Information Sciences 523 (2020) 234–244

Contents lists available at ScienceDirect

Information Sciences

journal homepage: www.elsevier.com/locate/ins

Processing knowledge graph-based complex questions through question decomposition and recomposition

Sangjin Shin, Kyong-Ho Lee*

Department of Computer Science, Yonsei Univesity, 50 Yonsei-Ro, Seodaemun-gu, Seoul 03722, Republic of Korea

ARTICLE INFO

Article history:
Received 24 June 2019
Revised 19 February 2020
Accepted 24 February 2020
Available online 25 February 2020

Keywords:
Question answering
Complex questions
Decomposition
Constraints

ABSTRACT

Nowadays, question answering (QA) over complex questions is the most spotlighted research topic in diverse communities. Different from existing QA approaches for simple questions that require a single relation in a knowledge graph (KG) to find an answer set, QA for complex questions requires to process multiple KG relations. Complex questions often include some representations for constraints (ordinal, temporal, etc.) that restrict an answer set for the given question. Despite lots of efforts to process such questions, there are still limitations in processing constraints, multiple relations, and variables. To solve the issues, we propose a novel QA method that first decomposes an input question and then generates a correct query graph with fully complete semantics. In order to fill with lossy semantics caused by the decomposition task, the proposed method employs Bi-GRU based model. The model integrates individual compositional semantics of sub-query graphs matched to decomposed sub-questions. Experimental results show the best performance compared to the state-of-art on complex questions.

1. Introduction

The KG-QA (Question Answering over Knowledge Graph) is a series of tasks for getting an answer set (KG entities) from a given natural language question [1–6]. Most of the existing KG-QA works have focused on simple questions [7,8]. Simple question is the question that requires only a single KG relation for providing an answer set. One example is a question, “Where is santa clara university?”. An answer set for this question can be obtained by simply executing the query (Santa Clara University, containedby, ?x).

Let us see another question “What movie did tom hanks won his first oscar?”. Different from the former question, this question includes two entity mentions ‘tom hanks’ and ‘oscar’ and the representation for the ordinal constraint ‘first’. Accordingly, this question cannot be represented as a single relation. Fig. 1 illustrates a query structure for this question. To answer this type of question, it is required to process multiple relations and mathematic operations for constraints (the ordinal representation ‘first’ in this example). In detail, we firstly should find movies for which Tom Hanks won the Oscar award. As shown in Fig. 1, two named entities and two predicate sequences should be identified for this task. Once the movies from this step are found, it is required to sort them based on the release date. The first movie in the sorted ones is considered as the answer. We call this type of question as a complex question. Formally, the complex question is the question that requires to process multiple KG relations, entities and diverse constraints such as ordinal, temporal, etc.

* Corresponding author.
E-mail address: khlee89@yonsei.ac.kr (K.-H. Lee).

https://doi.org/10.1016/j.ins.2020.02.065
0020-0255/

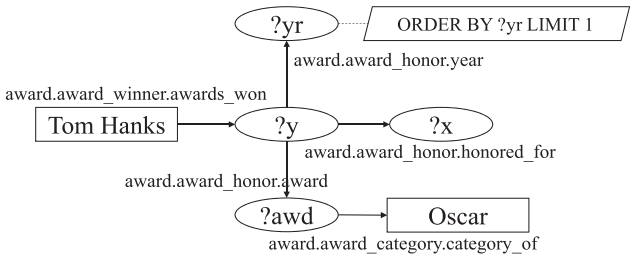

Fig. 1. Query structure for complex question

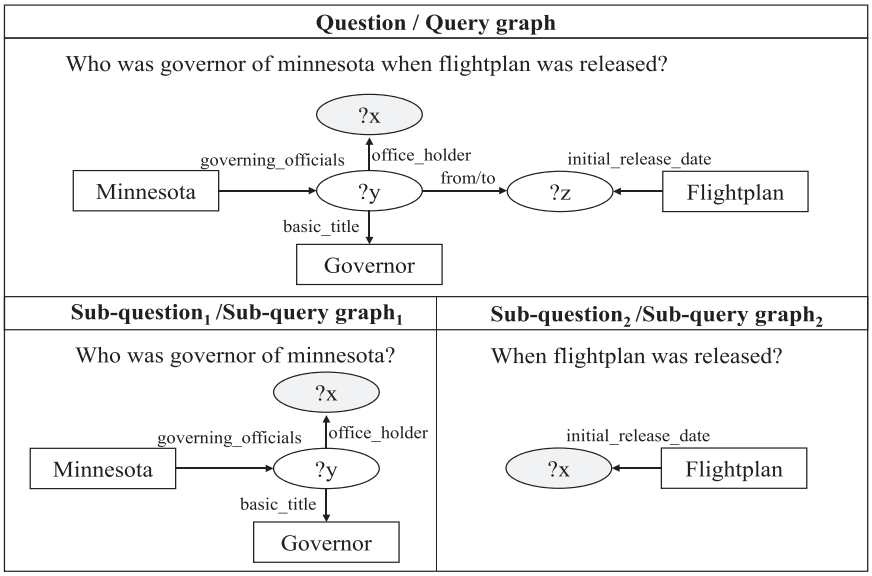

Fig. 2. One example in the question-query graph library.

Recently, several QA systems for complex questions were proposed like STAGG [9], MULCQ [10], the work of Luo et al [11]. Given a complex question, their systems collect candidate query graphs based on staged query generation methods. This task is achieved by interconnecting an answer entity, all other entities and constraints detected from the question. Since there may be ambiguous representations for entities in the question, NER such as S-MART [12] may return multiple entities per each of entity mentions. In this situation, a number of candidate query graphs can be generated. In order to capture the best intention of the given question, the systems rank candidates based on Neural Network (NN) based models [13–15]. The models are learned using training question-answer pairs of WebQuestions [16] and ComplexQuestions [10].

In the step for ranking candidate query graphs, existing works split a query graph into multiple paths [10,11]. A path is represented as a single predicate or predicate sequence (when there is a mediator object). The path structure has advantages in encoding predicates into vector representations. To calculate the similarity between a given question and each of candidate query graphs, they compare parts of the question with paths (the corresponding predicate sequences) based on the NN model. By summing up individual similarity scores, they select the candidate query graph with the best score as the correct query graph.

