Answering Complex Questions on Knowledge Graphs

Xin Wang[1,2], Min Luo[2], Chengliang Si[2(✉)], and Huayi Zhan[2]

[1] Southwest Petroleum University, Chengdu, China
[2] Sichuan Changhong Electronic Ltd. Co, Mianyang, China
{chengliang.si,Huayi.zhan}@changhong.com

Abstract. The topic of knowledge-based question answering (KBQA) has attracted wide attention for a long period. A series of techniques have been developed, especially for *simple questions*. To answer *complex questions*, most existing approaches apply a semantic parsing-based strategy that parses a question into a query graph for result identification. However, due to poor quality, query graphs often lead to incorrect answers. To tackle the issue, we propose a comprehensive approach for query graph generation, based on two novel models. One leverages attention mechanism with richer information from knowledge base, for core path generation and the other one employs a memory-based network for constraints selection. The experimental results show that our approach outperforms existing methods on typical benchmark datasets.

Keywords: Complex questions · Knowledge base · Question answering · Attention · Memory network

1 Introduction

Question answering on knowledge bases is an active area since proposed. Its main task is to answer natural language questions from a structured knowledge base. The KBQA system has been used as QA machine in many fields. At its early stage, investigators have proposed techniques for answering *simple questions*.

Driven by practical requirements, techniques for answering *complex questions*, which contain multiple constraints are in urgent demand. However, prior techniques for answering *simple questions* can not be easily adapted for answering *complex questions*. Taking the question **"who plays claire in lost?"** from WebQuestion [4] as an example, if one simply parses it into a triple query ⟨**claire**, **is_played_by**, **?**⟩, the answer will be anyone who played claire in any series. However, the correct answer should be the actor in "lost". Therefore, answering *complex questions* is far more complicated than that of *simple questions*.

To answer *complex questions*, most semantic-parsing-based approaches parse a question into a *query graph* and use that *query graph* to find answers from a KB. Typically, a query graph is defined as a formal graph, whose nodes correspond to the topic entity, answer node, variables and constraints, and edges correspond

© The Author(s), under exclusive license to Springer Nature Switzerland AG 2022
G. Memmi et al. (Eds.): KSEM 2022, LNAI 13368, pp. 187–200, 2022.
https://doi.org/10.1007/978-3-031-10983-6_15

to relations [27]. Figure 1 shows two questions along with their query graphs. In each figure, the rectangle with bold font denotes the topic entity, rectangles with normal font denote constraints and rounded rectangle with dotted line includes a core path. The construction of query graphs can be split into three main tasks, *i.e.,* entity linking, core paths generation and constraints selection [3,8,10,15,17,27,28,30]. Due to semantic diversity, query graphs are often unable to precisely express the meaning of a given question.

The low quality of query graphs mainly lies in two issues. The first issue is the core path generation. For example, question $\mathbf{Q}_1$ in Fig. 1 has two candidate core paths, *i.e.,* ⟨fiction_character⟩[1] and ⟨TV_character⟩[2]. It is uneasy to determine which one is the correct one. While, if the *type* information, *e.g.,* "TV Character", of the topic entity can be used, then the candidate selection becomes much easier. Another issue lies in the constraints selection. Most existing works [3,17,27,30] simply add all identified constraints to a query graph without careful justification. Hence, a query graph may carry inappropriate constraints.

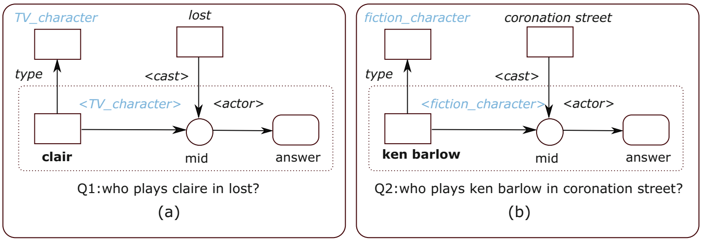

Fig. 1. Questions and their query graphs

Contributions. The contributions of this paper are summarized as follows.

1. We devise a novel metric that is based on *local similarity* and *global similarity* for measuring semantic similarity between a mention and a candidate entity. This new metric shows superiority than its counterparts.
2. We introduce an attention-based model for core path generation. In contrast to prior works, this model incorporates richer implicit information for path identification and shows excellent performance.
3. We propose a memory-based model for constraints selection. The model enables us to choose suitable constraints adaptively.
4. We compared our approach with the state-of-art methods on typical benchmark datasets and show promising results.

The reminder of the paper is organized as follows. We introduce related works in Sect. 2. We illustrate our method in Sect. 3. We conduct experiments to show the performance of our approach in Sect. 4, followed by conclusion in Sect. 5.

