotes the wrong answer, and γ is the margin which is always a small positive real number within 1.

The $[]_+$ here denotes the function $max(0, z)$. The basic idea of this training strategy is that the score of a question paired with a correct answer is higher than any wrong answer by at least γ. As a result, once the score of correct answers $S(q, a)$ is no higher than the score $S(q, a') + \gamma$, the loss will be counted. The minimized loss objective function is defined as follows:

$$\min \sum_q \frac{1}{|P_q|} \mathcal{L}_{q,a,a'}. \tag{16}$$

In the optimization process, we adopt the stochastic gradient descent (SGD) to minimize the learning process with mini-batches utilized.

3.6 Online Response

During the question answering stage, for each QA pair, we have the question q and the candidate answer set C_q. For every candidate answer $\hat{a} \in C_q$, we calculate the score

$S(q, \hat{a})$. Finally, the scores are ranked, and the answer with the highest score is selected as the final answer.

However, it is worth noting that some questions have more than one correct answer. The strategy mentioned above is improper because only one answer will be returned. We define the margin m, and the answers whose gap to the best score is less than m will be put into the final answer set $\hat{A}_q$ as follows:

$$\hat{A}_q = \{\hat{a} | S_{max} - S(q, \hat{a}) < m\}. \tag{17}$$

In practice, we find that for some questions, the number of candidate answers in the knowledge graph is very large, and the distribution of answers is unbalanced. Therefore, we could utilize a heuristic method to prune the candidate answer set. As observed from the candidate answers set, there are a lot of candidate answers that share the same answer type and answer relation. We first aggregate the candidate answers based on their answer type and answer relation. For each group, if the number of answers is greater than a threshold θ, we randomly select θ candidate answers from this group and put them into a new candidate set. If the number of groups is less than θ, all the answers in this group result are added to the new candidate set. After processing all groups, we generate a new candidate set, which we will use later to calculate the scores. Next, we rank all the candidate answers according to their scores and generate predicted answers by the procedures we have mentioned above.

For every predicted answer, if the rest answers in its group are not in the candidate set, we should re-put them into the candidate set for calculating. In this way, we have reduced the number of calculations, and it also makes the number of answers in the candidate set more balanced.

4 Experiments

Here, we report both the qualitative and quantitative studies of the proposed domain-aware KG-QA model.

4.1 Experimental Setup

4.1.1 Datasets

Two datasets are used in the experimental study. One is the mentioned insurance knowledge graph InsKG, and the other is commonly used in open-domain WebQuestions [1].

In the WebQuestions dataset, there are 5.8k q-a pairs, containing 3,778 training pairs training and 2,032 testing pairs. The questions are collected from Google Suggest API, and the answers are annotated by Amazon Mechanical Turk. The dataset is extracted from the open-domain knowledge graph Freebase [2].

The InsKG has fixed scheme patterns. As mentioned before, InsKG contains 200k triples involving insurance companies, insurance products, terms in the insurance field, and so on. In the InsKG dataset, we randomly select 2k question–answer pairs from the users' provided questions in our InsKG service.

4.1.2 Baselines

We use accuracy and average F1 scores in the experimental study. We choose several recent works as baselines in this experimental study:

- SimpleEmbedding [6]. Bag-of-words model is used to generate question and answer vectors.
- SubgraphEmbedding [3]. It takes the candidate answer's neighbor nodes within 2-hops for embedding.
- MemNNs [4]. It uses the memory network to embed the inputs.
- MCCNNs [9]. It includes different aspects of the answers, and text CNN is used for each question.
- Bi-LSTM [13]. It designs a bi-LSTM model to profile the words' forward and backward dependencies.
- Tree-LSTM with A-Q. The tree-LSTM model is utilized but only with answer-towards-question attention, which is a common attention model.

The WebQuestions dataset has a more varying scheme and covers more domains. Therefore, in the WebQuestions experiment, it is noting that our proposed method is lacking question and answer representation modifications.

4.1.3 Parameter Settings

We adopt the mini-batch stochastic gradient descent (SGD) as the optimization method to minimize the pairwise training loss. The batch size is set to 100, and the learning rate is set to 0.01.