However, the QA systems of STAGG, MulCQ, and Luo et al. only allow multiple entity constraints and one variable (the answer node), the expressivity is quite limited. Namely, the existing complex QA systems cannot cover all types of complex questions such as the question in Fig. 2, whose query graph contains multiple predicates or predicate sequences (i.e., governing_officials + basic_title, initial_release_date) and a variable (i.e., ?z) for interconnecting two sub-query graphs. The limited expressivity also causes other problems in terms of constraint binding. Since they separate a query structure based on an answer node, all the semantics of paths are concentrated on only the answer. Although there are some researches that tried to consider an order of constraint bindings and the binding positions, they permutate the bindings in all possible way. It is not only inefficient but also inaccurate. If there are duplicated entities with the same substring to gold answers in one of their permutation, the binding can be determined as ‘correct’ irrelevant to binding’s correctness.

In order to resolve the challenges mentioned above, we propose a novel QA method based on decomposition and recomposition. We first split an input question into multiple sub-questions based on syntax-based pattern and dependency parse tree of the question, while detecting and filtering the constraints that require mathematical operations. In this decomposi-

tion task, we do not fix the splitting point as only an answer node but consider all the variable nodes (like the phrase 'an actor' in the above example). Then, we search for sub-graphs which correspond to the decomposed sub-questions from the question-query graph library learned in offline processing. At this phase, we calculate semantics for each sub-graph from a local perspective.

Next, we find the most intended combination of sub-query graphs by using our novel NN-based model for capturing compositional semantics from a global perspective. The sub-query graph combination with the highest score is chosen for the complete query graph. Finally, we extract an answer set by reflecting the captured constraints into the complete query graph and merging intermediate results. The contribution of this paper is summarized below:

- **A novel approach based on question decomposition and recomposition in order to resolve complex QA task.** Different from existing works based on question decomposition, our work is the first attempt to assemble sub-query graphs corresponding to the sub-questions semantically.
- **Improved querying capability.** we flexibly decompose a complex question to meaningful units based on individual semantics, instead of concentrating on decomposition based on the answer node.
- **BiGRU-based model for capturing comprehensive compositional semantics from a global perspective.** Benefit from the model, individual local semantics are combined to understand the intention of whole sub-questions.
- **The best performance on diverse benchmarks.** For the benchmarks such as WebQuestions [16], ComplexQuestions [10] and ComplexQuestions* [17], our method gains the best F1 scores.

2. Related work

Freebase[1]~ is a representative KG that includes enormous amount (about 2.9 billion triples) of general information encoded as RDF triples $< subject, predicate, object >$. It also supports a mediator object called a compound value type (CVT) node used to collect multiple fields of an event or a special relationship (e.g., date). This is the main difference between DBpedia[2]~ and Freebase. While Freebase represents binary and non-binary relationships with the help of CVT, DBpedia represents only binary relations. Binary relations cannot represent complex and rich information, which is extractable from questions in WebQuestions [16]. So we deal with QA approaches that target Freebase as a background KG in this paper.

Most of QA researches on RDF datasets are divided into 1) Semantic parsing and 2) Information extraction. Semantic parsing based approach [9,18,19] translates natural language-based question into a logical form such as SPARQL query [20], lambda-DCS [18] and dependency tree-like structure [21]. Typically, a semantic parsing based approach targets a linguistically motivated domain-independent meaning representation so that the computer can understand. Meanwhile, information extraction based approaches find answers for a question by exploring the knowledge graph directly. Generally, a pattern analysis, such as a regular expression, is performed. Then, they get the most appropriate candidate answers based on a ranking technique such as learning to rank [22,23]. Although the information extraction based approach is seen as a separate technique from the semantic parsing, nowadays, the boundary line between them is getting blurry with the goal of raising the performance for finding the exact meaning of natural language text. In this paper, we propose the complex QA method that takes the merits of both techniques. In order to show what we take from two techniques explicitly, we describe the template-based question-query graph matching and neural network (NN) based semantic matching between natural language terms and KG items. For easier understanding of the template-based query generation, we introduce some key works as follows:

Recently, the QA systems that use pre-defined templates to convert a natural language question into a structured query have been proposed. Authors of [24] propose manually generated templates to deal with natural language-based questions. However, such manually designed templates have limitations when covering complexly written questions. A question can be more complex, according to the user's needs. To resolve this issue, authors of [17] propose QUINT system that automatically learns utterance-query templates solely from user questions paired with their answers. In the proposed method, we employ QUINT to process sub-questions of the original complex question.

It is relatively easy to find the structure of the query graph for the given question. It can be realized by searching for pairs that are composed of a query graph and the same structured question with an input question. However, the most serious problem is how to map words in the question to KG items in the query graph. The template-based approach requires a system to enter the correct KG entity and predicate that correspond to words. Thus, entity and relation mentions should be identified correctly to find the corresponding entities and predicates. To do this, most of the existing QA systems use lexicons to cover entity and relation mentions.

With the help of lexicons, mappings between entity mentions and entities and between relation mentions and predicate can be generated. Their data is filled based on distinct supervision techniques [19,25]. However, existing lexicons are noisy due to typos and other problems as well as incomplete, resulting in incorrect answering. The author of [20] has tried to use external text data to resolve the issues. This idea has been shown effective in entity linking for web search queries.

[1] https://developers.google.com/freebase/.
[2] http://wiki.dbpedia.org/about/.

Instead of using lexicons, the authors of [9,10] use Siamese convolutional neural networks (CNN) to learn relationships between natural language phrases and KG items. Their approaches use cosine similarity between vector representations for natural language phrases and KG items to find the best appropriate mapping. In detail, they train two neural networks after encoding NL phrases and KG items to vectors (using one-hot encoding). Inputs are a sequence of vectors for the phrases on the one side of the network and a sequence of vectors for KG items on the other side of the network. They state that this continuous space representation approach has shown better results compared to lexical matching approaches.

Meanwhile, a large number of matches between natural language phrases and KG items can be generated even though a matching task is completely performed. It is due to duplicated strings in both phrases and KG items. Therefore, most of the existing approaches have tried to solve this challenge through a sophisticated ranking method and data-driven approach [22]. Here, the data-driven approach means the method which verifies whether a KG subgraph, including all the entity and predicate candidates inferred from entity and relation phrases, exists in the underlying KG or not.

