SPARQA: Skeleton-Based Semantic Parsing for Complex Questions over Knowledge Bases

Yawei Sun, Lingling Zhang, Gong Cheng, Yuzhong Qu
National Key Laboratory for Novel Software Technology, Nanjing University, China
{ywsun, llzhang}@smail.nju.edu.cn, {gcheng, yzqu}@nju.edu.cn

Abstract

Semantic parsing transforms a natural language question into a formal query over a knowledge base. Many existing methods rely on syntactic parsing like dependencies. However, the accuracy of producing such expressive formalisms is not satisfying on long complex questions. In this paper, we propose a novel skeleton grammar to represent the high-level structure of a complex question. This dedicated coarse-grained formalism with a BERT-based parsing algorithm helps to improve the accuracy of the downstream fine-grained semantic parsing. Besides, to align the structure of a question with the structure of a knowledge base, our multi-strategy method combines sentence-level and word-level semantics. Our approach shows promising performance on several datasets.

1 Introduction

Question answering over knowledge bases (KBQA) has been a popular application of NLP technologies. Many recent approaches are based on semantic parsing (Herzig and Berant 2018; Jie and Lu 2018; Labutov, Yang, and Mitchell 2018; Dong and Lapata 2018; Dong, Quirk, and Lapata 2018; Chen, Sun, and Han 2018). They transform a natural language question into a formal query, for example, a SPARQL query, which in turn is executed over a knowledge base (KB) such as Freebase to retrieve answers. State-of-the-art methods (Hao et al. 2018; Mohammed, Shi, and Lin 2018; Wang et al. 2018; Chen, Wu, and Zaki 2019) have achieved promising results on simple questions that are represented as a formal query with a single predicate, for example, "who is the *wife* of Obama?" However, difficulties are faced when processing complex questions that correspond to a formal query with multiple predicates (Talmor and Berant 2018b), for example, "what movie that Miley Cyrus *acted in* had a *director* named Tom Vaughan?" We will use this question as a running example throughout the paper.

Challenges. To understand and answer a complex question, we identify two challenges among others.

First, semantic parsing often relies on syntactic parsing like dependencies (Abujabal et al. 2017; 2018; Luo et al. 2018). Errors in syntactic parsing, which are expected for complex questions, will be propagated to the downstream semantic parsing and influence the overall performance. In our running example, as shown in Figure 2, dependency parsing misses the long-distance dependency between "movie" and "had", but generates an incorrect relation between "in" and "had".

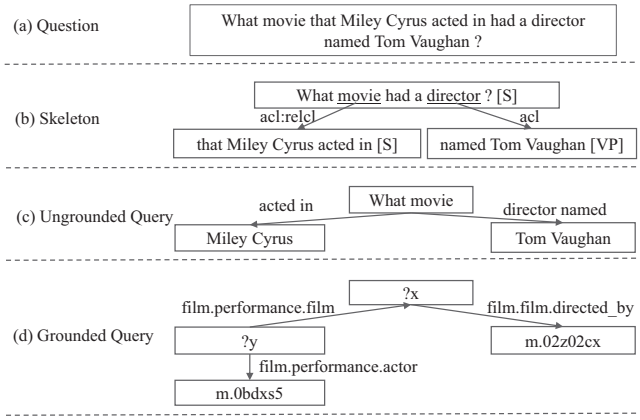

Figure 1: (a) A complex question, (b) its skeleton, (c) the derived ungrounded query and (d) grounded formal query.

Second, a question is often transformed into a KB-independent graph-structured ungrounded query (Reddy et al. 2017; Cheng et al. 2017; Hu et al. 2018), which in turn is grounded over the underlying KB into a formal query which may have a different structure. Such heterogeneity in grounding is common for complex questions with multiple predicates. In our running example, the ungrounded query in Figure 1(c) has two predicates: *acted in* and *director*, but the grounded query in Figure 1(d) has three predicates due to the use of a mediator node (`performance`) for representing n-ary relations (`actor-film-character`) in Freebase.

Contributions. We address these two challenges with a new approach called SPARQA, that is an abbreviation for *S*keleton-based semantic *PAR*sing for *Q*uestion *A*nswering. Figure 3 presents an overview of our approach. From an input question, we identify its high-level skeleton structure, to help accurate generation of an ungrounded query. This

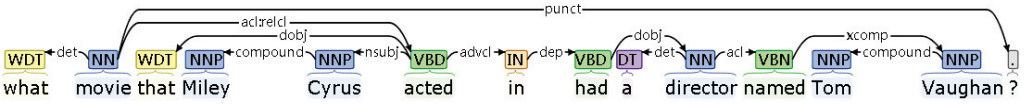

Figure 2: An example of erroneous dependencies.