In the InsKG experiment, the word embedding dimension is set to 128 and the hidden size of the LSTM cell is set to 128. The margin here for pairwise training is set to 1. The negative example number is set to 200. For the meta-path, with the help of the heuristic algorithm, the method generates 18 meta-path schemes, and we keep 12 meta-path schemes according to prior knowledge.

In the WebQuestions experiment, the vector dimension of words is set to 256, and thus the tree structure LSTM cell's hidden size is also set to 256. The margin γ for pairwise training is set to 0.8. For every positive answer, we sample 1000 negative examples. For knowledge graph embedding, we set the embedding dimension to 256. The margin γ_t for TransE is set to 1.

All these hyperparameters of the proposed network are determined according to the performance on the validation set.

4.2 Quantitative Study of KG-QA

4.2.1 General Dataset, WebQuestions

KG-QA results on the WebQuestions dataset are listed in Table 2.

The neural network methods have better ones, but cannot compete with LSTM variants. The proposed approach has the best performance, but just with a slight increase.

SimpleEmbedding and SubgraphEmbedding merely get the worst results under the bag-of-word assumptions. Bi-LSTM drops the bag-of-words assumption and considers three different aspects of answers and resorts to the sequence modeling. Tree-LSTM model with answer-towards-question attention also has achieved a better performance than the previous work. The proposed model takes both question-towards-answer and answer-towards-question attention into account. The improvements show the advantage of deep learning models.

MemNNs also uses the bag-of-words model to represent the questions, but it achieves a relatively better performance by designing the KG-QA model with the help of the memory networks framework.

MCNNs drop the bag-of-words assumption and consider three different aspects of answers.

4.2.2 Specific Domain, InsKG

Results on InsKG are shown in Fig. 7. Similar to the experiments on the general WebQuestions dataset, embedding lines cannot compete with LSTM methods. The proposed Tree-LSTM with modifications model has significant improvement. When the training ratio is 60% in Table 3, the improvement is most significant that the proposed method ranks first at 76.8% and followed by common Tree-LSTM at 72.5%. It proves the usefulness of the meta-path in capturing the contextual structure information in the knowledge graph, as well as the value of domain information in modeling vector for answers.

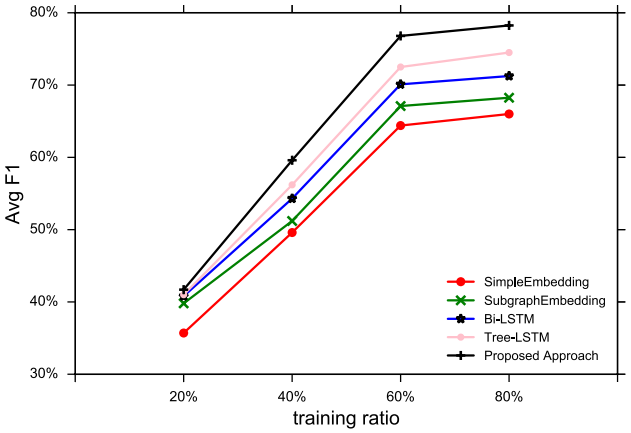

Fig. 7 QA result on InsKG

Compared to the general model Tree-LSTM, Tree-LSTM can effectively improve the performance in QA tasks over specific-domain knowledge graphs.

These quantitative QA experiments reveal that the domain context features are valuable for KG-QA, and the proposed domain context KG-QA is better at specific-domain knowledge graph usages.

Additionally, we find that the model which utilizes meta-path for sampling in the knowledge graph achieves even better performance than the model Tree-LSTM (W2V).

4.3 Model Component Analysis

4.3.1 Answer Context Contribution

The KG-QA experiments have revealed the contributions of meta-path features. We continue to analyze the details of different path types. In this experiment on InsKG, we take "*company* → *product* → *type* → *type*," "*product* → *type* → *type* → *type*," and "*company* → *product* → *status*" as the starting meta-path schemes, as shown in Fig. 8. Within each meta-path scheme, we compare their corresponding questions' QA performances of Tree-LSTM and Tree-LSTM with meta-path modification methods.