All the works we mentioned have contributed a positive effect on simple questions. However, they cannot be applied to complex questions without a novel idea for modification. Consequently, a performance of QA over simple questions has been gradually enhanced so far. Apart from this, nowadays, the more challenging issue is how to process complex questions efficiently. The complex question is the one which includes a variety of constraints such as multi-entity, type, explicit/implicit temporal, ordinal, and aggregation constraints. The authors of [9–11] propose QA methods for processing complex questions, which include all the constraints mentioned above. Their approaches show the best performance in the current benchmarks. However, to the best of our knowledge, their approach does not determine a processing order of the sub-queries where constraints are bound. In the ranking phase, their systems permutate all possible orders for processing sub-queries. Such duplicated computations not only raise the number of candidate queries but also lower the accuracy of the QA. QUINT system [17] also deals with complex questions, which include two kinds of constraints (i.e., multi-entity and type). Although the authors of QUINT have tried to cover compound questions using some heuristics, their approach originally does not target complex questions with diverse constraints. TAQA [26] is the system that answers complex questions with rich semantic constraints. However, its target KB is open KB, which stores assertions (i.e., a tuple composed of a subject; relation phrase; argument) instead of RDF triples. Since there is no sophisticated schema in open KB, overall processes are based on string matching in a brute-force way.

3. Overview

The proposed method performs various tasks in two phases: Offline processing and Online processing. The offline processing focuses on generating the materials to be used at the online processing. The two key tasks are like below:

- Collecting all possible query graphs as well as sub-query graphs for sub-questions, from the training question/answer pairs.
- Designing a semantic matching model that is used to find the most intended semantics from a global perspective.

First, we find correct query graphs for all the questions from the training question-answer pairs. Then, we split questions and query graphs into sub-questions and sub-query graphs, respectively. After that, we store them into question-query graph library. The library includes pairs for an original question and its query graphs, and pairs for sub-questions of the original question and their sub-query graphs, as shown in Fig. 2. Second, we design an NN-based semantic matching model. The model yields the value for co-relation among sub-query graphs, namely the global semantics. We consider that the frequently co-used sub-query graphs have a strong relationship compared to others. Thus, they have a higher possibility of being a complete query graph.

In online processing, the proposed method performs all the tasks for answering a given question. The key tasks are as follows:

- Simplifying and decomposing a given question to make it compatible with the sub-questions stored in the question-query graph library.
- Searching for sub-query graphs that satisfy local semantics and generating combinations of sub-query graphs corresponding to each of sub-questions.
- Ranking combinations of sub-query graphs based on the NN-based semantic matching model and binding constraints to the best-intended combination of sub-query graphs.

We first simplify an input question to make it compatible with the questions and query graphs stored in the library. In this step, mathematic constraints are temporally filtered out. These constraints are reflected in the final step of the proposed method. For the simplified questions, we decompose them into sub-questions. Next, we find possible sub-query graphs corresponding to the sub-questions from the question-graph library. After that, we generate combinations of sub-query graphs for each sub-questions. Using the semantic matching model, we find the best combination with the highest similarity score. By applying constraints into proper sub-query graphs and merging the sub-query graphs based on the co-reference resolution, we compute the final answer for the given question.

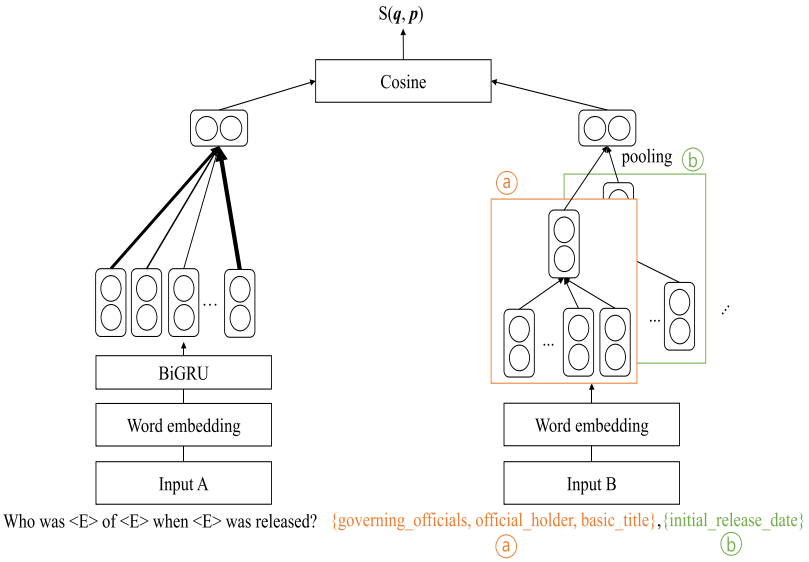

Fig. 3. The proposed semantic matching model.

4. Offline processing

We focus on generating the materials to be used at the query time, in the offline processing. Since the proposed method converts a natural language question into a query graph by combining sub-query graphs corresponding to the parts of the question, we should collect matches between a question and its query graph and between a sub-question and the corresponding sub-query graph.

Question-query graph library Construction. We collect all the possible query and sub-query graphs from the training questions of WebQuestions and ComplexQuestions. For this purpose, we directly explore KG to find the query graphs of the questions. We consider a KG sub-graph, which includes the entities posed in a question and an answer entity, as a query graph. After collecting query graphs similar to QUINT [17], we split the query graphs into smaller-sized semantic units. For this purpose, we once decompose the questions using our manually designed patterns and the existing syntax-based patterns [27,28]. Then, we also divide with query graphs in a way that reflects the semantics of the decomposed sub-questions. Here, we call such decomposed query graphs as sub-query graphs in this paper. Fig. 2 shows sub-questions and sub-query graphs of an original question and its query graph.

In the proposed method, we need to calculate a degree of relationship between an input question and a set of sub-query graphs derived from the question-query graph library. For this purpose, we design an NN-based semantic matching model. Here, we introduce how to encode questions and sub-query graphs.

Question representation. Prior to encoding question into vectors, we firstly perform NER on the question to identify entity mentions. For this, S-MART [12] system and lexicons are used. Since there is a lot of literature for constructing lexicons, we omit how to collect the contents of lexicons in this paper. When named entities are identified, we replace the entity mentions with symbols $< E >$. This task is performed to make the matching model more generic.

