Deep-gAnswer: A Knowledge Based Question Answering System

Yinnian Lin, Minhao Zhang, Ruoyu Zhang, and Lei Zou(✉)

Peking University, Beijing, China
{linyinnian,zhangminhao,ry_zhang,zoulei}@pku.edu.cn

Abstract. In this demonstration, we present Deep-gAnswer, a knowledge-based question answering system. gAnswer is based on semantic parsing and heuristic rules for entity recognition, relation recognition, and SPARQL generation. By making use of a pre-trained model, we implement new entity and relation recognition networks. Also, it is found that the traditional method works better when information of entity and relation is correctly given. Therefore, we combine entity and relation recognition networks with the previous SPARQL generation process to get Deep-gAnswer. Experimental results show that Deep-gAnswer outperforms the previous one, especially on Chinese dataset.

1 Introduction

Knowledge graph has been through rapid development and is applied in many fields. Knowledge-based question answering (KBQA) [9] is one of its most popular applications. Given a natural language question, KBQA systems are designed to extract the answers from a background knowledge base. A common solution in previous systems is transforming the question into a knowledge base query such as SPARQL [3] to return answers. gAnswer [4] is one of such systems with satisfying performance, which won the first place in QALD-9 [7].

However, gAnswer's performance severely relies on its two components: the dependency tree parser and paraphrase dictionary. gAnswer parses the question into a dependency tree for node and relation recognition that decides which parts of the question (usually called mentions) may refer to an entity, a relation, or a variable. Paraphrase dictionary aims to find the proper matching between relation mentions and predicates in the knowledge graph. However, most dependency tree parsers only work well on simple questions. When the number of relations in the question increases, errors often occur in the dependency parsing stage and are passed to the following stages, severely harming the overall performance. Meanwhile, the paraphrase dictionary requires a very large amount of data and time to construct. What's worse, the dictionary is closely related to the background knowledge graph. So, when we change to another knowledge graph, the former dictionary may become useless because the predicates in the new knowledge graph vary.

To overcome the problems above, we make use of the up-to-date pre-trained models. Such models like BERT [5] are trained on massive data and able to

extract underlying features from natural language sentences. In other words, their generality is better than the old ways in gAnswer.

In this demo, we propose a novel KBQA framework based on gAnswer, named Deep-gAnswer. We take node recognition as a sequence labeling task while relation recognition as a ranking task, and train two models respectively. The recognition model is intended to improve mention detection and dependency tree parsing while the relation ranking model can replace paraphrase dictionary.

We conduct a comparative experiment between gAnwser and Deep-gAnswer and experimental results show that Deep-gAnswer prevails gAnswer in terms of F1 score, especially on Chinese questions and complex questions.

2 System Architecture

The Deep-gAnswer system consists of four parts: question understanding, query graph construction, SPARQL generation, and answer collection. The system architecture is depicted in Fig. 1.

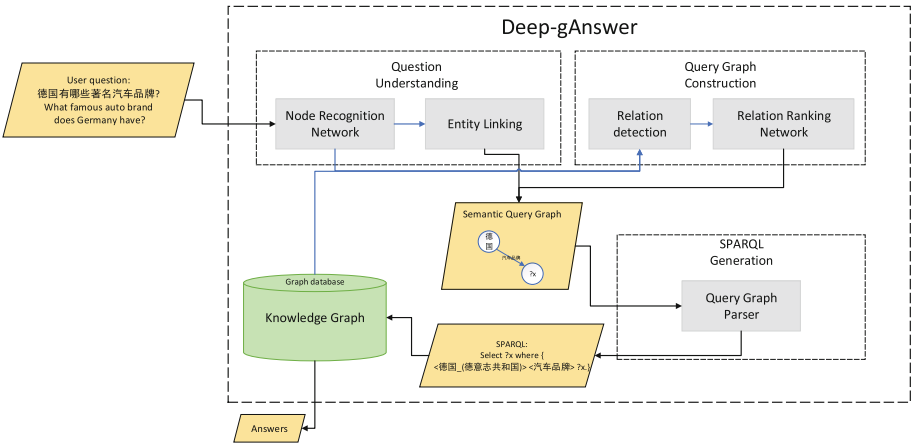


Fig. 1. The Deep-gAnswer architecture

Question Understanding. This is the first step for Deep-gAnswer to answer a question. The purpose of this procedure is to detect all the mentions of entities and variables via a node recognition network. Then, in entity linking module, every entity mention will be mapped to a set of exact entity names in the knowledge graph with a dictionary and string similarity. Notice that one mention may be linked to multiple entities as long as their similarity is high enough. In practice, the system will maintain a fixed number of linking results of a mention and generate a list of ranked SPARQL.