Table 2 QA result on WebQuestions

Methods	Avg F1
SimpleEmbedding	29.7
SubgraphEmbedding	39.2
MemNNs	42.2
MCCNNs	40.8
Bi-LSTM	42.9
Tree-LSTM with A-Q	43.6
Proposed approach	44.1

Table 3 QA accuracy on InsKG (training ratio 60%)

Methods	Avg F1
SimpleEmbedding	64.4
SubgraphEmbedding	67.1
Bi-LSTM	70.1
Tree-LSTM	72.5
Proposed approach	76.8
Only with W2V	74.6
Only with MP	75.2

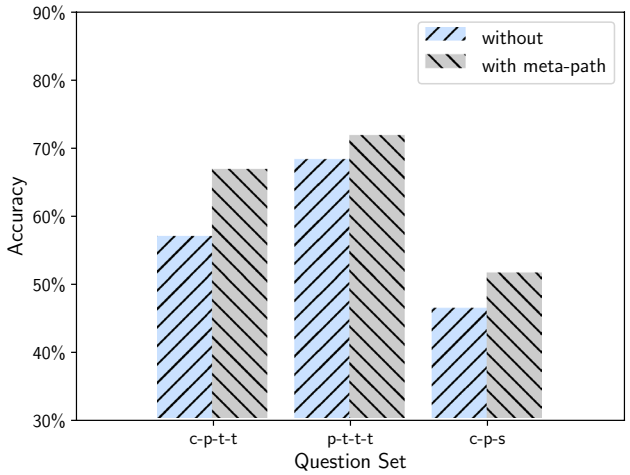

Fig. 8 QA accuracy under different meta-path context

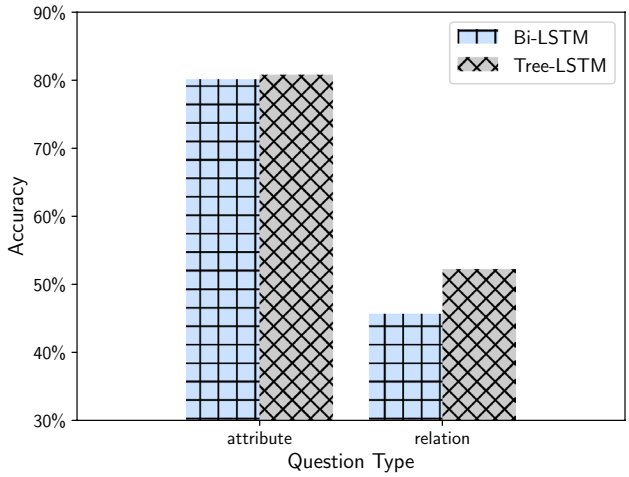

Fig. 9 QA accuracy with/without tree model in questions

The results show that for some simple/short meta-path scheme, two methods get similar results. In the complex/long meta-path scheme, the method with the meta-path feature gains an impressive improvement. The meta-path features are valuable for complex pattern extraction, leveraging the KG-QA performance.

In contrast, such meta-path can significantly improve the accuracy of "type" questions, 10.5%. Here, the 'type' questions refer to examples like "What types of insurance products does China Life sell?".

4.3.2 Question Context Contribution

We further analyze the contribution from the question's tree model extraction. Here, we classify the questions into fact and relation types based on the returned answer sets. In each category, we compare their QA performance of bi-LSTM and Tree-LSTM models. The results are shown in Fig. 9.

We find that common fact/attribute questions have roughly the same performance with/without tree models. It is reasonable that, for factual questions, their returned answers usually cover just the entity and directly connected attributes or categories, and the tree model parsing of input questions cannot deliver additional associations. In contrast, for relationship questions, sometimes their answers cover several entities and the interconnected edges in the knowledge graph. The tree model is good at representing the question intent/type and question focus/entity topic. Though the relationship questions take a small portion of users' input questions, the tree model's parsing potential contributes to the answering of the complex questions.

By carefully checking the knowledge graph, we find that the products' types in the knowledge graph are in hierarchical structures. Based on the analysis of the results of the models without meta-path, we find that because of the type of insurance products often have a hierarchical relationship. For example, in the question "What types of insurance products does China Life sell?", China Life sells a product whose type is cancer insurance. Thus, the cancer insurance should be included in the answer set. However, cancer insurance belongs to a more general category, i.e., insurance type, which is disease insurance. Disease insurance should also be returned in the result, though it is usually ignored.

With the help of the meta-path information, this problem is relieved. This example proves that the meta-path "*company* → *product* → *type* → *type*" well captures the relationship between types.

