BERT-based Semantic Query Graph Extraction for Knowledge Graph Question Answering

Zhicheng Liang[*,†,1], Zixuan Peng[†,2], Xuefeng Yang[2], Fubang Zhao[2], Yunfeng Liu[2], and Deborah L. McGuinness[1]

[1] Department of Computer Science, Rensselaer Polytechnic Institute, USA
[2] Zhuiyi Technology, China

Abstract. Answering complex questions involving multiple entities and relations remains a challenging Knowledge Graph Question Answering (KGQA) task. To extract a Semantic Query Graph (SQG), we propose a BERT-based decoder that is capable of jointly performing multi-tasks for SQG construction, such as entity detection, relation prediction, output variable selection, query type classification and ordinal constraint detection. The outputs of our model can be seamlessly integrated with downstream components (e.g. entity linking) of a KGQA pipeline to construct a formal query. The results of our experiments show that our proposed BERT-based semantic query graph extractor achieves better performance than traditional recurrent neural network based extractors. Meanwhile, the KGQA pipeline based on our model outperforms baseline approaches on two benchmark datasets (LC-QuAD, WebQSP) containing complex questions.[§]

1 Introduction

Semantic parsing (SP) based approaches to knowledge graph question answering (KGQA) aim at building a semantic parser that first converts natural language questions into some logical forms, and then into formal queries like SRARQL that can be executed on the underlying KG to retrieve answers. For these approaches, constructing the semantic query graph (SQG) plays a vital role. For example, the SQG of a natural language query (NLQ) "*What awards have been won by the executive producer of Fraggle Rock?*" involves three nodes and two labeled edges, i.e. {(*Fraggle Rock*, `dbo:executiveProducer`, *?x*), (*?x*, `dbo:award`, *?uri*)} if represented using triples, where *?x* and *?uri* are some free variables. The answers to this query should be the grounded KG nodes for the output variable *?uri*. Despite some work on abstract query graph prediction [1, 8], there is yet to be an end-to-end model that jointly performs query graph identification along with entity mention detection and relation prediction. To this end, we propose a novel BERT-based neural network to extract SQG in an end-to-end manner for answering complex questions with multiple triple patterns. We evaluate our approach on two KGQA benchmark datasets containing complex questions. The experimental results demonstrate that our approach, by using a simple pipeline built on top of our proposed SQG extractor, improves the overall KGQA performance outperforming the baseline approaches.

[*]Work partially done during an internship at Zhuiyi Technology.

[†]Equal contribution.