[1] ⟨tv.tv_character.appeared_in_tv_program,tv.regular_tv_appearance.actor⟩.
[2] ⟨fictional_universe.fictional_character.character_created_by⟩.

2 Related Work

The problem of answering complex questions on KBs has been investigated for a period. Investigators have developed several techniques, that can be categorized into *information retrieval* (abbr. IR) and *semantic parsing* (abbr. SP).

Information Retrieval. Methods in this line mainly work as follows: (a) identify the topic entity from the underlying KB for the given question; (b) gain a subgraph around the topic entity over KB; (c) extract features of candidate answers from the subgraph; and (d) match the feature of the query to choose answers.

One bottleneck of IR-based methods is feature extraction. Hence, existing approaches adopt different feature extraction methods to catch important information from the question and the answer, respectively. Early approaches perform as follows. [7,26] first gain candidate answers over KB according to dependency tree, then use feature map to process feature extraction on the question and candidate answers respectively. With the development of neural networks, [11,14] introduce techniques by incorporating CNN and attention mechanism in feature extraction. Moreover, some approaches [9,16] adopted memory networks for the task.

Since IR-based approaches retrieve answers from a KB, the completeness of the KB is imperative. [13,20] make efforts on reasoning under incomplete KB. While [22,29] proposed techniques to dynamically update reasoning instructions.

Semantic Parsing. Most semantic parsing approaches show better performance on complex KBQA [8,9,15]. Methods in this line mainly work as follows: (a) question understanding: understand a given question via semantic and syntactic analysis; (b) logical parsing: translate the question into logical forms; (c) KB grounding: instantiate logical forms by conducting semantic alignment on underlying KB; (d) KB execution: execute the logical forms to obtain answers.

In the early stage, traditional semantic parsing methods [4,12] used templates to conduct logical parsing. Recently, for question understanding, syntax dependencies [1,2,17] were introduced to accomplish the task, while [21] treats a complex question as the combination of several simple questions.

The logical form generation is crucial for SP-based methods, approaches, *e.g.,* [8,27] focus on the task. A staged query graph generation mechanism was introduced by [27], where each stage added some semantic components with rules. Followed by [27], [3] further investigated the staged query graph generation, and summarized the types of constraints. A hierarchical representation that based on words, questions and relation levels was proposed by [28]. [15] developed a state-transition model to generate query graphs. While [17] and [18] improved techniques for ranking query graphs. [10] analyzed historical datasets and put forward a set of structured templates for generating complex query graphs. Followed by [10], [8] proposed a generative model to generate abstract structures adaptively, and match the abstract structures against a KB to gain formal query graphs.

3 Approach

Problem Formulation. The task of complex question answering on KBs can be formulated as follows. Given a natural language question **Q** represented as a sequence of words $\mathcal{L}_Q = \{w_1, ..., w_T\}$ and a background knowledge base KB as input, it is to find a set of entities in KB as answers for the question **Q**.

Typically, SP-based techniques, which gain better performances, involves *question patterns* and *query graphs* (*a.k.a.* semantic graphs). We next formally introduce the notions (rephrased) as follows.

Question Pattern. Given a question **Q**, one can generate a new question by replacing an entity mention in **Q** with a generic symbol ⟨e⟩. Then this new question is referred to as a *question pattern* (denoted by **P**) of **Q**. For example, a *question pattern* of **"who plays claire in lost?"** is **"who plays** ⟨e⟩ **in lost?'**, where the entity mention "claire" in **Q** is replaced by ⟨e⟩.

Query Graph. Typically, a query graph Q_g is defined as a logical form in λ-calculus [27] and consists of a core path and various constraints. Specifically, a core path includes a topic entity, a middle node (not essential), an answer node and some predicates among these nodes. Constraints are categorized into four types according to their semantics, *e.g.,* entity constraints, type constraints, time constraints, and ordinary constraints. Two query graphs are shown in Fig. 1.

Overview. Our approach consists of three modules *i.e.,Topic Entity Recognition*, *Core Path Generation* and *Constraints Selection*. In a nutshell, given a question **Q** issued on a KB, the *Topic Entity Recognition* module first retrieves mentions from it and identifies a set of topic entities as candidates from the underlying KB. On the set of candidate entities, the *Core Path Generation* module identifies the core path with an attention-based model. Finally, the *Constraints Selection* module employs a memory-based model to select the most suitable constraints, which are then used to produce the query graph. Below, we illustrate the details of these modules in Sects. 3.1, 3.2 and 3.3, respectively.

