An Integrated Path Formulation Method for Open Domain Question Answering over Knowledge Base

Wen Dai, Huiwen Liu, Yan Liu, Rongrong Lv, Shuai Chen

Xiaomi Corporation, AI
{daiwen, liuhuiwen, liuyan15, lvrongrong, chenshuai3}@xiaomi.com

Abstract. Knowledge Based Question Answering (KBQA) plays a crucial role in the scope of open-domain Question Answering. In prior work, we extract the entity, predict the relation and formulate the Sparql query as a pipeline. The downstream tasks rely on the output of upstream tasks. In COVID 19 Question-answering contest of CCKS2020, we propose an integrated method, which treats all individual tasks in candidate paths generation as a whole, and utilize semantic similarity and re-ranking techniques to select the best path. Our method obtained an F1 score of 85.45% on the final leaderboard.

Keywords: KBQA, integrated, semantic similarity.

1 Introduction

Knowledge Based Question Answering (KBQA) plays a crucial role in the scope of open-domain Question Answering. In prior work, entity-linking, relation prediction, path formulation, among others, are implemented as a pipeline to obtain the answer[1]. The downstream tasks rely on the output of upstream tasks. Thus, errors in the upstream tasks would propagate to the downstream tasks. For example, if an incorrect entity is extracted in the first step, we could hardly obtain the expected answer regardless of what strategies we take in the following steps.

In this paper, we propose an integrated method, which treats entity-linking, relation prediction and path formulation as a whole. The proposed methods is composed of three main procedures. In the first step, we generate all possible paths for the question. Instead of extracting entity, predicting relation and formulating paths one by one, we implement them as an integrated module, which we call candidate paths extraction. We use brute matching and fuzzy matching to extract all possible starting nodes. Then following several path patterns, we extract almost all possible candidate paths. In the second step, we utilize semantic similarity techniques to select the more probable paths. In fact, we rely on semantic similarity module to select top candidate paths for the question, instead of selecting possible ones in the first step. In this way, we can avoid errors in entity-linking and relation prediction affect the final answer selection. In the third step, we use re-ranking techniques to choose the best path from the top candidates. Entity-linking results act as one important factor, among others, in re-ranking module. The workflow of the proposed method in this article is illustrated in Fig. 1.

The main contribution of our method is as below:

1. We integrate the entity-linking, relation prediction and path formulation as a whole process, and extract nearly all possible candidate paths, which is able to reduce the influence of upstream errors on the final results. We use an ensemble of semantic similarity models in the back end to choose the more probable candidate paths.
2. We use about 10 path patterns to extract more candidate paths for the questions, which is the most as far as we know. The more patterns we extract, the higher coverage we can expect.
3. We propose an effective way to convert the candidate path to plain text for semantic similarity calculation. We remove intermediate nodes in the sparql query and convert the answer node to an unique symbol "^".

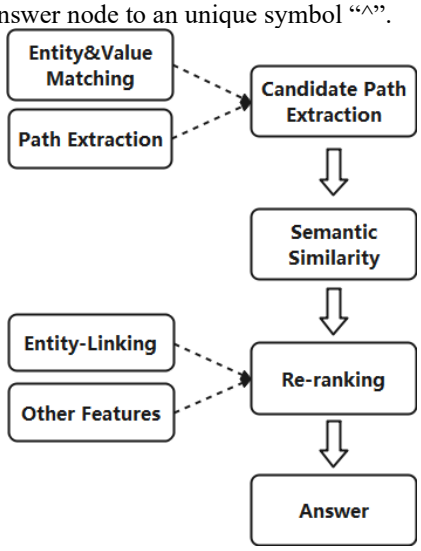


Fig. 1. Workflow of our method

2 Related Work

There are two main methods in Knowledge Based Question Answering: semantic paring and information retrieval. In semantic parsing, we translate natural language into an intermediate logical forms, and then convert it to an executable language for knowledge base[2, 3]. Traditional semantic paring methods use predefined templates to convert natural language into a formalized representation. But this approach needs lots of human labeling, and lacks ability of generalization. Some researchers decompose parsing into several main steps: entity-linking, predicate sequence identification, constraint distinguishing to form the query graph[4]. In information retrieval, we recognize mentions in the question, and links them to entities in the knowledge base, then subgraphs related to these entities are extracted as candidate sets. Afterwards, features are exploited to rank the candidates and select the best candidate to obtain the answer[5, 6].