[§]Our code and data are available at: https://github.com/gychant/BERT-NL2SPARQL

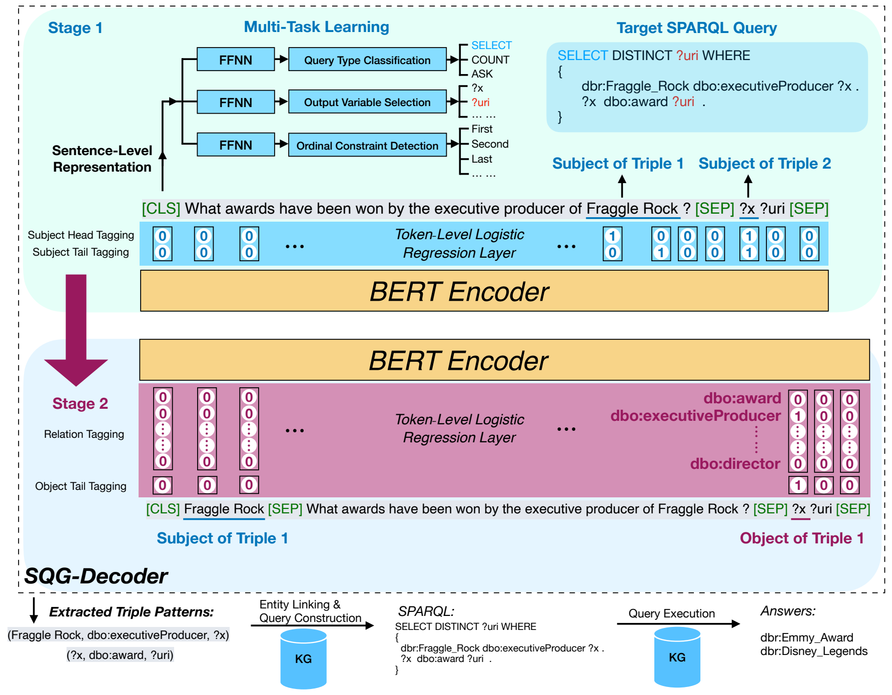
Stage 1
Multi-Task Learning
Target SPARQL Query
FFNN
Query Type Classification
SELECT
COUNT
ASK
FFNN
Output Variable Selection
?x
?uri
… …
FFNN
Ordinal Constraint Detection
First
Second
Last
… …
SELECT DISTINCT ?uri WHERE
{
dbr:Fraggle_Rock dbo:executiveProducer ?x .
?x dbo:award ?uri .
}
Sentence-Level Representation
Subject of Triple 1
Subject of Triple 2
[CLS] What awards have been won by the executive producer of Fraggle Rock ? [SEP] ?x ?uri [SEP]
Subject Head Tagging
Subject Tail Tagging
Token-Level Logistic Regression Layer
BERT Encoder
BERT Encoder
Stage 2
Relation Tagging
Object Tail Tagging
Token-Level Logistic Regression Layer
dbo:award
dbo:executiveProducer
dbo:director
[CLS] Fraggle Rock [SEP] What awards have been won by the executive producer of Fraggle Rock ? [SEP] ?x ?uri [SEP]
Subject of Triple 1
Object of Triple 1
SQG-Decoder
Extracted Triple Patterns:
(Fraggle Rock, dbo:executiveProducer, ?x)
(?x, dbo:award, ?uri)
Entity Linking & Query Construction
KG
SPARQL:
SELECT DISTINCT ?uri WHERE
{
dbr:Fraggle_Rock dbo:executiveProducer ?x .
?x dbo:award ?uri .
}
Query Execution
KG
Answers:
dbr:Emmy_Award
dbr:Disney_Legends

Fig. 1: An overview of the KGQA pipeline with our Semantic Query Graph Decoder (SQG-Decoder) illustrated by a running example.

2 Our approach

An overview of our approach for KGQA is given in Fig. 1. We propose a semantic query graph decoder (SQG-Decoder) that, given an NLQ, aims to predict a set of triple patterns, each in the form of (*Subject*, *Relation*, *Object*), where *Subject* and *Object* are either some text spans for entities, or some introduced free variables, while *Relation* is a grounded relation/predicate defined by the KG schema. As opposed to single-relation questions, complex questions involve multiple triple patterns that form a directed acyclic graph (DAG). SQG-Decoder uses BERT [2] as an encoder to obtain both sentence-level and token-level contextual embeddings of the input NLQ. Specifically, we construct the input of the BERT encoder in the format of "`[CLS] NLQ [SEP] Free variables [SEP]`". This enables free variables to interact with the entire NLQ sentence via attention mechanisms used in BERT.

With BERT as an encoder, we propose a novel decoding scheme to extract triple patterns given an NLQ. It predicts each triple pattern (s_j, p_j, o_j) in two stages, as shown in Fig. 1. The first stage is to determine the span of a subject, which denotes an entity mention in the NLQ or a free variable. Given the token-level vector representations obtained from BERT, i.e. $\mathcal{H} = \{\mathbf{h}_0, \mathbf{h}_1, ..., \mathbf{h}_{l-1}\}$, $\mathbf{h}_i$ is used to predict whether the i-th token is the head/tail token of the subject span s_j by applying a linear feed-forward layer (or stacking multiple such layers with non-linearity) with sigmoid activation to perform binary classification with logistic regression. The second stage is to determine the span of the object and the type of relation for the triple pattern, conditioned on the predicted subject in the first stage. To achieve this, the predicted s_j and the NLQ are concatenated (with $\{0, 1\}$ masks to distinguish the two different parts) and fed into the encoder again to obtain the subject-aware token-level representations $\mathcal{H}' = \{\mathbf{h}'_0, \mathbf{h}'_1, ..., \mathbf{h}'_{l-1}\}$. Specifically, the relation p_j is jointly predicted with the head token of the object mention o_j considering that they are dependent on each other. The tail token of the object mention o_j is identified in a similar way as that of the subject span s_j. In this way, the encoder layer is shared across the described two stages along with task-specific output layers for subject and object-relation extraction, respectively.