Query Graph Construction. In this step, the system first parses the question to find the relations among entity mentions and variable mentions. The system builds a dependency tree of the question to extract the relation between mentions. Then, the system enumerates every relation and feeds it to the relation ranking network to get its possible predicate. Having all entity, variable, and relation information, the system builds a semantic query graph in a depth-first search manner.

SPARQL Generation and Answer Collection. As mentioned before, most errors of gAnswer come from node and relation recognition and we have improved them with deep-learning-based models. Therefore, we simply follow the subgraph matching strategy from gAnswer for SPARQL generation and answer collection. The system generates a list of ranked SPARQLs and sends them as queries to a graph database to get the final answers.

3 Techniques

In this section, we mainly focus on the implementation of the new node recognition network and relation ranking network.

Node Recognition Network. In our definition, there are four kinds of nodes in a question: entities, variables, literals, and types. Generally, an entity represents a specific thing or person and a type refers to a category of entity. Sometimes type itself can be taken as an entity. A literal means a value or an attribute. For example, a specific actor is an entity, while his height and nickname are literals. A variable is an unknown node that can indicate an entity, a type, or a literal. We design the node recognition network to solve a sequence labeling task with tag space $\{O, Eb, Ei, Vb, Vi, VTb, VTi, Tb, Ti, VLb, VLi, Lb, Li\}$. The specific meanings of these tags are shown in Table 1.

In terms of the NER problem, BERT-based models have been proved successful in previous works [1,5]. Therefore, we adopt this strategy to use a BEAT-based model as an encoder. A question first goes through RoBERTa [6] encoder

Table 1. Tag meanings

Tag	Meaning
O	Not an entity nor variable
Eb, Ei	Mention to an entity
Vb, Vi	Mention to variable that refers to an entity
VTb, VTi	Mention to variable that refers to a type
Tb, Ti	Mention to a type
VLb, VLi	Mention to a variable that refers to a literal value
Lb, Li	Mention to literal value

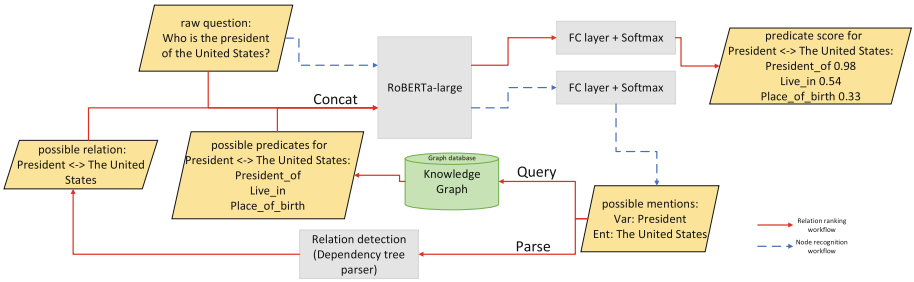


Fig. 2. Workflow of the node recognition network and the relation ranking network

and then to the output layer for the tag sequence. We choose RoBERTa because it outperforms other models in our experiments.

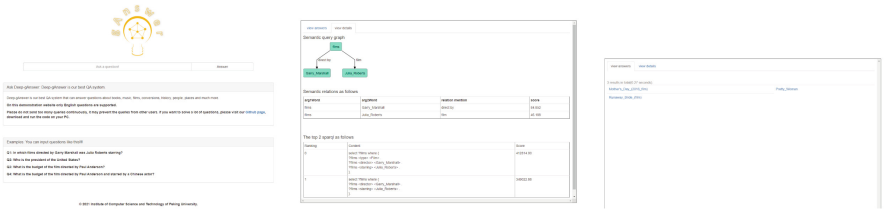
To train the model, we also developed a dataset by ourselves annotating node-to-mention mappings. We manually tagged natural language questions with the previously mentioned tag space. The questions come from the existing KBQA question sets. We mainly used LC-QUAD question set [8] for English questions and CCKS question set [2] for Chinese questions. Both question sets provide natural language questions and corresponding SPARQLs. Tagged questions serve as the input of our node recognition network and relations extracted from the given SPARQLs are feed to the relation ranking model.