We find that even only one kind of optimization is adopted, we can get a certain degree of improvement in performance. It proves the effectiveness of pre-word embedding and Meta-path. At the same time, the performance of using Meta-path is better than that of using PWE. It demonstrates that the use of meta-path can well capture the relationship information between entities in the knowledge graph.

4.3.3 Cross-attention Models

To illustrate the contributions of the cross-attention mechanism, we continue to take the question "What city does PICC locate?" as the running example. In this question, the answer relation path is "locate" and the answer type is denoted as "location.city." We consider four answer aspects, which are answer entity, answer relation, answer type, and answer context. In the heat map, the abscissa of the table is the left subtree's root, root, right subtree's root, and the ordinate is the answer entity, answer relation, answer type, and context. The heavier the color, the higher the value of attention weight (Fig. 10).

As discussed in Sect. 3, we use the constituency tree to parse the input question. The left subtree plays a more

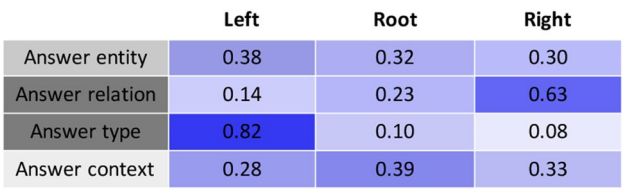

Fig. 10 Cross-attention heat map

important role in predicting the type of the answer, while the right subtree is more critical in answer relation prediction. For the answer-towards-question attention, we first focus on the attention distribution for answer type. The results show that the left subtree root's weight is more significant, indicating the dominant role that the left subtree plays in the answer type prediction. The right subtree root obtains the most significant value weight in the distribution. For the question-towards-answer attention, the answer type and the answer relation are more important in the final answer selection. It is vivid that the distribution of attention weights fits our expectations, and the tree model well uncovers the intents of the input question.

For the question-towards-answer attention, we find the answer type, and the answer relation is more important in the final answer decision, which is also in line with our intuitive experience.

In conclusion of the experimental study, we find that the proposed approach shows an advantage in both the KG-QA tasks and the question/answer explanation. The introduced meta-path features are beneficial for complex answer processing. The tree structure is good at question understanding. For the KG-QA performance, the new approach is especially useful in domain knowledge graph usages.

5 Conclusion

In this paper, we propose a new KG-QA approach by leveraging the domain context, with the help of a cross-attention model. We parse the question tree and utilize meta-path to enrich the representation for answers. We also introduce the cross-attention mechanism and reveal the mutual influences between the questions and answers. The experimental results demonstrate the effectiveness and improvement in the proposed model in the specific-domain knowledge graph. Besides the improved KG-QA performance, the introduced domain context is better at capturing the correlation within the knowledge graphs. We are investigating its extensions in the relationship inference and entity analytics in future work.

Acknowledgements This work is supported by Natural Science Foundation of China 61972151, 61502169, U1509219 and the Fundamental Research Funds for the Central Universities.