3.1 Topic Entity Recognition

The *Topic Entity Recognition* module works as follows. (1) It employs, *e.g.,* BiLSTM + CRF to recognize entity mentions from a given question **Q**. (2) It performs the entity linking task with the state-of-the-art entity linking tool, *i.e.,* S-MART [24] to link a mention to entities in a KB. Note that, a mention may be linked to multiple entities in a KB. Hence, after above process, a collection of triples $\langle m, e, s_e \rangle$ are returned, where m, e and s_e represent a mention, an entity that is linked from m and a score measuring semantic similarity between m and e, respectively. (3) It determines the best topic entity according to a novel metric for measuring semantic similarity between a mention and an entity in a KB.

Similarity Metric. Given a question **Q**, a list of n mentions $[m_1, m_2, ..., m_n]$ can be extracted from it. For a mention m_i ($i \in [1, n]$), it is linked to k different entities $[e_i^1, e_i^2, \cdots, e_i^k]$ in a KB and each entity e_i^j is assigned with a value s_i^j

indicating the matching degree between m_i and e_i^j ($j \in [1, k]$). We define the largest matching degree $\hat{s}_i$ of m_i as $\hat{s}_i = \max\{s_i^1, s_i^2, \cdots, s_i^k\}$, then the entity with $\hat{s}_i$ is considered the best entity of m_i. The *local similarity* l_i^j and *global similarity* g_i^j between a mention m_i and its j-th entity e_i^j are hence defined as $l_i^j = s_i^j / \hat{s}_i$ and $g_i^j = \frac{s_i^j}{\max\{\hat{s}_1, \hat{s}_2, \cdots, \hat{s}_n\}}$, respectively. The final semantic similarity τ is defined as below.

$$\tau = \beta \cdot l_i^j + (1 - \beta) \cdot g_i^j, \tag{1}$$

where β is an adjustment coefficient tuned by users.

Based on the metric τ, the best topic entity e is determined and returned to the core path generation module for further processing.

3.2 Core Path Generation

Given the topic entity e of a question $\mathbf{Q}$, one can generate a set $\mathbf{R} = \{R_1, \cdots, R_T\}$ of *core paths* as candidates, where each R_i takes e as the initial node and consists of h hop nodes and edges of e in a KG. Then one only needs to determine the best one from multiple candidates. Here, the quality of a core path R_i *w.r.t.* $\mathbf{Q}$ indicates the semantic distance between R_i and the question pattern of $\mathbf{Q}$.

To tackle the issue, a host of techniques *e.g.*, [3,17,27] have been developed. These techniques mostly learn correlation between a question pattern and corresponding candidate core paths for the best core path determination. While, the correlation from explicit information alone is insufficient. For example, given

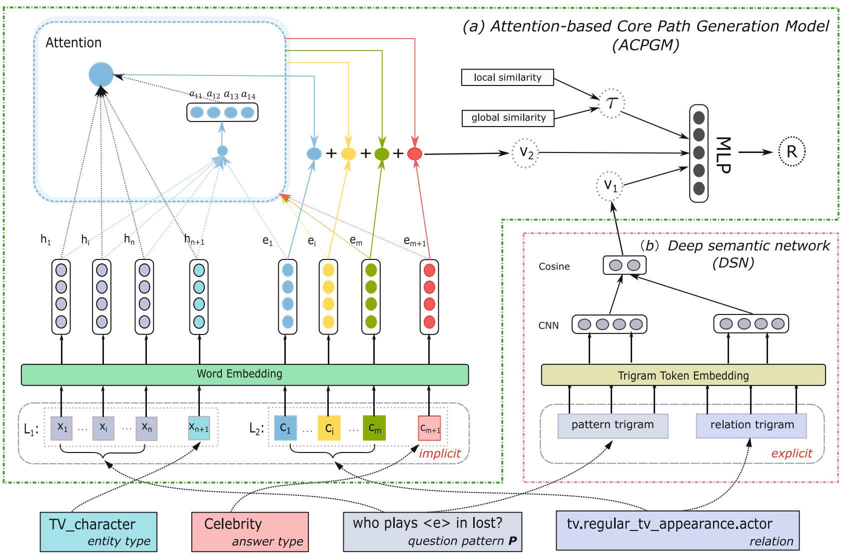

Fig. 2. Attention-based core path generation model & Deep semantic network

the question **"who plays ken barlow in coronation street?"**, it is easy to obtain a topic entity as well as a question pattern **"who plays ⟨e⟩ in coronation street?"**. Note that the pattern corresponds to multiple candidate paths (*a.k.a.* relations) in a KG, *e.g.,* ⟨fiction_character⟩ and ⟨TV_character⟩, while only one "golden" path exists. If the "type" information of the topic entity, *i.e.,* 'TV Character', can be incorporated, one can easily determine the *golden* path: ⟨TV_character⟩. This example shows that *implicit information*, *e.g., type* of the topic entity, plays a crucial role in determining the *golden* path.