Relation Ranking Network. The goal of relation recognition is to find all pairs of related nodes and their predicates. Due to the complexity of the natural language questions and lack of training data, an end-to-end model may not handle this task very well. However, we can first attain all related node pairs and use a ranking model to get the top k most likely predicates easily.

Our experiments show that with node mentions given, a dependency tree parser can reach a satisfying accuracy. Therefore, we can extract all related node pairs from the dependency tree. For each pair, we query the knowledge graph for candidate predicates. If one of the nodes is an exact entity set, we can simply search for its connected predicates. If both nodes are variables, we search the query graph to find an entity and get its k-hops-away predicates as candidates. For each candidate predicate, we concatenate the question, the mentions of the two related nodes, and the predicate itself to form an input sentence and sent it to RoBERTa encoder. We use a full connected layer as a decoder to output a score. The encoder is shared between both node recognition and relation ranking network to learn global information. In this way, we get a ranked predicate list for each relation. The workflow of the two models is in Fig. 2.

4 Demonstration

We build a website to demonstrate how Deep-gAnswer answers a natural language question, providing users with a friendly interface and straightforward



(a) Query Page (b) Visual Results (c) Answer page

Fig. 3. Demonstration of Deep-gAnswer

visual results of the translation from natural language questions to SPARQL. We use an English knowledge graph, DBpedia, to set up the background system.

Figure 3(a) is our query page. Users can input a question freely in the center text box and ask the system. Here we use a complex question, *In which films directed by Garry Marshall was Julia Roberts starring?* as an example.

Figure 3(b) shows the result demonstration page. At the top is the query graph. We can see *Gary Marshall*, *Julia Roberts* and *films* are recognized as mentions. In the middle, user can see the result of relation detection. With correct node mentions given, the dependency parser can successfully identify the relation and its corresponding mentions. The generated SPARQL list is at the bottom. Relation mentions become actual predicates with the help of the relation ranking model and the score of a SPARQL here is based on the score of each predicate. To check the final answer, users can jump to the answer page as shown in Fig. 3(c).

Acknowledgment. This work was supported by National Natural Science Foundation of China(NSFC) under grant 61932001. The corresponding author of this work is Lei Zou.

References

1. Akbik, A., Blythe, D., Vollgraf, R.: Contextual string embeddings for sequence labeling. In: Proceedings of the 27th International Conference on Computational Linguistics, pp. 1638–1649 (2018)
2. Han, X., et al.: Overview of the CCKS 2019 knowledge graph evaluation track: entity, relation, event and QA. arXiv preprint arXiv:2003.03875 (2020)
3. Hommeaux, E.P.: SparQL query language for RDF (2011)
4. Hu, S., Zou, L., Yu, J.X., Wang, H., Zhao, D.: Answering natural language questions by subgraph matching over knowledge graphs. IEEE Trans. Knowl. Data Eng. **30**(5), 824–837 (2017)
5. Kenton, J.D.M.W.C., Toutanova, L.K.: Bert: pre-training of deep bidirectional transformers for language understanding. In: Proceedings of NAACL-HLT, pp. 4171–4186 (2019)
6. Liu, Y., et al.: Roberta: a robustly optimized BERT pretraining approach. arXiv preprint arXiv:1907.11692 (2019)

7. Ngomo, N.: 9th challenge on question answering over linked data (QALD-9). Language 7(1) (2018)
8. Trivedi, P., Maheshwari, G., Dubey, M., Lehmann, J.: LC-QuAD: a corpus for complex question answering over knowledge graphs. In: d'Amato, C., et al. (eds.) ISWC 2017. LNCS, vol. 10588, pp. 210–218. Springer, Cham (2017). https://doi.org/10.1007/978-3-319-68204-4_22
9. Unger, C., Freitas, A., Cimiano, P.: An introduction to question answering over linked data. In: Reasoning Web International Summer School (2014)