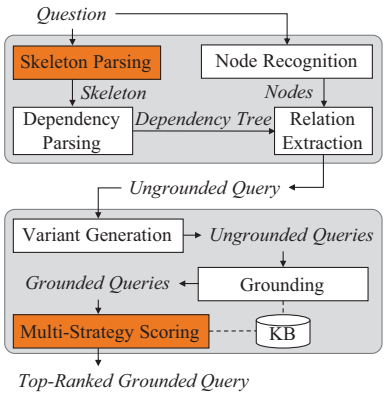

Figure 3: Overview of SPARQA.

ungrounded query and its structural variants are grounded against a KB. The generated grounded queries are ranked by a multi-strategy scorer, and the top-ranked grounded formal query is executed to retrieve answers from the KB. Our contribution in the paper is summarized as follows.

- We propose a skeleton grammar to represent the high-level structure of a complex question. This lightweight formalism and our parsing algorithm help to improve the accuracy of the downstream semantic parsing.
- To train and evaluate our algorithm for skeleton parsing, we manually annotate the skeleton structure for over 10K questions in two KBQA datasets. We make this resource public to support future research.
- We combine sentence-level and word-level scoring to rank grounded queries. The former mines and matches sentence patterns. The latter processes bags of words and trains a novel neural model to compute similarity.

The remainder of this paper is organized as follows. Section 2 presents an overview of SPARQA. Section 3 introduces skeleton parsing. Section 4 describes multi-strategy scoring. Section 5 reports experiment results. Section 6 discusses related work. Finally, we conclude in Section 7.

2 Overview of the Approach

Figure 3 presents an overview of our approach SPARQA. Our implementation is open source.[1]

To transform a question such as Figure 1(a) into a KB-independent graph-structured ungrounded query in Figure 1(c), Hu et al. (2018) propose a method named NFF (short for Node-First Framework). It assumes nodes have been recognized and extracts relations between nodes based on the dependency parse of the question. Considering the standard dependency parsing of a complex question is prone to errors, we propose skeleton grammar—a subset of the dependency grammar, to represent the high-level structure of a complex question as relations between text spans, as illustrated in Figure 1(b). The standard dependency parse of each text span are then joined to be fed into NFF for relation extraction. We will detail skeleton parsing in Section 3. For recognizing nodes, e.g, mentions of entities, classes (including WH question words), and literals, we solve it as a sequence labeling problem. We use a combination of Stanford's NER (Finkel, Grenager, and Manning 2005), SUTime (Chang and Manning 2012), and a BERT-based classifier (Devlin et al. 2019), and give priority to long mentions.

To address structural heterogeneity in grounding, we generate a set of structural variants of the original ungrounded query by contracting an edge between class nodes and/or subdividing an edge with an inserted mediator node. Each ungrounded query is grounded into a set of formal queries by linking each node to an entity, class, or literal in the KB and enumerating all possible predicates that connect adjacent nodes. For linking, we use a dictionary compiled from ClueWeb (Gabrilovich, Ringgaard, and Subramanya 2013) as well as KB-specific resources. For the Freebase KB we use its topic names and aliases. All the grounded queries are scored by a multi-strategy method, which we will detail in Section 4. The top-ranked grounded formal query such as Figure 1(d) is executed to retrieve answers from the KB.

3 Skeleton Parsing

In this section, we first introduce the skeleton grammar, and then we present our algorithm for skeleton parsing.

Skeleton Grammar

Our skeleton grammar is essentially a selected subset of the dependency grammar for specifically representing the high-level structure of a complex question. This dedicated coarse-grained representation, which is likely to feature an accurate parsing algorithm due to its simplicity, helps to improve the accuracy of the downstream fine-grained semantic parsing.

Definition 1 (Skeleton) *The skeleton of a question sentence is a directed tree where nodes representing text spans in the sentence are connected by edges representing attachment relations. Specifically, a text span is attached from a headword in another span.*

Definition 2 (Text Span) *Text span represents a phrase-level semantic unit. We consider four types of text spans in phrase structure grammars: Clause (`S`), Noun Phrase (`NP`), Verb Phrase (`VP`), and Prepositional Phrase (`PP`).*

[1] https://github.com/nju-websoft/SPARQA

Algorithm 1 Skeleton Parsing

Require: A sentence Q
Ensure: The skeleton of Q
  $T \leftarrow$ tree with a root node Q
  while Split(Q) is true **do**
    $s \leftarrow$ TextSpanPrediction(Q)
    $h \leftarrow$ HeadwordIdentification(s, Q)
    $r \leftarrow$ AttachmentRelationClassification(s, Q)
    Remove s from Q
    Grow T with relation r from $h \in Q$ to s
  end while
  return T

Note that these types are only for reader comprehension. A skeleton parser is not required to label the type of a text span.

Definition 3 (Attachment Relation) *Attachment relation represents a dependence between text spans. We consider seven types of relations that are common in standard dependency grammars: adjectival clause (`acl`), its sub-type relative clause modifier (`acl:relcl`), nominal modifier (`nmod`), its sub-type possessive alternation (`nmod:poss`), coordination (`conj`), open clausal complement (`xcomp`), and adverbial clause modifier (`advcl`).*

A skeleton has a tree structure. When we determine the granularity of our skeleton grammar and define allowed types of text spans and attachment relations, a key feature of this tree structure we want to provide is: by iteratively removing its leaf nodes, the remaining text spans in each iteration always comprise a maximal well-formed sentence, until reaching a simple sentence such that further split is not possible. This high-level structure helps to hierarchically distinguish the backbone of a complex question from other parts.