Motivated by this, we develop an **A**ttention-based **C**ore **P**ath **G**eneration **M**odel, denoted as ACPGM, to generate a core path for a given question.

Model Details. As shown in Fig. 2(a), ACPGM learns correlation between a question and a core path by using *explicit correlation* v_1 and *implicit correlation* v_2 and semantic similarity τ of the topic entity. In particular, the degree of *explicit correlation* is calculated with a deep semantic network, which is shown in Fig. 2(b). We next illustrate with more details.

Explicit & Implicit Correlation. Intuitively, the semantic correlation between a question pattern $\mathbf{P}$ and a core path R_i is referred to as the *explicit correlation*. On the contrary, the correlation that is inferred from certain *implicit information*, is referred to as *implicit correlation*. Taking the question pattern **"who plays** ⟨e⟩ **in lost?"** of $\mathbf{Q}_1$ as example, it corresponds to multiple candidate paths, among which ⟨fictional_character⟩ and ⟨TV_character⟩ are typical ones. Observe that the "type" (as one kind of implicit information) of the topic entity is "TV Character", which already appeared in the second candidate. If the "type" information can be used to guide the selection, then the correct answer can be obtained since the second candidate is more preferred.

Structure of the Model. The entire model consists of two parts for learning *explicit* correlation and core paths selection, respectively.

(I) A deep semantic network (abbr. DSN) is developed to capture the *explicit* correlation between a question pattern $\mathbf{P}$ and a candidate path R_i. The structure of DSN is shown in Fig. 2 (b). It encodes $\mathbf{P}$ and R_i as letter-trigram vectors, and applies two convolutional neural networks with the same structure to learn their semantic embeddings, respectively. Then DSN computes semantic similarity (denoted by v_1) between $\mathbf{P}$ and R_i by calculating the cosine distance of two embeddings.

(II) ACPGM takes two lists L_1 and L_2 as input and leverages attention scheme to improve performance. The first list L_1 consists of a question pattern $\mathbf{P}$ and the "type" information of the topic entity in $\mathbf{P}$. The second list L_2 essentially corresponds to the implicit information, *e.g.,* the first two and the last entry of a candidate path, along with the "type" information of the answer entity. Each word x_i (resp. c_i) in the first (resp. second) list is encoded as h_i (resp. e_j), via certain encoding techniques, *e.g.,* Word2Vec.

For each h_i in L_1 and each e_j in L_2, we calculate a weight w_{ij} of each pair $\langle h_i, e_j \rangle$, as attention to the answer with below function.

$$w_{ij} = W^T(h_i \cdot e_j) + b, \tag{2}$$

where $\cdot$ operates as the dot product, $W^T \in \mathbb{R}$ and b are an intermediate matrix and an offset value, respectively; they can be randomly initialized and updated during training. Note that there are in total $l = |L_1| * |L_2|$ pairs of weights for each input. Accordingly, the attention weight a_{ij} of the pair $\langle h_i, e_j \rangle$, in terms of the implicit information can be computed via below function.

$$a_{ij} = \frac{exp(w_{ij})}{\sum_{k=1}^{l} exp(w_{ik})} \tag{3}$$

The attention weights are then employed to compute a weighted sum for each word, leading to a semantic vector that represents the question pattern. The similarity score v_2 between a question $\mathbf{Q}$ and a core path R_i is defined as below.

$$q_i = \sum_{j=1}^{n+1} a_{ij} h_j \tag{4}$$

$$v_2 = \sum_{i=1}^{m+1} q_i e_i \tag{5}$$

By now, the *explicit* and *implicit* correlation between a question and a candidate core path, *i.e.,* v_1 and v_2 are obtained. Together with score τ for measuring topic entity, a simple multi-layer perceptron, denoted by $f(\cdot)$ is incorporated to calculate the final score that is used for core paths ranking.

$$v = f(v_1, v_2, \tau) \tag{6}$$

Here, the loss function of ACPGM is defined as follows.

$$Loss(\mathbf{Q}, R) = max(\gamma + v(\mathbf{Q}, a_0) - v(\mathbf{Q}, a), 0), \tag{7}$$

where γ is the margin parameter, a and a_0 refer to the positive and negative samples, respectively. The intuition of the training strategy is to ensure the score of positive question-answer pairs to be higher than negative ones with a margin.

3.3 Constraints Selection

Given a core path, it is still necessary to enrich it with constraints imposed by the given question and produce a query graph for seeking the final answer. To this end, we first categorize constraints into four types, *i.e., entity constraint*, *type constraint*, *time constraint* and *ordinary constraint*, along the same line as [17], and then apply an approach, that works in two stages, for constraints selection.