Note that our SQG-Decoder is capable of predicting multiple triple patterns for a query. While doing this, we need to handle the issue of overlapping text spans since an entity mention may appear in more than one triple pattern given an NLQ, e.g. in Fig. 1 the variable *?x* is involved in two triple patterns. Our solution is, instead of labeling the text span of an entity by using `1` for all involved tokens and `0` otherwise, we label each token in the NLQ as follows: (`1`, `0`) for the head of entity mention, (`0`, `1`) for the tail of entity mention, (`1`, `1`) if a token is both the start and the end of an entity mention, and (`0`, `0`) otherwise.

To train our SQG-Decoder, we need annotations of text spans in an NLQ that correspond to the canonical entity names of the subject/object of a triple pattern. To automate this process, we adopt a reverse linking algorithm mainly based on n-gram fuzzy matching to annotate the spans of entities mentioned in the corresponding SPARQL query. To further improve linking performance, we also leverage pre-trained word embeddings for measuring the similarity between entity names from the KG and their surface forms in text.

Given an NLQ, the model parameters are learned by maximizing the likelihood of a set of ground truth triple patterns. Practically, our decoding scheme can be adapted to various neural-net-based encoders (e.g. BiLSTMs, RNNs, Transformers) from which token-level representations of the encoder input can be obtained. To ground extracted text spans to the KG, we employ the same entity linking tools used by the baselines in our experiments for fair comparison. Specifically, we built an Apache Lucene Index to retrieve entity candidates, via string similarity between their names and entity mention queries following the baselines [3, 5] for LC-QuAD, and used the entity linking results [6] for WebQSP.

In addition to triple pattern extraction, we leverage BERT to perform three auxiliary classification tasks that are important to constructing the query graph as follows. **1) Output Variable Selection.** Since a query graph may have multiple variables, we need to determine which one represents the returned answer. **2) Query Type Classification.** Query type refers to the type of information that is asked upon the output variable. Here we focus on three types of queries: `SELECT`, `COUNT`, and `ASK` (boolean) query. **3) Ordinal Constraint Detection.** In some cases, ordinal constraints are imposed on the output variables. For instance, to answer the question "*What is the first book Sherlock Holmes appeared in?*", we need to sort the books in which the detective appeared by publication date and then return the first as the answer.

To perform the above three tasks, we feed the transformer output for the `[CLS]` token into separate classification layers. Since these tasks are related to each other, we adopt a multi-task learning strategy to train the SQG-Decoder jointly by combining the cross-entropy losses of these tasks together with the binary cross-entropy losses of the principal task, i.e. triple pattern extraction.

3 Experiments & Results

Datasets. We evaluate our approach on two datasets, **LC-QuAD** [4] and **WebQSP** [6], both targeting complex questions and having ground truth SPARQL annotations.
Evaluation Metrics. We use accuracy of the answer set as the main evaluation metric (i.e. the percentage of test questions for which the predicted answer set exactly matches the ground-truth answer set). Precision, recall and F_1 scores are also reported. We use the official evaluation script for WebQSP and the average F_1 is reported by [6].
Experimental Setup. Our SQG-Decoder is fine-tuned based on the pre-trained language model BERT [2]. We compare $\text{BERT}_{\text{BASE}}$ (12-layers, 768-hidden, 12-heads) with $\text{BERT}_{\text{LARGE}}$ (24-layers, 1024-hidden, 16-heads) to examine the performance with different model sizes. We use the Adam optimizer with a learning rate of 2×10^{-5} and a batch size of 32. The best model is selected using the development set.
End-to-end KGQA Performance. We only compare with previous work that reported end-to-end KGQA performance, i.e. those starting with raw input query without linked entities or relations given. Table 1 & 2 show that we achieve the best F_1 score of 54.9 on LC-QuAD and the best accuracy of 66.0 on WebQSP, both using $\text{BERT}_{\text{LARGE}}$.