Example. Figure 1(b) shows the skeleton for our running example. The question sentence is divided into three text spans connected by two attachment relations from different headwords (underlined).

Headwords are used to join the standard dependency parse of each text span into a full dependency tree for the original question, to be fed into the downstream semantic parsing as described in Section 2. Such a two-stage dependency parsing is expected to feature improved accuracy as the skeleton parsing in the first stage adopts a lightweight formalism, and the standard dependency parsing in the second stage processes simple text spans.

Algorithm for Skeleton Parsing

Algorithm 1 parses a question sentence Q into its skeleton denoted by T. Initially, T comprises a single root node representing the entire sentence Q as a text span. Then iteratively, we grow T by splitting Q and adding a new edge. The Split procedure decides whether Q needs further split. If needed, the TextSpanPrediction procedure predicts the next text span to be split from Q, denoted by s. It will be removed from Q and attached from a headword h in the remaining Q, which is identified by the HeadwordIdentification procedure. The AttachmentRelationClassification procedure determines the attachment relation. After iterations, T is returned when Q needs no further split.

Example. In Figure 1, text span "named Tom Vaughan" is firstly split from the question and attached from its headword "director" in the remaining question with relation `acl`. Then, text span "that Miley Cyrus acted in" is split and attached from "movie" with relation `acl:relcl`. The remaining "what movie had a director" needs no further split, so skeleton parsing is completed. Note that a skeleton is generally not limited to have a star structure as in this example.

The four procedures mentioned above are implemented based on BERT (Devlin et al. 2019), where the authors illustrated fine-tuning BERT on several tasks:

- SPC: sentence pair classification,
- SSC: single sentence classification, and
- QA: question answering.

Our BERT-based implementation of the four procedures is shown in Figure 4 and detailed below. For our experiments, we manually annotated the skeleton structure for questions in the training set to fine-tune our BERT models.

Split. This procedure decides whether Q needs further split. We formulate it as an SSC task that is supported by a standard fine-tuned BERT model. We treat Q as the single sentence fed into the model, which outputs a binary value.

TextSpanPrediction. This procedure predicts the next text span s to be split from Q. We formulate it as a QA task that is supported by a standard fine-tuned BERT model. We disable the question input and treat Q as the paragraph fed into the model, which outputs a span in Q as s.

HeadwordIdentification. This procedure identifies a headword in the remaining Q from which s is attached. We formulate it as a QA task that is supported by a standard fine-tuned BERT model. We treat the remaining Q as the paragraph and treat s as the question fed into the model, which outputs a span (restricted to a single word) in the remaining Q as h.

AttachmentRelationClassification. This procedure determines the attachment relation r from h in the remaining Q to s. We formulate it as an SPC task that is supported by a standard fine-tuned BERT model. We treat s and the remaining Q as two sentences fed into the model, which outputs one of the seven predefined attachment relations as r.

4 Multi-Strategy Scoring

Candidate grounded queries are ranked by their total scores output by a sentence-level scorer and a word-level scorer, which we describe in this section.

Sentence-Level Scorer

This scorer exploits known mappings from questions to formal queries by mining and matching sentence/query patterns. The pattern of a question is obtained by replacing entity mentions with dummy tokens. For a question in the training set, the pattern of its underlying formal query is obtained by replacing entities that correspond to dummy tokens in the question with placeholders.