Candidates Identification. As the first stage, this task targets at collecting valid constraints as candidates. Given a question $\mathbf{Q}$ and its corresponding core path p_c (identified by ACPGM), we first identify 1-hop entities e_c connected to nodes (excluding the topic entity and answer node) of p_c from the underlying KB. For each identified entity e_c, if part of its name appears in the original question $\mathbf{Q}$, e_c

is considered as a possible *entity constraint*. Moreover, we treat type information associated to the answer node as *type constraint*. The *time constraint* is recognized via certain regular expression, and *ordinary constraint* is distinguished via some typical hand-weaved rules. All the identified constraints are then included in a set of size n for further process.

Constraints Selection. Given the set of candidate constraints, a **M**emory-based **C**onstratint **S**election **M**odel, denoted as MCSM, is developed to choose constraints. The idea of MCSM is to measure the relevance between a question and its constraints and then select the most suitable ones to update the core path.

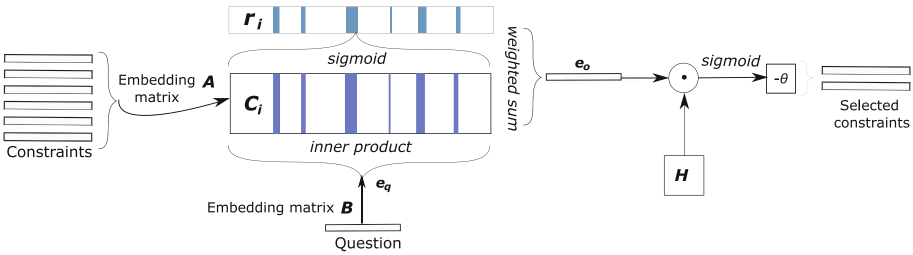

Fig. 3. Constraint selection model

It works as follows. For each candidate constraint, it is converted into a vector c_i ($i \in [1, n]$) of dimension d with an embedding matrix $A \in \mathbb{R}^{d\times n}$, which is initialized randomly and learned during training. The set of embedding $\mathbf{C} = \{c_1, c_2, \cdots, c_n\}$ will be stored in memory for querying. The input question is also encoded as a matrix e_q of dimension d with another matrix $B \in \mathbb{R}^{d\times n}$. Via encoding, the relevance r_i between a candidate constraint and the question can be measured with Eq. 8.

$$r_i = \sigma(\mathbf{e}_q \cdot c_i) \tag{8}$$

Here σ indicates a nonlinear transformation with sigmoid function. In this way, r_i can be deemed as the relevance degree between the i-th constraint and the question. Based on the relevance set, a new embedding $\mathbf{e}_o$ as the representation of the output memory can be generated as follows:

$$\mathbf{e}_o = \sum_{i=1}^{n} c_i \times r_i \tag{9}$$

Intuitively, $\mathbf{e}_o$ captures the total relevance between a question and all its constraints. Finally, the output embedding $\mathbf{e}_o$ is transformed via Eq. 10 to be a numerical number val for judgement.

$$val = \sigma(H^T \cdot \mathbf{e}_o) - \theta \tag{10}$$

Here, matrix $H \in \mathbb{R}^{n \times d}$ is randomly initialized and optimized via training, and θ is a predefined threshold for constraint selection, *i.e.,* a constraint with *val* above zero can be accepted.

After constraints are determined, the core path p_c is updated as follows. For an *entity constraint*, it is associated with a node (excluding the topic entity and answer node) in p_c. For a *type constraint*, it is connected to the answer node directly. If the constraint is a *time constraint* or *ordinary constraint*, p_c is extended by connecting the constraint to the middle node or answer node. After above steps, the query graph is generated. Then the final answer can be easily obtained.

4 Experimental Studies

4.1 Settings

We introduce details of model implementation, dataset and baseline methods.

Model Implementation. Our models were implemented in Keras v2.2.5 with CUDA 9.0 running on an NVIDIA Tesla P100 GPU.

Knowledge Base. In this work, we use Freebase [6], which contains 5,323 predicates, 46 million entities and 2.6 billion facts, as our knowledge base. We host Freebase with the Virtuoso engine to search for entities and relations.

Questions. We adopted following question sets for performance evaluation.

– CompQuestion [3] contains 2,100 questions collected from Bing search query log. It is split into 1,300 and 800 for training and testing, respectively.
– WebQuestion [4] contains 5,810 questions collected from Google Suggest API and is split into 3,778 training and 2,032 testing QA pairs.