Given the word sequence $\{w_1, ..., w_n\}$, we use a word embedding matrix $E \in R^{|V| \times d}$ to convert the sequence into word embeddings $\{\boldsymbol{w}_1, ..., \boldsymbol{w}_n\}$. Here, $|V|$ and d denote the vocabulary size of words and the embedding dimension, respectively. Afterwards, the embeddings are fed into a bi-directional GRU network to get hidden representations $\boldsymbol{H} = \{\boldsymbol{h}_1, ..., \boldsymbol{h}_n\}$. Then, we encode a variable length sentence into a fixed size embedding. It is achieved by choosing a linear combination of the n Bi-GRU hidden vectors in $\boldsymbol{H}$. Since computing the linear combination requires the self-attention mechanism, we apply the self-attention to $\boldsymbol{H}$ similar to the work [29]. As a result, the question can be represented as $\boldsymbol{q} = \frac{1}{n} \Sigma_k \alpha_k \boldsymbol{h}_k$.

Sub-query graph representation. As we mentioned briefly, the proposed question-query graph library includes an original question, its sub-questions, KG sub-graph of the original question, and KG sub-query graphs of the sub-questions. We simply represent a sub-query graph as a set of KG predicates or predicate sequences. For example, the sub-query graph in left side of the Fig. 2 can be represented as {governing_officials, office_holder, basic_title}. Given the predicate sequence $p^i = \{p_1, ..., p_n\}$ of the i_{th} sub-query graph, we use the same word embedding matrix E to convert the sequence into word embeddings $\{\boldsymbol{p}_1, ..., \boldsymbol{p}_n\}$. Then, we represent the predicate sequence using word averaging: $\boldsymbol{p}^i = \frac{1}{n} \Sigma_k \boldsymbol{p}_k$.

Using above representations, we learn the proposed network using the pairs $(\boldsymbol{q}, \{\boldsymbol{p}^1, ..., \boldsymbol{p}^l\})$ of the question-query graph library. In the online processing time, we compute association score between an input question and a set of sub-query graph derived from question-query graph library. Fig. 3 shows a network structure.

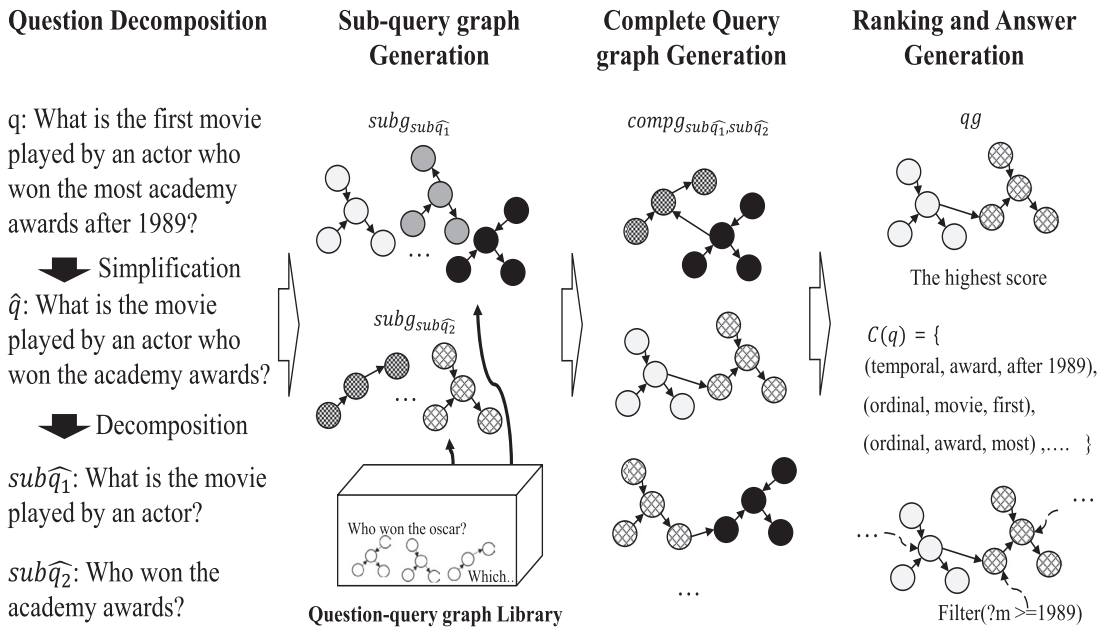

Fig. 4. The mechanism of the online processing.

5. Online processing

The online processing focuses on the processes for answering the given question. In a nutshell, we perform the following 4 steps: Question decomposition, Sub-query graph generation, Complete query graph generation, Ranking and Answer generation.

In this section, we perform a series of tasks for answering a given question q as follows. First, we simplify the q and then decompose it to multiple sub-questions $sub\hat{q}$s. Second, we search for the KG sub-query graphs $subg\hat{q}$s corresponding to $sub\hat{q}$s from the question-query graph library L, which is learned at the offline processing. Third, we generate combinations of sub-query graphs. They are called as complete query graphs $comp\hat{q}_{sub\hat{q}_1,sub\hat{q}_2,\ldots,sub\hat{q}_m}$s in this paper. Fourth, we rank them to identify a correct query graph qg that reflects the best intention. Finally, we compute an answer set A by applying constraints $C(q)$ into the query graph qg. Fig. 4 illustrates the mechanism of the proposed method.

5.1. Question decomposition

As mentioned previously, a complex question often is composed of multiple sub-questions. Also, it contains a variety of constraints. In the proposed method, we delay the binding process of mathematical constraints into the final step. Prior to decomposition task, we detect and filter out such constraints from a given question q. It is simply done based on the pre-defined patterns such as '*second*' and '*after*'+*numeric*, a dependency parse tree [30] and S-MART NER [12]. Afterwards, we store them into a constraint set $C(q) = \{(c_i, target_i, ph_i)\}$ where $0 \leq i \leq \|N\|$. The $\|N\|$ indicates the number of tokens of the question q. Also, the c_i, $target_i$ and ph_i denote the i_{th} constraint type, the target where the constraint is applied, and the phrase which represents the constraint c_i, respectively. When all the constraints are pruned out, we decompose the simplified question $\hat{q}$ into multiple sub-questions like: $(\hat{q}, \{sub\hat{q}_1, sub\hat{q}_2, \ldots, sub\hat{q}_m\})$ based on our manually designed patterns and the syntax-based patterns [27,28]. Fig. 5 shows one of manually designed patterns. When such a relative clause is detected based on the our rule, we can decompose a question into multiple sub-questions.