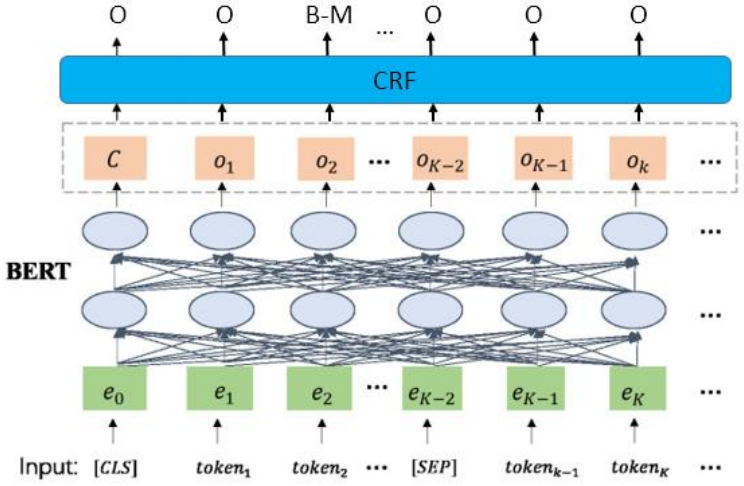
3 Methods

3.1 Entity Recognition and Linking

Entity Recognition and Linking (ERL) is mainly composed of two parts, entity recognition and entity disambiguation. For each part, we use BERT model for fine-tuning. We present a neural approach to discovering gold mentions and linking them to the correct entities in the given knowledge base. In case of mistakes of ERL would affect the downstream tasks, we use results of ERL as a feature in re-ranking instead of extracting candidate paths based on ERL.

3.1.1 Entity Recognition

We implement entity recognition tasks in two ways. In the first method, BERT + CRF is used to extract mentions as illustrated in Fig. 2. BERT+CRF is a sequence labeling model that extracts mentions in questions by labeling positions. Firstly, we use the dictionary to find all possible candidate words. Then, we use BERT + CRF to encode the context to find all possible mentions, and if there are some mentions in them, we will keep some of them.



Text_a: 龙卷风的英文名是什么？ Text_b: None

Fig. 2. Illustration of entity recognition model "BERT + CRF"

In the second method, the BERT + FC model is used to extract the mentions in the question as illustrated in Fig. 3. BERT + FC is a text classification model that converts sequence labeling tasks into text classification problems by marking the position of candidate words in the question. Firstly, we also need to use a dictionary to find all possible candidate words. Then, we extract a candidate word and mark it in the question, and judge whether the marked candidate word is a mention of the question.

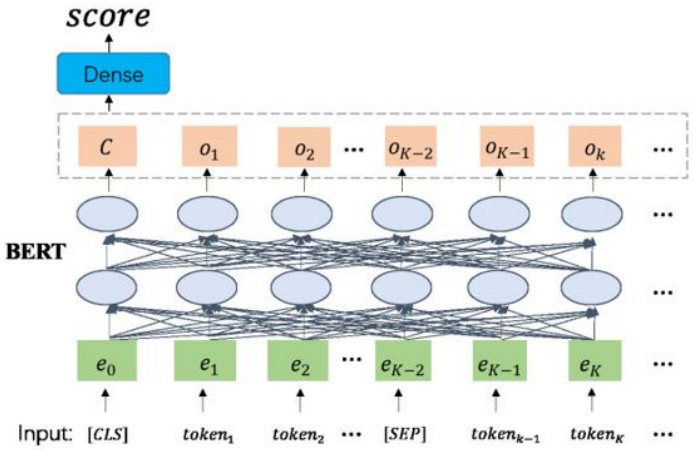


Fig. 3. Illustration of entity recognition model "BERT + FC"

Both the BERT+CRF model and the BERT+FC model are trained using the 5-fold cross-validation method, and then use the voting mechanism to select candidate words with more than 2 occurrences as the question mention.

3.1.2 Entity Disambiguation

In order to improve the accuracy of entity linking, we only disambiguate the entity description information and mention different candidate entities. The entity description information and mention are the same, and are associated by default. As same as entity recognition task, we still use BERT+FC model for the entity disambiguation, we will mark the question with the mention location as text A and the entity description as text B, as illustrated in Fig. 4. We score the similarity of all entities in the set of candidate entities, and then sort them according to the score.