Baseline Methods. We compared our approach with following baseline methods: (1) Yih *et al.* 2015 [27], a technique for staged query graph generation; (2) Bao *et al.* 2016 [3], extended from Yih's work with techniques for constraint detection and binding; (3) Luo *et al.*, 2018 [17], that treats a query graph as semantic components and introduces a method to improve the ranking model; (4) Hu *et al.* 2018 [15] which gains the best performance on WebQuestion with a state-transition mechanism; and (5) Chen *et al.* 2020 [8], that achieves the best performance on CompQuestion.

4.2 Model Training

We next introduce training strategies used by our models.

Training for DSM. The DSM is fed with a collection of question pattern and core path pairs for training. To improve performance, a subtle strategy is applied to ensure the correctness of positive samples as much as possible. Specifically, for each question $\mathbf{Q}$ in the original training set, a core path that takes topic entity

e in $\mathbf{Q}$ as starting node, with length no more than two in the underlying KB and $F1$ score no less than 0.5 is chosen as training data. Here $F1$ is defined as $\frac{2*\mathsf{precision}*\mathsf{recall}}{\mathsf{precision}+\mathsf{recall}}$, where $\mathsf{precision} = \frac{\#\mathsf{true\_paths\_predicted}}{\#\mathsf{predicted\_paths}}$ and $\mathsf{recall} = \frac{\#\mathsf{true\_paths\_predicted}}{\#\mathsf{true\_paths}}$.

Training for ACPGM. To train ACPGM, a collection of question pattern and core path pairs $\langle \mathbf{P}, R \rangle$ needs to be prepared. Besides the set of paths identified for training DSM, n incorrect paths were chosen randomly as negative samples, for each question pattern $\mathbf{P}$; In this work, margin γ is set as 0.1, n is set as 5 and h, indicating number of hops, is set as 3.

Training for MCSM. The constraint with the largest $F1$ score is deemed as the *golden* constraint. As a multi-classification model, MCSM calculates the binary cross entropy loss for each class label, and sum all the label loss as the final loss. The probability threshold θ is set to 0.8.

During the training, we use a minibatch stochastic gradient descent to minimize the pairwise training loss. The minibatch size is set to 256. The initial learning rate is set to 0.01 and gradually decays. We use wiki answer vectors as the initial word-level embedding.

4.3 Results and Analysis

We use the $F1$ score over all questions as evaluation metric. Here, $F1$ score for final answer is defined similarly as before, with the exception that the precision and recall are defined on answer entities rather than core paths.

Overall Performance. Table 1 shows the results on two benchmark datasets. As can be seen, our method achieves 53.2% and 42.6% $F1$ values on WebQuestion and CompQuestion, respectively, indicating that our method excels most of the state-of-the-art works, *w.r.t.* $F1$ score.

Table 1. Performance evaluation ($F1$)

Methods	CompQuestion (%)	WebQuestion (%)
[Dong *et al.*, 2015] [11]	–	40.8
[Yao *et al.*, 2015] [25]	–	44.3
[Berant *et al.*, 2015] [5]	–	49.7
[Yih *et al.*, 2015] [27]	36.9	52.5
[Bao *et al.*, 2016] [3]	40.9	52.4
[Xu *et al.*, 2016] [23]	36.9	52.5
[Abujabal *et al.*, 2017] [2]	–	51.0
[Luo *et al.*, 2018] [17]	42.8	52.7
[Hu *et al.*, 2018] [15]	–	53.6
[Zhu *et al.*, 2020] [30]	–	52.1
Ours	42.6	53.2

We noticed that Chen *et al.*, 2020[8] claimed higher performance on both dataset. However, this work introduced additional information, proposed in [17] as ground truth during training, while we just use the original WebQuestion. Moreover, Hu *et al.*, 2020[15] introduced external knowledge to solve implicit relation to achieve higher performance on WebQuestion.

Performance of Sub-modules. As our approach works in a pipelined manner, the overall performance is influenced by the sub-modules. It is hence necessary to show the performance of each sub-module. To this end, we introduce a metric, which is defined as $\mathsf{Acc}_u = \frac{|\mathsf{TP}|}{|\mathsf{test}|}$, where TP is a set consisting of intermediate results returned by a sub-module, such that each of them can imply a correct answer, and test refers to the test set. Intuitively, the metric Acc_u is used to show the upper bound of the prediction capability of each sub-module. Besides Acc_u, the metric $F1$ is also applied for performance evaluation. Table 2 shows Acc_u and $F1$ scores of sub-modules on WebQuestion and CompQuestion, respectively.