For easier understanding, let us see the example question q "what is the first movie played by an actor who won the most academy awards after 1989?". This example question is manually designed to verify if the proposed method has the ability to resolve the challenges. In fact, WebQuestions and ComplexQuestions do not include the questions which require to process both ordinal constraints and multiple sub-questions (which require sequential processing). As shown in the Fig. 4, the phrases 'first', 'most' and 'after 1989' are filtered out for question simplification. Then, they are kept in the constraint set, such as $(temporal, Academy\ awards, after\ 1989) \in C(q)$ until the final step of the proposed method. As a result, the question is simplified as "what is the movie played by an actor who won academy awards?". Next, the simplified question $\hat{q}$ is decomposed into two sub-questions $sub\hat{q}_1$ "what is the movie played by an actor?" and $sub\hat{q}_2$ "who won academy awards?".

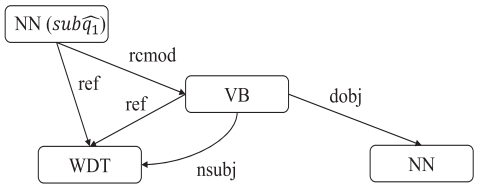

Fig. 5. One example pattern for question decomposition.

5.2. Sub-query graph generation

For each sub-question $sub\hat{q}_i$, we find appropriate sub-query graphs $\{subg_{sub\hat{q}_i}\}$ from the question-query graph library L. Following the works [31,32], we adopt an unsupervised semantic similarity function. The similarity function is composed of two sub-functions: question likelihood based on a language model and word embedding-based similarity based on *word2vec*.

For a $sub\hat{q}_i$ and sub-questions stored in the question-query graph library, the first sub-function (1) computes question likelihood as follows:

$$LM(sub\hat{q}_i, sub\hat{q}_j^L) = \prod_{w \in sub\hat{q}_i} (1-\lambda) \cdot P_{ml}(w|sub\hat{q}_j^L) + \lambda \cdot P_{ml}(w|C)] \tag{1}$$

where P_{ml} is the maximum likelihood probability of a word w estimated from $sub\hat{q}_j^L$, and w is a unigram, bigram or trigram of $sub\hat{q}_i$ or from paths of lengths one and two in the dependency parse tree of $sub\hat{q}_i$. $P_{ml}(w|C)$ is a smoothing item calculated as the maximum likelihood of w in a corpus C of questions from the question-query graph library, and $\lambda \in [0, 1]$ is a smoothing parameter that can be seen as a trade-off between the likelihood and the smoothing item.

In the second function (2), we utilize a *word2vec* model which is pre-trained on Google News corpora [33]:

$$w2v(sub\hat{q}_i, sub\hat{q}_j^L) = \frac{1}{|P|} \sum_{(w_k, w_l) \in P} cos(w2v(w_k), w2v(w_l)) \tag{2}$$

Here, $w2v(w)$ represents *word2vec* embedding vector of w where $w_k \in sub\hat{q}_i$, $w_l \in sub\hat{q}_j^L$. P denotes a set for pairs composed of a word of $sub\hat{q}_i$ and a word of $sub\hat{q}_j^L$ whose cosine value is larger than a threshold τ.

The final similarity (3) between the given sub-question $sub\hat{q}_i$ and a sub-question in the library $sub\hat{q}_j^L$ is calculated based on a linear combination of two sub-functions (1) (2). Here, α indicate a trade-off parameter.

$$Sim(sub\hat{q}_i, sub\hat{q}_j^L) = \alpha \cdot LM(sub\hat{q}_i, sub\hat{q}_j^L) + (1-\alpha) \cdot w2v(sub\hat{q}_i, sub\hat{q}_j^L) \tag{3}$$

5.3. Complete query graph generation

In the previous subsection, we find sub-questions $sub\hat{q}^L$ that satisfy $Sim(sub\hat{q}_i, sub\hat{q}_j^L) > 0.6$ for all the sub-questions $sub\hat{q}$ of a given simplified question $\hat{q}$ from the question-query graph library L. Accordingly, it is simple to fetch KG sub-query graphs $subg_{sub\hat{q}^L}$ corresponding to each of sub-questions $sub\hat{q}$. From now on, we generate all the possible combinations of $subg_{sub\hat{q}^L}$ for each $sub\hat{q}$. In this task, we replace all the entities in $subg_{sub\hat{q}^L}$ with placeholders. Instead, we insert the identified entities of $sub\hat{q}$ into the placeholders of $subg_{sub\hat{q}^L}$. The combination task is performed based on co-reference resolution [34]. During this task, a disambiguation is automatically performed because there may be no KG sub-graph composed of inappropriate sub-query graph combination in KG.

5.4. Ranking and answer generation

We rank the complete query graphs $\{compg_1, compg_2, ..., compg_n\}$ of the question $\hat{q}$ based on the association score function (4):

$$S_{ass}(\hat{q}, compg_{\hat{q}}) = S_{ass}(\boldsymbol{q}, \boldsymbol{p}) = cos(\boldsymbol{q}, \max_i \boldsymbol{p}^i) \tag{4}$$

To train the model, we adopt hinge loss to maximize the margin between positive candidates $compg_{\hat{q}}^+$ and negative candidates $compg_{\hat{q}}^-$. From each question of training sets, we find complete query graphs $compg_{\hat{q}}$s composed of $sub\hat{q}g$s. As mentioned previously, $sub\hat{q}g_i$ includes predicate sequence $p^i = \{p_1, ..., p_n\}$. So, we can consider $compg_{\hat{q}}$ as $\boldsymbol{p} = \{\boldsymbol{p}^1, ..., \boldsymbol{p}^l\}$. We collect positive data by finding a candidate $compg_{\hat{q}}$ whose F1 value of its answer is larger than a threshold (0.1). Also,