Table 1: LC-QuAD results

Methods	Pre.	Rec.	F_1	Acc.
QAmp[5]	25.0	50.0	33.0	-
WDAqua-core1[3]	59.0	38.0	46.0	-
Ours w/ $\text{BERT}_{\text{LARGE}}$	51.1	59.3	**54.9**	**46.6**
Ours w/ $\text{BERT}_{\text{BASE}}$	49.9	57.3	53.3	45.6

Table 2: WebQSP results

Methods	Pre.	Rec.	Avg. F_1	Acc.
STAGG[6]	70.9	80.3	**71.7**	63.9
HR-BiLSTM[7]	-	-	-	63.9
Ours w/ $\text{BERT}_{\text{LARGE}}$	85.9	75.5	70.3	**66.0**
Ours w/ $\text{BERT}_{\text{BASE}}$	85.1	73.1	69.2	65.5

Evaluation of Triple Pattern Extraction. This evaluates the main contribution of our work. We compare the precision, recall and F_1 scores of extracted triple patterns given that the SQG-Decoder is built on top of two types of encoders: Bi-LSTM encoder with GloVe Embeddings, and $\text{BERT}_{\text{BASE}}$, as summarized in Table 3. The results demonstrate that the latter always outperforms the former, mainly due to higher recall, but stays close in terms of precision. This shows that our proposed SQG-Decoder can be adapted to other types of encoders besides those based on transformers.
Error Analysis. We randomly select 100 incorrectly answered questions from the LC-QuAD test set. Major errors observed include: 1)**Missing triple patterns** (54%). Errors of this type are attributed to some predicates that are in the test set but are never seen in the training set, as well as the data sparcity issue. 2) **Mispredicting semantically**

Table 3: Triple pattern extraction performance

Methods	LC-QuAD			WebQSP		
	Pre.	Rec.	F_1	Pre.	Rec.	F_1
BiLSTM Encoder + GloVe	58.7	33.5	42.6	74.6	58.8	65.8
$\text{BERT}_{\text{BASE}}$ Encoder	62.8	56.1	59.2	75.3	70.9	73.0

close predicates (32%). This is typically because some predicates have very similar semantic meanings while the NLQ is not informative enough to distinguish between them. 3) **Entity span misidentification & redundant predicates, etc** (14%). Errors of this type include incorrect entity span detection that may cause failures in entity linking, and mispredicting redundant predicates, etc.

4 Conclusions and Future Work

We present a BERT-based Semantic Query Graph Decoder (SQG-Decoder) for answering complex natural language queries, using knowledge graphs, which jointly learns multiple subtasks in an end-to-end trainable manner. Our approach remarkably outperforms the baselines on two KGQA benchmarks that contain complex questions. As future work, we would like to explore answering questions with more complex temporal and spatial constraints using neural networks.

Acknowledgments This work was partially supported by the DARPA MCS program award number N660011924033 to RPI under USC-ISI West.

References

1. Chen, Y., Li, H., Hua, Y., Qi, G.: Formal query building with query structure prediction for complex question answering over knowledge base. In: IJCAI. pp. 3751–3758 (2020)
2. Devlin, J., Chang, M.W., Lee, K., Toutanova, K.: BERT: Pre-training of deep bidirectional transformers for language understanding. In: NAACL. pp. 4171–4186 (2019)
3. Diefenbach, D., Both, A., Singh, K., Maret, P.: Towards a question answering system over the semantic web. Semantic Web (Preprint), 1–19
4. Trivedi, P., Maheshwari, G., Dubey, M., Lehmann, J.: LC-QuAD: A corpus for complex question answering over knowledge graphs. In: ISWC. pp. 210–218. Springer (2017)
5. Vakulenko, S., Fernandez Garcia, J.D., Polleres, A., de Rijke, M., Cochez, M.: Message passing for complex question answering over knowledge graphs. In: CIKM. pp. 1431–1440. ACM (2019)
6. Yih, W.t., Richardson, M., Meek, C., Chang, M.W., Suh, J.: The value of semantic parse labeling for knowledge base question answering. In: ACL. pp. 201–206 (2016)
7. Yu, M., Yin, W., Hasan, K.S., Santos, C.d., Xiang, B., Zhou, B.: Improved neural relation detection for knowledge base question answering. In: ACL. pp. 571–581 (2017)
8. Zheng, W., Zhang, M.: Question answering over knowledge graphs via structural query patterns. arXiv preprint arXiv:1910.09760 (2019)