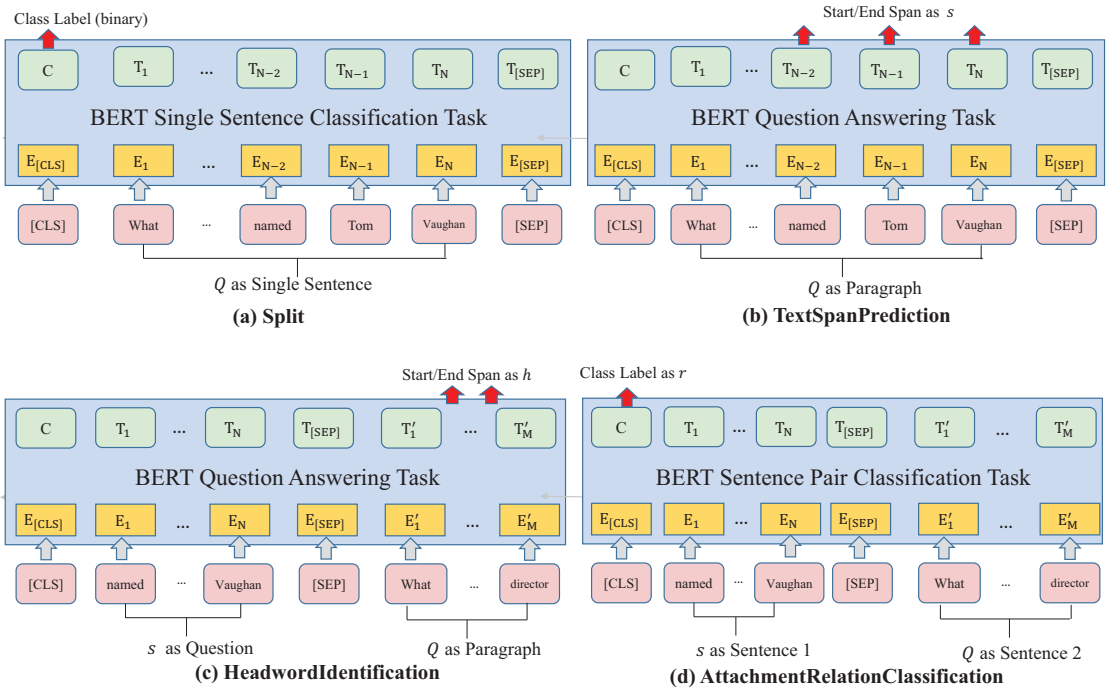

Figure 4: BERT-based implementation of skeleton parsing.

Example. The pattern of our running example is "what movie that $\langle E_1 \rangle$ acted in had a director named $\langle E_2 \rangle$?" If this is a question in the training set, the pattern of its underlying formal query is derived from Figure 1(d) by replacing "m.0bdxs5" and "m.02z02cx" with placeholders.

Given a test question, we find its most similar question in the training set that has the same number of dummy tokens in their patterns. We replace the placeholders in the pattern of this training question's underlying formal query with the corresponding entities in the test question, to generate a grounded formal query. This query will score 1.0 if it retrieves non-empty results, or 0.0 otherwise. All the other candidate grounded queries trivially score 0.0.

Similarity is computed by a standard fine-tuned BERT model that supports the SPC task, which is fed with the patterns of two sentences and outputs a numerical value representing similarity. During training, we use pairs of questions in the training set that have the same question pattern and the same query pattern as positive examples, and use other random pairs as negative examples. During testing, we use the fine-tuned model to predict the similarity between the test question and each training question.

Word-Level Scorer

This scorer is based on bag of words. We train a novel neural model shown in Figure 5 to score a grounded formal query. Specifically, we represent questions and formal queries as bags of words, where we remove concrete entities and stop

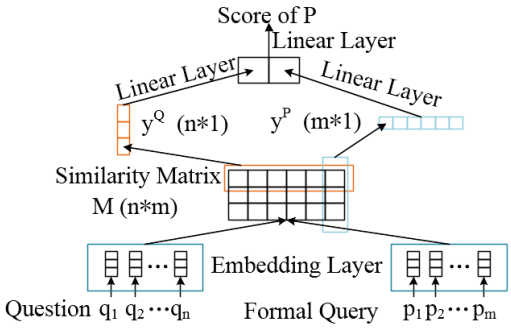

Figure 5: Word-level scorer.

words. The remaining words mainly represent predicates. Let $Q = \{q_1, \ldots, q_n\}$ be a bag of words for a question. Let $P = \{p_1, \ldots, p_m\}$ be a bag of words for a formal query. In our model, words are converted into their pre-trained GloVe embeddings (Pennington, Socher, and Manning 2014). We calculate the cosine similarity between pairs of embedding vectors in Q and P, forming an $n \times m$ similarity matrix denoted by $M = [M_{i,j}]_{n*m}$. We take the maximum value in each row and in each column, producing an n-dimensional vector and an m-dimensional vector, denoted by $\mathbf{y}^Q$ and $\mathbf{y}^P$,

respectively:

$$\mathbf{y}^Q = [y_i^Q]_n\,, \quad y_i^Q = \max_{1\le j\le m} M_{i,j}\,,$$
$$\mathbf{y}^P = [y_j^P]_m\,, \quad y_j^P = \max_{1\le i\le n} M_{i,j}\,.$$

We feed $\mathbf{y}^Q$ and $\mathbf{y}^P$ into linear layers, which in turn are fed into another linear layer to output a numerical score for the formal query.

During training, we use questions in the training set and their underlying formal queries as positive examples, and use other random combinations as negative examples. During testing, we use the learned model to predict the score of each candidate grounded formal query.

5 Experiments

In this section, we evaluate our approach SPARQA on two standard KBQA datasets with complex questions. We compare with baselines, perform an ablation study of our approach, and finally we analyze each component of our skeleton parser. Note that as our implementation relies on an existing method for fine-grained semantic parsing (Hu et al. 2018), our experiments will give more attention to the effectiveness of our proposed add-on (i.e., skeleton parsing) rather than the overall performance of our full approach.

Datasets

The experiments were performed on two public datasets involving complex questions.