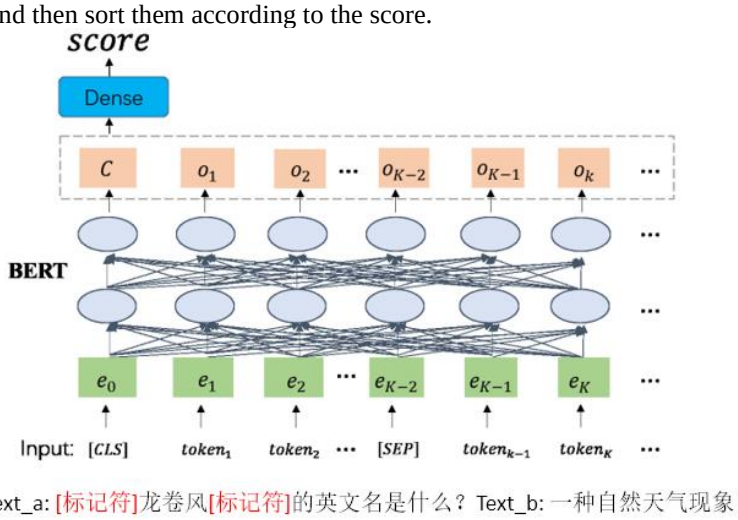


Fig. 4. Illustration of entity disambiguation model

3.2 Candidate Path Extraction

In most approaches, candidate path extraction is based on the results of ERL. Here, we decouple path extraction from ERL in case of mistakes of ERL would affect the downstream tasks. In Knowledge Graph, a fact is usually composed of entity, property and value. We match the question to all possible entities and values in Knowledge Graph mainly by Brute Matching and Fuzzy Matching.

3.2.1 Entity&Value Matching

We use two matching ways to extract mentions in the question: brute matching and fuzzy matching.

Most of the mentions in the question can be directly matched to entities (surrounded by <>, like <木星_（太阳系八大行星之一）>) or values (surrounded by "", like "炎帝最小的女儿") in the Knowledge Graph. Apart from this, some mentions can be matched to alias (e.g., '简称','别称') of entities. All of the above matching results are taken into consideration, and we call this process as brute matching. Some of the results in brute matching are short and mismatched. We follow some straightforward rules for filtering, for example, filter the matching results with only one character or number.

In brute matching, some longer values can't be directly matched to mentions in the question. For example, as to the question "广州呼吸疾病研究所所长的籍贯", the correct node <2013年7月至今广州呼吸疾病研究所所长> can't be exactly matched to any mentions in the question. In order to get the correct nodes in Knowledge Graph, we build an inverted search index with Elasticsearch. During fuzzy matching with Elasticsearch, we give a higher weight to numbers in calculation of correlation score, and date formats should be normalized.

3.2.2 Path Extraction

Starting from the matched nodes, we extract candidate paths from Knowledge Graph. One-hop and paths-combination are two kinds of operations that can extract new paths based on some path patterns. This process is shown in Fig.5 and all the path patterns are shown in Table 1.

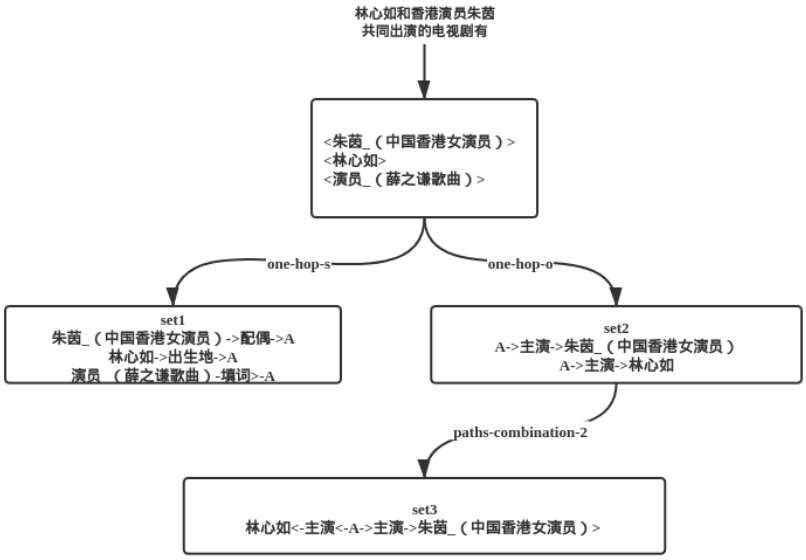


Fig. 5. Example for path extraction. One-hop-s/o: the current node hop from s or o by a triple. Paths-combination-n: n paths which have the same current node could be combined as a new path and this operation restrictive conditions on the base paths. All the path set together constitute the final candidate paths.

Table 1. Path patterns in extraction.