Table 2. Performance evaluation *w.r.t.* sub-modules

Sub-modules	WebQuestion		CompQuestion	
	Acc_u (%)	$F1$ (%)	Acc_u (%)	$F1$ (%)
Topic entity recognition	93.5	77.4	93.4	64.4
Core path generation	65.0	54.7	60.3	44.3
Constraints selection	56.8	53.2	52.1	42.6

Topic Entity Recognition. For an identified entity, if it can lead to the "golden" answer, it is treated as a correct entity, otherwise, it will be regarded as an incorrect prediction. Based on this assertion, the Acc_u and $F1$ score *w.r.t.* topic entity recognition can be defined. As is shown, the Acc_u of the sub-module reaches 93.5% on WebQuestion and 93.4% on CompQuestion. This shows that our sub-module for topic entity recognition performs pretty well. However, the $F1$ scores of the sub-module reaches 77.4% and 64.4% on WebQuestion and CompQuestion, respectively. The gap between the Acc_u and $F1$ scores partly lies in the incompleteness of the answer set of a question in the test set. Taking WebQuestion as an example, it was constructed by Amazon Mechanical Turk (AMT) manually, for each question, it only corresponds to no more than ten answers.

Core Path Generation. As shown in Table 2, the Acc_u of our core path generation sub-module is 65% on WebQuestion and 60.3% on CompQuestion, respectively; while the $F1$ scores on WebQuestion and CompQuestion are 54.7% and 44.3%, respectively. The figures tell us following. (1) It is more difficult to determine a correct core path on CompQuestion than that on WebQuestion, as both Acc_u and $F1$ score on CompQuestion are lower than that on WebQuestion, which is also consistent with the observation on both dataset. (2) Both metrics of the sub-module are significantly lower than that of Compared with the sub-module for topic entity recognition,

There still exists a big gap between Acc_u and F_1 scores on two dataset. We will make further analysis in Error analysis part.

Constraints Selection. After constraints selection, the final answers are obtained. Thus the $F1$ score of the sub-module is the same as that of the entire system.

We also compared the MCSM with a variant (denoted as MemN2N) of the model introduced by [19] to show its advantages *w.r.t.* training costs. Compared with MCSM, MemN2N leverages two matrices to produce different embedding of constraints, which brings trouble for model training. As shown in Table 3, MCSM consumes less time for model training, while achieves almost the same $F1$ score, no matter how training set and validation set are split. This sufficiently shows that the MCSM is able to achieve high performance with reduced training cost.

Table 3. MCSM v.s MemN2N

Train/Validation	MemN2N		MCSM	
	Training time (s)	$F1$ (%)	Training time (s)	$F1$ (%)
4:1	2549	52.8	2031	52.8
9:1	1610	52.8	1211	52.7
All Train	2340	52.7	1657	52.8

5 Conclusion

In this paper, we proposed a comprehensive approach to answering complex questions on knowledge bases. A novel metric for measuring entity similarity was introduced and incorporated in our topic entity recognition. As another component, our attention-based core path generation model leveraged attention scheme to determine the best core paths, based on both explicit and information. A memory network with compact structure was developed for constraints selection. Extensive experiments on typical benchmark datasets show that (1) our approach outperformed most of existing methods, *w.r.t.* $F1$ score; (2) our sub-modules performed well, *i.e.,* with high $F1$ values; (3) our sub-module *e.g.,* the constraints selection model was easy to train and consumes less training time; and (4) implicit information was verified effective for determining core paths.

Acknowledgement. This work is supported by Sichuan Scientific Innovation Fund (No. 2022JDRC0009) and the National Key Research and Development Program of China (No. 2017YFA0700800).