- **GraphQuestions** (Su et al. 2016) contains 5,166 questions—2,558 for training and 2,608 for testing. They can be transformed into SPARQL queries over Freebase (version June 2013).
- **ComplexWebQuestions** version 1.1 (Talmor and Berant 2018b) contains 34,689 questions with a split of 80-10-10 for training, validation, and test sets. They can be transformed into SPARQL queries over Freebase (version 2015-08-09). Alternatively, they can also be answered based on provided search engine snippets.

Baselines

On GraphQuestions, we compared with six methods with reported results on this dataset.

- **SEMPRE** (Berant et al. 2013) is a bottom-up beam-based semantic parsing method.
- **PARASEMPRE** (Berant and Liang 2014) is a paraphrase-based semantic parsing method.
- **JACANA** (Yao and Durme 2014) is an information extraction based method.
- **UDEPLAMBDA** (Reddy et al. 2017) is a dependency-based semantic parsing method.
- **SCANNER** (Cheng et al. 2017) is a transition-based neural semantic parsing method.
- **PARA4QA** (Dong et al. 2017) is a paraphrase-based neural method.

On ComplexWebQuestions, we compared with five methods with reported results on this dataset. Note that this dataset also provided relevant search engine snippets.

- **MHQA-GRN** (Song et al. 2018) uses graph neural networks to answer a question as a reading comprehension task over snippets.
- **SIMPQA + PRETRAINED** (Talmor and Berant 2018b) uses a pre-trained reading comprehension model over snippets.
- **SPLITQA + PRETRAINED** (Talmor and Berant 2018a) uses the same pre-trained model as SIMPQA+PRETRAINED but it decomposes a question and then recomposes the final answer.
- **SPLITQA + data augmentation** (Talmor and Berant 2018a) uses the same method as SPLITQA + PRETRAINED but is trained on ComplexWebQuestions examples as well as an additional large set of examples that are not accessible to our approach.
- **PullNet** (Sun, Bedrax-Weiss, and Cohen 2019) uses graph convolutional networks to extract answers from KB and/or snippets.

Configuration of SPARQA

Skeleton Parser. To train and test our skeleton parser, we manually annotated the skeleton structure for all the 5,166 questions in GraphQuestions and 5,000 questions in ComplexWebQuestions: 3,000 from the training set, 1,000 from the validation set, and 1,000 from the test set. We have made this resource public to support future research.[2]

In our skeleton parser, all the four BERT models were based on $\text{BERT}_{\text{BASE}}$ (L = 12, H = 768, A = 12, total parameters = 110M). Their hyperparameters were:

- Split: max sequence length = 32, learning rate = 3e-5, batch size = 32, training epochs = 100,
- TextSpanPrediction: max sequence length = 32, learning rate = 3e-5, batch size = 32, training epochs = 100,
- HeadwordIdentification: max sequence length = 32, learning rate = 3e-5, batch size = 32, training epochs = 100,
- AttachmentRelationClassification: max sequence length = 64, learning rate = 4e-5, batch size = 32, training epochs = 100.

Sentence-Level Scorer. The BERT model was configured as follows:

- max sequence length = 64, learning rate = 3e-5, batch size = 32, training epochs = 4.

Word-Level Scorer. We used 300-dimensional pre-trained GloVe embeddings. The neural model was trained with hinge loss with negative sampling size = 300, using Adam with learning rate = 0.001 and batch size= 32.

Overall Results

Following common practice in the literature, we report F1 on GraphQuestions and report Precision@1 (P@1) on ComplexWebQuestions averaged over all the test questions.

[2] https://github.com/nju-websoft/SPARQA

	F1
SEMPRE	10.80
PARASEMPRE	12.79
JACANA	5.08
UDEPLAMBDA	17.70
SCANNER	17.02
PARA4QA	20.40
SPARQA	21.53

Table 1: Overall results on GraphQuestions.

	P@1
MHQA-GRN	30.10
SIMPQA + PRETRAINED	19.90
SPLITQA + PRETRAINED	25.90
SPLITQA + data augmentation	34.20
PullNet	45.90
SPARQA	31.57

Table 2: Overall results on ComplexWebQuestions.

The results on GraphQuestions are summarized in Table 1. Around half of the questions in this dataset are complex questions. Others are simple questions. Our SPARQA achieved a new state-of-the-art result on this dataset, outperforming all the known baseline results. F1 was improved from 20.40—the previous best result achieved by the PARA4QA, to 21.53, by an increase of 5.5%. We would like to highlight the result 17.70 achieved by UDEPLAMBDA, which also adopted dependency-based semantic parsing but relied on a set of predefined rules. It demonstrated the effectiveness of our skeleton-based semantic parsing. Moreover, our skeleton parsing did not hurt accuracy on the 1,172 simple questions in this dataset, where SPARQA (F1=27.68) was comparable with PARA4QA (F1=27.42).

The results on ComplexWebQuestions are summarized in Table 2. All the questions in this dataset are complex questions. Our SPARQA outperformed most baselines except for two that used additional data. Specifically, SPLITQA + data augmentation used additional 28,674 magic training examples that were not accessible to our approach. PullNet used not only KB but also external search engine snippets which were not used in our approach.