References

1. Berant J, Chou A, Frostig R, Liang P (2013) Semantic parsing on freebase from question–answer pairs. In: Proceedings of EMNLP, pp 1533–1544
2. Bollacker KD, Evans C, Paritosh P, Sturge T, Taylor J (2008) Freebase: a collaboratively created graph database for structuring human knowledge. In: Proceedings of SIGMOD, pp 1247–1250
3. Bordes A, Chopra S, Weston J (2014) Question answering with subgraph embeddings. In: Proceedings of EMNLP, pp 615–620
4. Bordes A, Usunier N, Chopra S, Weston J (2015) Large-scale simple question answering with memory networks. arXiv preprint arXiv:1506.02075
5. Bordes A, Usunier N, García-Durán A, Weston J, Yakhnenko O (2013) Translating embeddings for modeling multi-relational data. In: Proceedings of NIPS, pp 2787–2795
6. Bordes A, Weston J, Usunier N (2014) Open question answering with weakly supervised embedding models. In: Proceedings of ECML, pp 165–180
7. Cai Q, Yates A (2013) Large-scale semantic parsing via schema matching and lexicon extension. In: Proceedings of ACL, pp 423–433
8. Deng Y, Xie Y, Li Y, Yang M, Du N, Fan W, Lei K, Shen Y (2019) Multi-task learning with multi-view attention for answer selection and knowledge base question answering. In: Proceedings of AAAI
9. Dong L, Wei F, Zhou M, Xu K (2015) Question answering over freebase with multi-column convolutional neural networks. In: Proceedings of ACL, pp 260–269
10. Dong Y, Chawla NV, Swami A (2017) metapath2vec: Scalable representation learning for heterogeneous networks. In: Proceedings of KDD, pp 135–144
11. Fader A, Zettlemoyer L, Etzioni O (2014) Open question answering over curated and extracted knowledge bases. In: Proceedings of KDD, pp 1156–1165
12. Fan Z, Wei Z, Li P, Lan Y, Huang X (2018) A question type driven framework to diversify visual question generation. In: Proceedings of IJCAI, pp 4048–4054
13. Hao Y, Zhang Y, Liu K, He S, Liu Z, Wu H, Zhao J (2017) An end-to-end model for question answering over knowledge base with cross-attention combining global knowledge. In: Proceedings of ACL, pp 221–231
14. He X, Qian W, Fu C, Zhu Y, Cai D (2018) Translating embeddings for knowledge graph completion with relation attention mechanism. In: Proceedings of IJCAI, pp 4286–4292
15. Hermann KM, Kociský T, Grefenstette E, Espeholt L, Kay W, Suleyman M, Blunsom P (2015) Teaching machines to read and comprehend. In: Proceedings of NIPS, pp 1693–1701
16. Kwiatkowski T, Zettlemoyer LS, Goldwater S, Steedman M (2010) Inducing probabilistic CCG grammars from logical form with higher-order unification. In: Proceedings of EMNLP, pp 1223–1233
17. Li H, Min MR, Ge Y, Kadav A (2017) A context-aware attention network for interactive question answering. In: Proceedings of KDD, pp 927–935
18. Liang P, Jordan MI, Klein D (2013) Learning dependency-based compositional semantics. Comput Linguist 39(2):389–446

19. Mikolov T, Chen K, Corrado G, Dean J (2013) Efficient estimation of word representations in vector space. In: ICLR
20. Mikolov T, Sutskever I, Chen K, Corrado GS, Dean J (2013) Distributed representations of words and phrases and their compositionality. In: Proceedings of NIPS, pp 3111–3119
21. Miyanishi T, Hirayama J, Kanemura A, Kawanabe M (2018) Answering mixed type questions about daily living episodes. In: Proceedings of IJCAI, pp 4265–4271
22. Qu Y, Liu J, Kang L, Shi Q, Ye D (2018) Question answering over freebase via attentive RNN with similarity matrix based CNN. arXiv preprint arXiv:1804.03317
23. Saha A, Pahuja V, Khapra MM, Sankaranarayanan K, Chandar S (2018) Complex sequential question answering: towards learning to converse over linked question answer pairs with a knowledge graph. In: Proceedings of AAAI, pp 705–713
24. Tai KS, Socher R, Manning CD (2015) Improved semantic representations from tree-structured long short-term memory networks. In: Proceedings of ACL, pp 1556–1566
25. Yang M-C, Duan N, Zhou M, Rim H-C (2014) Joint relational embeddings for knowledge-based question answering. In: Proceedings of EMNLP, pp 645–650
26. Yao X, Van Durme B (2014) Information extraction over structured data: question answering with freebase. In: Proceedings of ACL, pp 956–966
27. Yih W, He X, Meek C (2014) Semantic parsing for single-relation question answering. In: Proceedings of ACL, pp 643–648
28. Yin J, Zhao WX, Li X-M (2017) Type-aware question answering over knowledge base with attention-based tree-structured neural networks. J Comput Sci Technol 32(4):805–813
29. Yin W, Yu M, Xiang B, Zhou B, Schütze H (2016) Simple question answering by attentive convolutional neural network. In: Proceedings of COLING, pp 1746–1756
30. Yu M, Yin W, Hasan KS, Santos CDN, Xiang B, Zhou B (2017) Improved neural relation detection for knowledge base question answering. In: Proceedings of ACL, pp 571–581
31. Zhang Y, Dai H, Kozareva Z, Smola A, Song L (2018) Variational reasoning for question answering with knowledge graph. In: Proceedings of AAAI, pp 6069–6076