Set name	Operation	Base
Spa	One-hop-s	nodes
Apo	One-hop-o	nodes
sppa	One-hop-s	spa
spopa	One-hop-o	spa
opspa	One-hop-s	apo
oppa	One-hop-o	apo
two-ent-path	Paths-combination-2	spa,apo
two-ent-path-1-hop	One-hop-s	two-ent-path
three-ent-path	Paths-combination-3	spa,apo
three-ent-path-1-hop	One-hop-s	three-ent-path
two-ent-path-append	Paths-combination-2	oppa,apo

3.3 Semantic Similarity Calculation

After generating all possible candidate paths under the guidance of predefined path patterns, we design a semantic similarity module to calculate the probability of each

path to be a correct parsing for the question. This module regards question and candidate path as a pair of sentences and calculate their semantic similarity. The higher their similarity is, the more likely the candidate is a correct parsing for the question. In order to get a better similarity result, we remove subordinate elements in the candidate path. For example, as to the question “蜀道难的作者祖籍在哪里”, there is a candidate path “select ?x where { <蜀道难_（李白作古体诗）> <作者> ?y. ?y <祖籍> ?x. }”. We remove the underline and the suffix in the entity name because the suffix sometimes has nothing to do with the question literally and would confuse the similarity model. In fact, we postpone the entity-linking in the later re-ranking module. Also, we remove the intermediate node “?y” because it is necessary for query language but indeed insignificant in human language. Finally, we replace the answer node as symbol “^” to represent the terminal of the query language. Thus, our goal is to measure the similarity of these two sentences: “蜀道难的作者祖籍在哪里” and “蜀道难作者祖籍^”. It should also be noted that we remove the predicate “<中文名>” from all cases, because it is not reasonable semantically and we can always find a replacement without “<中文名>”. For example, as to the question “演员刘涛的丈夫是做什么的”, we replace the candidate path “select ?z where { ?x <中文名> "刘涛" . ?x <丈夫> ?y . ?y <职业> ?z . }” with “select ?z where { <刘涛_（中国内地女演员）> <丈夫> ?y . ?y <职业> ?z . }”.

We use official training data to generate positive samples, and use the candidate paths for training data to generate negative samples. In order to make up a reasonable training set, on one hand, we repeat the positive samples N times so that we can emphasize on the correct paths, on the other hand, we randomly select a specific proportion P of candidate paths to cover different patterns of incorrect paths. When selecting negative samples, we filter the ones that has occurred in the positive samples. Then RoBERTa[7] and NEZHA[8] are utilized to learn from the training samples and serve as the semantic similarity models. We set different N and P to learn a set of semantic similarity models.

Next, we use the models to calculate the semantic similarity between the question and candidate paths for the test set. The scores of different models are averaged to obtain the final semantic similarity. For each question, we rank the candidate paths according to their final semantic similarities, and select the top 20 paths for the next re-ranking module.

3.4 Re-ranking

Semantic similarity scores provided by previous module are not the only factor applied to path selection. An ensemble model, --the random forest, is proposed following the semantic similarity model, which takes other valuable features for consideration as well, such as entity-linking results, answer type, literal match between queries and paths, etc. The top candidate paths sorted by the semantic similarity scores would be re-ranked by the ensemble model scores. Then, the current best path is selected as the one ranked first.

In this module, 40 kinds of features are applied to path ranking and the best path selection, approximately. Most of them could be coarse-grained categorized as follows:

Literal Similarities between Path Candidates and Query: A practice is that concatenating entities, properties and relations mentioned in the path and calculating the distance between them and the corresponding question. Two metrics of distance are used there: jaccard distance and Chinese character overlap. However, the literal diversity of entities and their mentions could impact on the similarities (e.g., 张汉卿 - <张学良>). Therefore, these features are replicated with the entities and properties of the path replaced by their mentions. Considering the fine-grained literal similarity, a feature is created by searching the longest common subsequences between path elements (entities/properties/relations) and the query, and by adding them all for similar calculation. A case is shown below:

Query: "成败一知己，生死两妇人"所说的人物有什么重大成就？

Path: <成败一知己，生死两妇人> <主要人物> ?x ?x <主要成就> ?y

Subsequences: 成败一知己，生死两妇人，人物，成就

Time, Number and Name Features: Generally, time tokens and number tokens that mentioned in the question contain vital information for answer searching. In some cases, they are essential constraint to the final path, e.g., “出生在9月25日的秘书有哪些？”. In other cases, they could be implemented to distinguish the true entity from literal similar ones, which may be confused by the match model, e.g., “第85届奥斯卡金像奖最佳女配角”/“第84届奥斯卡金像奖最佳女配角”. The principle of features is that extracting these token from questions, and checking whether they occur in the path or answers. In addition, many questions query names, such as “英文名”，“别称”，“中文名”. Whether name in the question matches the relation is considered in our feature engineering as well.