Ablation Study

We analyzed the usefulness of our two technical contributions: skeleton parsing and multi-strategy scoring.

Skeleton Parsing. Recall that in SPARQA, skeleton parsing is performed to generate more accurate dependencies, which are then fed into an existing method (Hu et al. 2018) to generate an ungrounded query. To explore the usefulness of skeleton parsing, we removed skeleton parsing and directly fed the dependencies generated by Stanford CoreNLP into the downstream module.

The results on ComplexWebQuestions are shown in Table 3. By excluding skeleton parsing, P@1 decreased notably from 31.57 to 29.39 by 6.9%. As all the questions in this dataset are complex questions, the result demonstrated the usefulness of our proposed skeleton parsing in improving the accuracy of dependency-based semantic parsing.

	P@1
SPARQA	31.57
SPARQA w/o skeleton parsing	29.39
SPARQA w/o sentence-level scorer	26.45
SPARQA w/o word-level scorer	26.11

Table 3: Ablation study of SPARQA.

Overall LAS	93.73
Accuracy of Split	99.42
Accuracy of TextSpanPrediction	97.17
Accuracy of HeadwordIdentification	97.22
Accuracy of AttachmentRelationClassification	99.14

Table 4: Accuracy of skeleton parsing.

Multi-Strategy Scoring. In SPARQA, two methods are implemented to score candidate formal queries: a sentence-level scorer and a word-level scorer. Their scores are combined in the end. To explore their usefulness, we removed either of them and only used the other.

The results on ComplexWebQuestions are shown in Table 3. By excluding the sentence-level scorer, P@1 decreased considerably from 31.57 to 26.45 by 16.2%. By excluding the word-level scorer, the decrease was even larger, from 31.57 to 26.11 by 17.3%. The results demonstrated the usefulness of our proposed two scorers in improving the accuracy of semantic parsing.

Accuracy of Skeleton Parsing

The above ablation study has demonstrated the usefulness of our skeleton parsing in an extrinsic manner—through the KBQA task. Below we show the results of intrinsic evaluation by comparing the output of our skeleton parser with manually annotated gold standard for the 1,000 test questions in ComplexWebQuestions.

The results are presented in Table 4. We computed LAS—Labeled Attachment Score, a commonly used metric for evaluating dependency parsing. The overall LAS of our skeleton parsing reached 93.73. We also evaluated the accuracy of each of the four components of our skeleton parser. The results were satisfyingly all above 97%. The results demonstrated the effectiveness of our BERT-based parsing algorithm. With this dedicated skeleton structure for representing complex questions and our accurate parsing algorithm, the performance of the downstream semantic parsing was therefore improved in the ablation study (Table 3).

Error Analysis

We randomly sampled 100 questions where our SPARQA achieved P@1=0 on ComplexWebQuestions. We classified the errors into the following categories.

Node Recognition and Linking (37%): It was mainly related to long mentions that were hard to recognize, for example, the entity mention “Rihanna: Live in Concert Tour” in

the question "where was the artist had a concert tour named Rihanna: Live in Concert Tour raised?"

Skeleton Parsing (5%): One typical error was in headword identification. Long-distance attachment was sometimes not found. For example, the text span "with a capital called Brussels" in the question "What country speaks Germanic languages with a capital called Brussels?" should be attached from headword "country", but was mistakenly linked to "languages" by our parser.

Ungrounded Query (10%): It was caused by the off-the-shelf method we used to generate ungrounded queries from dependencies (Hu et al. 2018).

Structural Heterogeneity (22%): For example, for the question "who is the prime minister of the country that has national anthem March Forward, Dear Mother Ethiopia", its skeleton and the derived ungrounded query were path-structured, but the correct formal query over Freebase was actually star-structured.

Scoring (15%): Our sentence-level scorer failed when the training set did not cover a test question's query pattern. Another case was more challenging. The test question and the training question had the same pattern, but corresponded to formal queries of different patterns. Our word-level scorer failed when the pattern of a question contained very few content words, for example, "what about $\langle E \rangle$?". Other errors were related to polysemy and synonyms.

Others (11%): Other errors were related to aggregate questions which we did not specifically process, as well as typos, character encoding issues, etc.

6 Related Work

Semantic Parsing has gained increasing research attention. Traditional rule-based methods are limited by their generalizability (Liang, Jordan, and Klein 2011; Berant et al. 2013; Berant and Liang 2014). Transition-based stated parsing defines states and actions to search for possible formal queries (Yih et al. 2015; Cheng et al. 2017; Hu, Zou, and Zhang 2018; Chen, Sun, and Han 2018). Recent neural methods employ encoder-decoder models to solve semantic parsing as a sequence transduction problem (Jia and Liang 2016; Gupta and Lewis 2018; Sorokin and Gurevych 2018; Dong and Lapata 2018; Dong, Quirk, and Lapata 2018; Luo et al. 2018). By comparison, dependency-based methods transform the dependency structure of a question into a formal query, thus being more explainable (Reddy et al. 2016; 2017; Abujabal et al. 2017; Hu et al. 2018). However, their performance is related to the performance of dependency parsing, which is not satisfying on complex questions. In our SPARQA, more accurate dependencies are obtained by joining local dependencies according to a global skeleton structure. The higher accuracy is attributed to the lightweight formalism of the skeleton grammar and the simplicity of the text spans where dependency parsing is performed.