Table 1
Ranking Features

ID	Description
1	# identified entities from S-MART
2	# identified entities from the lexicon
3-4	sum(3) and average(4) of scores of all entities from S-MART
5-6	sum(5) and average(6) of TF/IDF scores on all entities from the lexicon
7-8	sum(7) and average(8) of local semantics score $Sim(sub\widehat{q}_i, sub\widehat{q}_j^t)$
9	global semantics score $S_{ass}(\widehat{q}, compg_{\widehat{q}})$
10	# output answers

Table 2
Results on three benchmarks

Method	WebQ	CompQ	CompQ*
STAGG	52.36	37.42	-
MULCQ	52.43	41.75	-
Luo et al.	52.66	42.84	-
QUINT	51.0	-	49.2
Aqqu	49.4	-	27.8
Aqqu+	49.4	-	46.7
This work	**52.72**	**43.05**	**54.4**

we randomly collect 20 negative examples $compg_{\widehat{q}}^-$ whose F1 values are lower than the positive example $compg_{\widehat{q}}^+$. The loss function (5) can be defined below:

$$L = max\{0, \lambda - S(\boldsymbol{q}, compg_{\widehat{q}}^+) + S(\boldsymbol{q}, compg_{\widehat{q}}^-)\} \tag{5}$$

Based on the model, association scores for complete query graphs are calculated. From now on, the ranking which considers all the features including local semantics similarity, global semantic similarity (association score) and etc. comprehensively is performed in order to find the most intended query graph qg. For this, we utilize a learning-to-rank technique [17,22]. Specifically, we adopt a random forest model [35] for classification to learn a preference function between a pair of complete query graphs in a similar way to the work of [22]. In order to learn the classifier, we identify training query graphs from the 3778/1300 training questions in WebQuestions and CompexQuestions datasets. Note that we collect training query graphs for the training questions where we removed the phrases corresponding to the constraints (except for multiple-entity and type constraints). At this time, we only consider the training query graphs which satisfy F1 score $>$ 0. We assign the correct complete query graph into a positive class (e.g., the graphs which return gold answers) and the remaining query graphs into a negative class.

Table 1 shows the features considered in our feature vector $\phi(\widehat{q}, compg_{\widehat{q}})$. The feature vector used is a result of the difference between two complete query graphs $\phi(\widehat{q}, compg_{\widehat{q_1}}) - \phi(\widehat{q}, compg_{\widehat{q_2}})$. Based on the learned classifier, the complete query graphs generated from each of the questions are evaluated to find a top-ranked complete query graph. Finally, we get the complete query graph, which has the highest score. The complete query graph with the highest association score value is considered as the best-intended query graph qg in the proposed method. In this step, we bind the captured constraints $C(q)$ into nodes of qg to compute the final answer set. Based on the $C(q)$ that is composed of c_i, $target_i$ and ph_i, the constraint binding is performed. By processing constraints and binding them in an appropriate order, we get the final answer set.

6. Evaluation

We evaluate the proposed approach on the WebQuestions, ComplexQuestions, and ComplexQuestions* dataset. Also, we compare the proposed method to previous works evaluated on the three benchmarks over Freebase. As illustrated in Table 2, our method shows 52.72%, 43.05% and 54.4% as average F1 scores on WebQuestions, ComplexQuestions and ComplexQuestions*, respectively.

6.1. Experimental settings

We used two datasets to evaluate our QA approach: Freebase for the dataset as a KG, WebQuestions, ComplexQuestions, and ComplexQuestions* as question-answer sets. The POS tagger used is Stanford POS tagger based on english-left3words-distim model, and the dependency parser used is Stanford dependency parser [36].

Freebase. We used a full dump of Freebase, which consists of 2.9 billion facts on 48 million entities, as our target knowledge graph.

WebQuestions. A dataset that contains 3778 training/2032 test questions paired to sets of answers. This dataset is collected via Google Suggest service, together with the answers annotated on Amazon Mechanical Turk. Since answers to the questions were collected by crowdsourcing, they include somewhat noisy items. Several questions contain only a subset

of the correct answers as a gold answer set. This dataset contains not only question-answer pairs but also KG entities and predicates. We used the original train-set split: 3778 questions (70%) for constructing question-query graph library and 2032 questions (30%) for testing.

ComplexQuestions. A dataset that contains 1300 training / 800 test questions paired to sets of answers. This dataset is collected from WebQuestions (596), Free917 (300), and logs of a practical search engine (878). The 878 manually labeled QA pairs are multi-constraint question-answer pairs.

ComplexQuestions*. A dataset that contains 150 test questions that exhibit compositionality through multiple clauses. It has no training questions and SPARQL queries. Accordingly, the existing methods which evaluated on this benchmark utilized the training question-answer pairs of the WebQuestions.

Evaluation measures. We use an average F1 as an evaluation measure for the WebQuestions, ComplexQuestions and ComplexQuestions* benchmarks. Since they sometimes contain a subset of answers as the gold standard answer of a question, the average F1 can be the most appropriate measure. The equation for the measure is given as follows:

$$averageF1 = \frac{1}{|Q|}\Sigma_{i=1}^{|Q|}F1(ga_i, sa_i) \tag{6}$$

where $F1(ga_i, sa_i)$ (7) is computed by:

$$F1(ga_i, sa_i) = 2 * \frac{(precision(ga_i, sa_i) * recall(ga_i, sa_i))}{(precision(ga_i, sa_i) + recall(ga_i, sa_i))} \tag{7}$$

$$precision(ga_i, sa_i) = \frac{|ga_i \cap sa_i|}{|sa_i|} \tag{8}$$

$$recall(ga_i, sa_i) = \frac{|ga_i \cap sa_i|}{|ga_i|} \tag{9}$$

Here, ga_i and sa_i denote a gold answer set (an answer set defined in original dataset) and a system answer set for the i-th question q_i, respectively.

Implementation. We use GloVe [37] to initialize word embeddings with dimensions (300). The size of Bi-GRU hidden layer is also set to 300. We fix the margin λ as 0.5, and the batch size as 32.

6.2. Experimental results

We evaluated our system and compared it to the state of the art QA systems evaluated on both WebQuestions and ComplexQuestions. For the following works, we did not reproduce experimental results but reported existing results.