Entity/Property/Relation Features: Whether the key tokens of questions are in the path or whether the key elements of path are in the question is an important factor for path selection. Such features are as follows: whether tokens provided by entity-linking are in the path; whether key tokens in especial punctuation are in the path (i.e., 《》, （）, “”, ‘’); whether relations/properties of the path are in the question. Besides, the longest mentions of questions would be recorded for path checking.

Answer/ Path Related Features: Whether answers are in the question; The number of answers; The hop of the path; The number of entities/properties in the path.

Semantic Similarities: Scores provided by the match model, the difference between the maximum score path and the current path, normalized score, and so on.

Answer Type Match: Based on LTP (HIT-SCIR), we propose a LAT (lexical answer type) tool for typical token extraction from the question and answer type examination. In addition, syntax results provided by LTP are used for feature engineering. A feature is that whether tokens marked “sbv” are in relations of the path.

4 Experiments and Results

4.1 Dataset

We use CCKS 2020 dataset to evaluate our approach. The dataset is published by the CCKS 2020 COVID 19 Question-answering task, which includes a knowledge base, a

mention-entity file, and Q&A pairs for training, validation, and testing. Different from CCKS 2019 Question-answering task, the task this year includes some data and questions regarding COVID 19. The training set has 4000 question and answer pairs, the dev set has 1529 questions, and the test set has 1599 questions.

4.2 Candidate Path Extraction

Firstly, we obtain entity&value matching results for each question. Then starting from the matched nodes, we use spark to extract all possible candidate paths. For the training set, we extracted about 50,000,000 candidate paths. For the test set, we extracted about 23,000,000 candidate paths.

4.3 Semantic Similarity Calculation

We set different hypermeters (N and P) to train semantic similarity models as shown in Table 2. Then the models are used to predict testing samples and output prediction scores. The prediction scores are averaged to obtain each candidate path's final similarity score. Finally, the candidate paths are ranked according to their final similarity score. We select top 20 candidate paths for each question.

Table 2. Semantic similarity training.

N	P	Model
5	0.0045	RoBERTa
15	0.0135	RoBERTa
5	0.0045	NEZHA
10	0.009	NEZHA

4.4 Re-ranking

Under the training set, we use the top 20 candidate paths to collect features and train the re-rank model. Then the re-rank model, along with some useful rules, are utilized to re-rank the top 20 candidate paths for the test set.

The F1-score of our proposed method is illustrated in Table 3. From the comparison, we can find that integrated path formulation brings a large improvement in performance, owing to getting rid of upstream errors. Fuzzy matching, semantic similarity model ensemble and re-ranking model also brings improvement as expected.

Table 3. F1-score on dev set.

method	F1-score
Entity-linking + Relation recognition + Answer selection as a pipeline	0.68
Integrated path formulation with one semantic similarity model	0.80
Add fuzzy matching	0.82
Integrated path formulation with semantic similarity model ensemble	0.85

With Re-ranking	0.88

5 Conclusion

We introduce an integrated path formulation method for open-domain question answering. The method consists of three main steps, candidate paths extraction, semantic similarity calculation and re-ranking. Our method obtained an F1-score of 85.45% on the test set.

References

1. Qu, Y., et al., *Question answering over freebase via attentive RNN with similarity matrix based CNN.* arXiv preprint arXiv:1804.03317, 2018. **38**.
2. Steedman, M., *A very short introduction to CCG*. Unpublished paper. http://www. coqsci. ed. ac. uk/steedman/paper. html, 1996.
3. Berant, J., et al. *Semantic parsing on freebase from question-answer pairs*. in *Proceedings of the 2013 conference on empirical methods in natural language processing*. 2013.
4. Yih, S.W.-t., et al., *Semantic parsing via staged query graph generation: Question answering with knowledge base.* 2015.
5. Yao, X. and B. Van Durme. *Information extraction over structured data: Question answering with freebase*. in *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. 2014.
6. Bordes, A., S. Chopra, and J. Weston, *Question answering with subgraph embeddings.* arXiv preprint arXiv:1406.3676, 2014.
7. Liu, Y., et al., *RoBERTa: A robustly optimized BERT pretraining approach. arXiv 2019.* arXiv preprint arXiv:1907.11692.
8. Wei, J., et al., *NEZHA: Neural contextualized representation for chinese language understanding*. arXiv preprint arXiv:1909.00204, 2019.