Predicate Mapping is a key step in question answering over knowledge bases. Early feature-rich methods exploit lexicons and syntactic information (Yao and Van Durme 2014; Bast and Haussmann 2015). Recent neural methods encode questions and predicates for computing their similarity. They encode predicates at different granularities, for example, at the character level (Yih et al. 2015; Yin et al. 2016) or at the word level (Yu et al. 2017; Luo et al. 2018). Various models have been used, from the simple CNN (Yih et al. 2015; Yin et al. 2016) to more complex models that combine multi-level representations and similarity (Yu et al. 2017). By comparison, we focus on sentence-level and word-level semantics. Our novel implementation shows effectiveness in the ablation study.

7 Conclusion and Future Work

Our skeleton parsing has shown usefulness in processing complex questions. With our proposed skeleton structure of a question, more accurate dependencies are derived, which in turn benefit the downstream fine-grained semantic parsing. Our skeleton parser can be combined with other dependency-based KBQA methods, not limited to the one used in our implementation. It may also find application in other tasks where complex sentences are common. Besides, our simple yet effective word-level scoring model can also be used as a generic similarity measure. We will experiment with these extensions in future work.

Experiments reveal the following limitations of our work to be overcome in future work. First, node recognition and linking, though being out of the scope of our contribution, is a major weakness of our full approach. It is still an open problem in question answering research. Second, for structural heterogeneity, one may explore some graph-based methods, such as graph edit distance and graph neural network. Third, aggregate questions are not the focus of our approach, but occupy a considerable proportion in complex questions. It may be possible to learn templates for parsing such questions.

8 Acknowledgments

This work was supported in part by the National Key R&D Program of China under Grant 2018YFB1005100, and in part by the NSFC under Grant 61772264. Gong Cheng was supported by the Qing Lan Program of Jiangsu Province.

References

Abujabal, A.; Yahya, M.; Riedewald, M.; and Weikum, G. 2017. Automated template generation for question answering over knowledge graphs. In *WWW 2017*, 1191–1200.

Abujabal, A.; Saha Roy, R.; Yahya, M.; and Weikum, G. 2018. Never-ending learning for open-domain question answering over knowledge bases. In *WWW 2018*, 1053–1062.

Bast, H., and Haussmann, E. 2015. More accurate question answering on freebase. In *CIKM 2015*, 1431–1440.

Berant, J., and Liang, P. 2014. Semantic parsing via paraphrasing. In *ACL 2014, Volume 1: Long Papers*, 1415–1425.

Berant, J.; Chou, A.; Frostig, R.; and Liang, P. 2013. Semantic parsing on freebase from question-answer pairs. In *EMNLP 2013*, 1533–1544.

Chang, A. X., and Manning, C. D. 2012. Sutime: A library for recognizing and normalizing time expressions. In *LREC 2012*, 3735–3740.

Chen, B.; Sun, L.; and Han, X. 2018. Sequence-to-action: End-to-end semantic graph generation for semantic parsing. In *ACL 2018, Volume 1: Long Papers*, 766–777.

Chen, Y.; Wu, L.; and Zaki, M. J. 2019. Bidirectional attentive memory networks for question answering over knowledge bases. *CoRR* abs/1903.02188.

Cheng, J.; Reddy, S.; Saraswat, V.; and Lapata, M. 2017. Learning structured natural language representations for semantic parsing. In *ACL 2017, Volume 1*, 44–55.

Devlin, J.; Chang, M.; Lee, K.; and Toutanova, K. 2019. BERT: pre-training of deep bidirectional transformers for language understanding. In *NAACL-HLT 2019*, 4171–4186.

Dong, L., and Lapata, M. 2018. Coarse-to-fine decoding for neural semantic parsing. In *ACL 2018*, 731–742.

Dong, L.; Mallinson, J.; Reddy, S.; and Lapata, M. 2017. Learning to paraphrase for question answering. In *EMNLP 2017*, 875–886.

Dong, L.; Quirk, C.; and Lapata, M. 2018. Confidence modeling for neural semantic parsing. In *ACL 2018*, 743–753.

Finkel, J. R.; Grenager, T.; and Manning, C. D. 2005. Incorporating non-local information into information extraction systems by gibbs sampling. In *ACL 2005*, 363–370.

Gabrilovich, E.; Ringgaard, M.; and Subramanya, A. 2013. Facc1: Freebase annotation of clueweb corpora, version 1 (release date 2013-06-26, format version 1). *Note: http://lemurproject. org/clueweb09/FACC1/Cited by* 5.