Aqqu. An end-to-end KBQA system that automatically translates questions to the matching SPARQL queries [22].

QUINT. A template-based QA system that automatically learns question-query templates from question-answer pairs [17].

STAGG. The semantic parser which applies a deep convolutional neural network model that matches question and predicate sequences [9].

MULCQ. A KBQA system to solve the multi-constraint question by translating it to a multi-constraint query graph. This system also applies the same convolutional neural network model [10].

The work of Luo et al. A KBQA method to resolve complex questions. This work integrates hidden vectors of various semantic components and encodes their interaction as the hidden semantics of the entire query graph [11].

First of all, our method outperformed all the compared methods on WebQuestions benchmark. The reason for this is that our method is able to process unseen (not learned) structured questions by re-utilizing the sub-questions newly generated from original questions. The test data in WebQuestions includes several numbers (about 2%) of unseen structured questions. Since the performance of existing works depends on the fixed number of training questions, it is natural that existing works have a relatively low score. Also, we could process several questions composed of the sub-questions with constraints that should be processed in appropriate processing order, in addition to simple coordinating conjunction. All of these factors influenced the improvement in performance.

Our method also outperformed all the compared methods on ComplexQuestions benchmarks. STAGG adopted CNN based predicate-relation phrase matching. Based on this feature, STAGG could significantly improve F1 score compared to the existing lexical similarity-based approaches. The significant difference between the proposed method and STAGG lies in the fact that we use a language model that computes the question likelihood, and *word2vec* model in the relation-predicate matching task. Instead of learning relationships between predicate and relation phrases, our method just compares an input question with the question stored in the library. In runtime, we find the appropriate sub-questions which have the similarity values larger than a threshold 0.6. The experimental results showed there is no significant difference in terms of the relation-predicate matching. We found that the actual improvement of accuracy was caused by a processing capability for the questions composed of multiple variables and constraint processing. STAGG deals with three kinds of constraints: an aggregation constraint, multiple-entity, and type constraints. Accordingly, it cannot be the perfect comparative study with ours.

For the MULCQ and the work of Luo et al., we saw the same phenomenon as STAGG. MULCQ permutates all the possible constraint bindings. During such permutation, correct binding can be considered as 'incorrect' and vice versa. It is due to

the fact that their NN-based models are learned in a way that raises F1 score without considering the question's structure and semantics. If there is much more training source, it will work positively. But, the manually designed rule is much more effective than the learning-based strategies on the small amount of training source. In the proposed approach, we delayed the constraint binding into the final step for computing answer sets. At the step for binding constraints into a complete query graph, we use the dependency for all terms in a question. Introducing the linguistic features into constraint binding seems simple, but it is very powerful as the experimental results showed. Another factor that contributed to the performance gain was the processing capability for the questions composed of two or more relation phrases. MULCQ and the work of Luo et al. used a limited query graph generation strategy. Their methods could not cover two or more explicit represented relation phrases.

For the methods on the ComplexQuestions* benchmarks, we also outperformed Aqqu and QUINT. The questions in ComplexQuestions* are composed of multiple clauses. QUINT only focused on sub-question answering and answer stitching. Namely, they only processed complex questions expressed through coordinating conjunction ('and'). Also, they failed to capture some sub-questions. Aqqu is not designed to handle ComplexQuestions*. Accordingly, it is natural that it shows a relatively low score. To guarantee a fair comparison, Aqqu++ manually decomposes each complex question into its constituent sub-questions and answers them using Aqqu, and runs QUINT's stitching mechanism on the answer sets of sub-questions to answer the complete question. Since QUINT has already limited capacity for processing complex structured questions, Aqqu++ also shows a low score.

While the existing methods adopted CNN or GRU to match natural language phrases to KG items, our method used the language model and *word2vec* model for capturing semantics in terms of sub-question - sub-query graph matching. And we only adopted an NN-based model for global semantics. Despite them, the proposed method showed the best F1 scores on three benchmarks. It implies that our method still has the potential to get superior accuracies on benchmarks.

6.3. Error analysis

We randomly analyzed 100 questions where the proposed method could not generate answer sets on one of the benchmarks, i.e., ComplexQuestions.

Incorrect sub-question matching(13%): Our semantic similarity function for local semantics returned below 0.6 score for some questions.

Entity linking(16%): There were no matchings between entity mentions and KG entities. e.g., tut (Tutankhamun). Also, there were duplicated matchings. e.g., kennedy (John F. Kennedy, Kennedy School, Kennedy Airport, etc.)

Miscellaneous(71%): There were some not-reasonable questions. e.g., who is the governor of hawaii now?. According to the current time, the answer can be different. There were 30 questions that include the phrase 'now' without additional DateTime. Also, there were some questions that contain the phrases 'was a teenager'. They also could not be identified explicitly.

7. Conclusion

We presented a method for answering complex questions based on question decomposition and recomposition. To the best of our knowledge, our work is the first attempt to semantically assemble sub-query graphs corresponding to the sub-questions while holding a complete compositional semantics from a global perspective. For this purpose, we design a novel NN-based semantic matching model to compute an association score between an input question and the sub-query graphs derived from the question-query graph library. We perform experiments on WebQuestions, ComplexQuestions, and ComplexQuestions*. The proposed method gains the best F1 scores (52.72%, 43.05%, 54.4%) on WebQuestions, ComplexQuestions, and ComplexQuestions*, respectively.

Authors' contribution

Sangjin Shin: Conceived and designed the analysis, Collected the data, Contributed data or analysis tools, Performed the analysis, Wrote the paper

Kyong-Ho Lee: Conceived and designed the analysis, Performed the analysis, Wrote the paper Critical revision, Other contribution Advise to the first author for overall idea

Declaration of Competing Interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgements

This research was supported by the Graduate School of YONSEI University Research Scholarship Grants in 2019 and the National Research Foundation of Korea(NRF) grant funded by the Korea government(MSIP; Ministry of Science, ICT & Future Planning) (No. NRF-2019R1A2B5B01070555).