References

1. Abujabal, A., Saha Roy, R., Yahya, M., Weikum, G.: Never-ending learning for open-domain question answering over knowledge bases. In: Proceedings of the 2018 World Wide Web Conference, pp. 1053–1062 (2018)
2. Abujabal, A., Yahya, M., Riedewald, M., Weikum, G.: Automated template generation for question answering over knowledge graphs. In: Proceedings of the 26th International Conference on World Wide Web, pp. 1191–1200 (2017)
3. Bao, J., Duan, N., Yan, Z., Zhou, M., Zhao, T.: Constraint-based question answering with knowledge graph. In: Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers, pp. 2503–2514 (2016)
4. Berant, J., Liang, P.: Semantic parsing via paraphrasing. In: Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics, pp. 1415–1425 (2014)
5. Berant, J., Liang, P.: Imitation learning of agenda-based semantic parsers. Trans. Assoc. Comput. Linguistics **3**, 545–558 (2015)
6. Bollacker, K.D., Evans, C., Paritosh, P.K., Sturge, T., Taylor, J.: Freebase: a collaboratively created graph database for structuring human knowledge. In: Proceedings of the International Conference on Management of Data, pp. 1247–1250 (2008)
7. Bordes, A., Chopra, S., Weston, J.: Question answering with subgraph embeddings. In: 2014 Conference on Empirical Methods in Natural Language Processing, EMNLP 2014, pp. 615–620 (2014)
8. Chen, Y., Li, H., Hua, Y., Qi, G.: Formal query building with query structure prediction for complex question answering over knowledge base. In: International Joint Conference on Artificial Intelligence (IJCAI) (2020)
9. Chen, Y., Wu, L., Zaki, M.J.: Bidirectional attentive memory networks for question answering over knowledge bases. In: Proceedings of NAACL-HLT, pp. 2913–2923 (2019)
10. Ding, J., Hu, W., Xu, Q., Qu, Y.: Leveraging frequent query substructures to generate formal queries for complex question answering. In: Proceedings of the Conference on Empirical Methods in Natural Language Processing and the International Joint Conference on Natural Language Processing, pp. 2614–2622 (2019)
11. Dong, L., Wei, F., Zhou, M., Xu, K.: Question answering over freebase with multi-column convolutional neural networks. In: Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing, pp. 260–269 (2015)
12. Fader, A., Zettlemoyer, L., Etzioni, O.: Open question answering over curated and extracted knowledge bases. In: Proceedings of the 20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pp. 1156–1165 (2014)
13. Han, J., Cheng, B., Wang, X.: Open domain question answering based on text enhanced knowledge graph with hyperedge infusion. In: Proceedings of the Conference on Empirical Methods in Natural Language Processing, pp. 1475–1481 (2020)
14. Hao, Y., Zhang, Y., Liu, K., He, S., Liu, Z., Wu, H., Zhao, J.: An end-to-end model for question answering over knowledge base with cross-attention combining global knowledge. In: Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics, pp. 221–231 (2017)
15. Hu, S., Zou, L., Zhang, X.: A state-transition framework to answer complex questions over knowledge base. In: Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, pp. 2098–2108 (2018)

16. Jain, S.: Question answering over knowledge base using factual memory networks. In: Proceedings of the NAACL Student Research Workshop, pp. 109–115 (2016)
17. Luo, K., Lin, F., Luo, X., Zhu, K.: Knowledge base question answering via encoding of complex query graphs. In: Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, pp. 2185–2194 (2018)
18. Suchanek, F.M., Kasneci, G., Weikum, G.: Yago: a core of semantic knowledge. In: Proceedings of the International Conference on WWW, pp. 697–706 (2007)
19. Sukhbaatar, S., Weston, J., Fergus, R., et al.: End-to-end memory networks. Advances in neural information processing systems 28 (2015)
20. Sun, H., Bedrax-Weiss, T., Cohen, W.: Pullnet: open domain question answering with iterative retrieval on knowledge bases and text. In: Proceedings of the Conference on Empirical Methods in Natural Language Processing and the International Joint Conference on Natural Language Processing, pp. 2380–2390 (2019)
21. Sun, Y., Zhang, L., Cheng, G., Qu, Y.: Sparqa: skeleton-based semantic parsing for complex questions over knowledge bases. In: Proceedings of the AAAI Conference on Artificial Intelligence, vol. 34, pp. 8952–8959 (2020)
22. Xu, K., Lai, Y., Feng, Y., Wang, Z.: Enhancing key-value memory neural networks for knowledge based question answering. In: Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, pp. 2937–2947 (2019)
23. Xu, K., Reddy, S., Feng, Y., Huang, S., Zhao, D.: Question answering on freebase via relation extraction and textual evidence. In: Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics, pp. 2326–2336 (2016)
24. Yang, Y., Chang, M.W.: S-mart: Novel tree-based structured learning algorithms applied to tweet entity linking. In: Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing, pp. 504–513 (2015)
25. Yao, X.: Lean question answering over freebase from scratch. In: Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Demonstrations, pp. 66–70 (2015)
26. Yao, X., Van Durme, B.: Information extraction over structured data: question answering with freebase. In: Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics, pp. 956–966 (2014)
27. Yih, S.W.t., Chang, M.W., He, X., Gao, J.: Semantic parsing via staged query graph generation: question answering with knowledge base (2015)
28. Yu, M., Yin, W., Hasan, K.S., dos Santos, C., Xiang, B., Zhou, B.: Improved neural relation detection for knowledge base question answering. In: Proceedings of the ACL, pp. 571–581 (2017)
29. Zhou, M., Huang, M., Zhu, X.: An interpretable reasoning network for multi-relation question answering. In: Proceedings of the 27th International Conference on Computational Linguistics, pp. 2010–2022 (2018)
30. Zhu, S., Cheng, X., Su, S.: Knowledge-based question answering by tree-to-sequence learning. Neurocomputing **372**, 64–72 (2020)