Gupta, N., and Lewis, M. 2018. Neural compositional denotational semantics for question answering. In *EMNLP 2018*, 2152–2161.

Hao, Y.; Liu, H.; He, S.; Liu, K.; and Zhao, J. 2018. Pattern-revising enhanced simple question answering over knowledge bases. In *COLING 2018*, 3272–3282.

Herzig, J., and Berant, J. 2018. Decoupling structure and lexicon for zero-shot semantic parsing. In *EMNLP 2018*, 1619–1629.

Hu, S.; Zou, L.; Yu, J. X.; Wang, H.; and Zhao, D. 2018. Answering natural language questions by subgraph matching over knowledge graphs. *IEEE Trans. Knowl. Data Eng.* 30(5):824–837.

Hu, S.; Zou, L.; and Zhang, X. 2018. A state-transition framework to answer complex questions over knowledge base. In *EMNLP 2018*, 2098–2108.

Jia, R., and Liang, P. 2016. Data recombination for neural semantic parsing. In *ACL 2016, Volume 1: Long Papers*.

Jie, Z., and Lu, W. 2018. Dependency-based hybrid trees for semantic parsing. In *EMNLP 2018*, 2431–2441.

Labutov, I.; Yang, B.; and Mitchell, T. M. 2018. Learning to learn semantic parsers from natural language supervision. In *EMNLP 2018*, 1676–1690.

Liang, P.; Jordan, M. I.; and Klein, D. 2011. Learning dependency-based compositional semantics. In *ACL 2011*, 590–599.

Luo, K.; Lin, F.; Luo, X.; and Zhu, K. Q. 2018. Knowledge base question answering via encoding of complex query graphs. In *EMNLP 2018*, 2185–2194.

Mohammed, S.; Shi, P.; and Lin, J. 2018. Strong baselines for simple question answering over knowledge graphs with and without neural networks. In *NAACL 2018*, 291–296.

Pennington, J.; Socher, R.; and Manning, C. 2014. Glove: Global vectors for word representation. In *EMNLP 2014*, 1532–1543.

Reddy, S.; Täckström, O.; Collins, M.; Kwiatkowski, T.; Das, D.; Steedman, M.; and Lapata, M. 2016. Transforming dependency structures to logical forms for semantic parsing. *Trans. Assoc. Comput. Linguist.* 4:127–141.

Reddy, S.; Täckström, O.; Petrov, S.; Steedman, M.; and Lapata, M. 2017. Universal semantic parsing. In *EMNLP 2017*, 89–101.

Song, L.; Wang, Z.; Yu, M.; Zhang, Y.; Florian, R.; and Gildea, D. 2018. Exploring graph-structured passage representation for multi-hop reading comprehension with graph neural networks. *CoRR* abs/1809.02040.

Sorokin, D., and Gurevych, I. 2018. Modeling semantics with gated graph neural networks for knowledge base question answering. In *COLING 2018*, 3306–3317.

Su, Y.; Sun, H.; Sadler, B.; Srivatsa, M.; Gur, I.; Yan, Z.; and Yan, X. 2016. On generating characteristic-rich question sets for qa evaluation. In *EMNLP 2016*, 562–572.

Sun, H.; Bedrax-Weiss, T.; and Cohen, W. 2019. PullNet: Open domain question answering with iterative retrieval on knowledge bases and text. In *EMNLP-IJCNLP 2019*, 2380–2390.

Talmor, A., and Berant, J. 2018a. Repartitioning of the complexwebquestions dataset. *arXiv preprint arXiv:1807.09623*.

Talmor, A., and Berant, J. 2018b. The web as a knowledge-base for answering complex questions. In *NAACL 2018*, 641–651.

Wang, Y.; Zhang, R.; Xu, C.; and Mao, Y. 2018. The apvaturbo approach to question answering in knowledge base. In *COLING 2018*, 1998–2009.

Yao, X., and Durme, B. V. 2014. Information extraction over structured data: Question answering with freebase. In *ACL 2014, Volume 1: Long Papers*, 956–966.

Yao, X., and Van Durme, B. 2014. Information extraction over structured data: Question answering with freebase. In *ACL 2014, Volume 1: Long Papers*, 956–966.

Yih, W.-t.; Chang, M.-W.; He, X.; and Gao, J. 2015. Semantic parsing via staged query graph generation: Question answering with knowledge base. In *ACL-IJCNLP 2015, Volume 1: Long Papers*, volume 1, 1321–1331.

Yin, W.; Yu, M.; Xiang, B.; Zhou, B.; and Schütze, H. 2016. Simple question answering by attentive convolutional neural network. In *COLING 2016*, 1746–1756.

Yu, M.; Yin, W.; Hasan, K. S.; dos Santos, C.; Xiang, B.; and Zhou, B. 2017. Improved neural relation detection for knowledge base question answering. In *ACL 2017, Volume 1: Long Papers*, volume 1, 571–581.