References

[1] A. Fader, L. Zettlemoyer, O. Etzioni, Open question answering over curated and extracted knowledge bases, in: SIGKDD, 2014, pp. 1156–1165.
[2] R. Álvaro, P. Anselmo, On evaluating the contribution of validation for question answering, IEEE Trans. Knowl. Data Eng. 27 (4) (2015) 1157–1161.
[3] S. Hu, L. Zou, J.X. Yu, H. Wang, D. Zhao, Answering natural language questions by subgraph matching over knowledge graphs, IEEE Trans. Knowl. Data Eng. 30 (5) (2018) 824–837.
[4] A. Fader, L. Zettlemoyer, O. Etzioni, Paraphrase-driven learning for open question answering, in: ACL, 2013, pp. 1608–1618.
[5] W. Zheng, H. Cheng, J.X. Yu, L. Zou, K. Zhao, Interactive natural language question answering over knowledge graphs, Inf. Sci. 481 (2019) 141–159.
[6] O. Kolomiyets, M.-F. Moens, A survey on question answering technology from an information retrieval perspective, Inf. Sci. 181 (24) (2011) 5412–5434.
[7] W. Yin, M. Yu, B. Xiang, B. Zhou, H. Schütze, Simple question answering by attentive convolutional neural network, in: COLING, 2016, pp. 1746–1756.
[8] W. Yih, X. He, C. Meek, Semantic parsing for single-relation question answering, in: ACL, 2014, pp. 643–648.
[9] W. Yih, M. Chang, X. He, J. Gao, Semantic parsing via staged query graph generation: question answering with knowledge base, in: ACL, 2015, pp. 1321–1331.
[10] J. Bao, N. Duan, Z. Yan, M. Zhou, T. Zhao, Constraint-based question answering with knowledge graph, in: COLING, 2016, pp. 2503–2514.
[11] K. Luo, F. Lin, X. Luo, K. Zhu, Knowledge base question answering via encoding of complex query graphs, in: EMNLP, 2018, pp. 2185–2194.
[12] Y. Yang, M.W. Chang, S-mart: Novel tree-based structured learning algorithms applied to tweet entity linking, in: ACL, 2015, pp. 504–513.
[13] M. Iyyer, J.L. Boyd-Graber, L.M.B. Claudino, R. Socher, H.D. III, A neural network for factoid question answering over paragraphs, in: EMNLP, 2014, pp. 633–644.
[14] D. Lukovnikov, A. Fischer, J. Lehmann, S. Auer, Neural network-based question answering over knowledge graphs on word and character level, in: WWW, 2017, pp. 1211–1220.
[15] M. Iyyer, W. Yih, M. Chang, Search-based neural structured learning for sequential question answering, in: ACL, 2017, pp. 1821–1831.
[16] W. Yih, M. Richardson, C. Meek, M.W. Chang, J. Suh, The value of semantic parse labeling for knowledge base question answering, in: ACL, 2016, pp. 201–206.
[17] A. Abujabal, M. Yahya, M. Riedewald, G. Weikum, Automated template generation for question answering over knowledge graphs, in: WWW, 2017, pp. 1191–1200.
[18] J. Berant, A. Chou, R. Frostig, P. Liang, Semantic parsing on freebase from question-answer pairs, in: EMNLP, 2013, pp. 1533–1544.
[19] M.A. Hearst, Automatic acquisition of hyponyms from large text corpora, in: COLING, 1992, pp. 539–545.
[20] D. Savenkov, E. Agichtein, When a knowledge base is not enough: question answering over knowledge bases with external text data, in: SIGIR, 2016, pp. 235–244.
[21] K. Xu, Y. Feng, S. Huang, D. Zhao, Hybrid question answering over knowledge base and free text, in: COLING, 2016, pp. 2397–2407.
[22] H. Bast, E. Haussmann, More accurate question answering on freebase, in: CIKM, 2015, pp. 1431–1440.
[23] L. Du, Y. Pan, J. Ding, H. Lai, C. Huang, Egrank: An exponentiated gradient algorithm for sparse learning-to-rank, Inf. Sci. 467 (2018) 342–356.
[24] C. Unger, L. Bühmann, J. Lehmann, A.C.N. Ngomo, D. Gerber, P. Cimiano, Template-based question answering over rdf data, in: WWW, 2012, pp. 639–648.
[25] M. Mintz, S. Bills, R. Snow, D. Jurafsky, Distant supervision for relation extraction without labeled data, in: ACL, 2009, pp. 1003–1011.
[26] P. Yin, N. Duan, B. Kao, J. Bao, M. Zhou, Answering questions with complex semantic constraints on open knowledge bases, in: CIKM, 2015, pp. 1301–1310.
[27] K. Xu, S. Reddy, Y. Feng, S. Huang, D. Zhao, Question answering on freebase via relation extraction and textual evidence, in: ACL, 2016, pp. 2326–2336.
[28] J. Bao, N. Duan, M. Zhou, T. Zhao, Knowledge-based question answering as machine translation, in: ACL, 2014, pp. 967–976.
[29] Z. Lin, M. Feng, C.N. dos Santos, M. Yu, B. Xiang, B. Zhou, Y. Bengio, A structured self-attentive sentence embedding, arXiv:1703.03130(2017).
[30] D. Chen, C.D. Manning, A fast and accurate dependency parser using neural networks, in: EMNLP, 2014, pp. 740–750.
[31] K. Zhang, W. Wu, F. Wang, M. Zhou, Z. Li, Learning distributed representations of data in community question answering for question retrieval, in: WSDM, 2016, pp. 533–542.
[32] A. Abujabal, R.S. Roy, M. Yahya, G. Weikum, Never-ending learning for open-domain question answering over knowledge bases, in: WWW, 2018, pp. 1053–1062.
[33] T. Mikolov, K. Chen, G. Corrado, J. Dean, Efficient estimation of word representations in vector space, arXiv:1301.3781(2013).
[34] V.I. Spitkovsky, A.X. Chang, A cross-lingual dictionary for english wikipedia concepts, in: LREC, 2012, pp. 3168–3175.
[35] L. Breiman, Random forests, Mach. Learn. 45 (1) (2001) 5–32.
[36] J.R. Finkel, T. Grenager, C. Manning, Incorporating non-local information into information extraction systems by gibbs sampling, in: ACL, 2005, pp. 363–370.
[37] J. Pennington, R. Socher, C. Manning, Glove: Global vectors for word representation, in: EMNLP, 2014, pp. 1532–1543.