SPaReL: A Semantic Parsing Relation Linking Method for Knowledge Base Question Answering

Xinbang Dai
xinbangdai@seu.edu.cn
Southeast University
Nanjing, Jiangsu, China

Huiying Li*
huiyingli@seu.edu.cn
Southeast University
Nanjing, Jiangsu, China

Tenggou Wang
wangtenggou@seu.edu.cn
Southeast University
Nanjing, Jiangsu, China

Lei Li
220201891@seu.edu.cn
Southeast University
Nanjing, Jiangsu, China

Yuxi Feng
220201942@seu.edu.cn
Southeast University
Nanjing, Jiangsu, China

Xin Li
220212072@seu.edu.cn
Southeast University
Nanjing, Jiangsu, China

ABSTRACT

Relation linking is an essential module of knowledge base question answering systems. To overcome the ambiguity of natural language and lack of training data, existing relation linking systems employ varieties of heuristics or aggregations of multiple systems. However, the current state-of-the-art methods heavily rely on the surface text and do not achieve optimal results. Since the semantic parsing structure of the question is a rich source of relation information that can strongly promote relation linking performance, we propose a two-stage Semantic Parsing Relation Linking system: SPaReL, which leverages semantic parsing using abstract meaning representation (AMR) to predict relations. The first stage employs a cross-encoder model that concatenates each candidate relation with a question semantic parsing slot to reduce the influence of irrelevant candidate relations on the prediction performance of the relation linking model. The second stage integrates the sentence information and the slot processed by the semantic parser, uses a dual-encoder model to link the remaining high-relevance candidate relations to predict the relation with the highest score. Our system achieves state-of-the-art results on four KBQA datasets, LC-QuAD 1.0, LC-QuAD 2.0, QALD-7 and QALD-9.

CCS CONCEPTS

• **Computing methodologies** → **Natural language processing**.

KEYWORDS

Relation Linking, Semantic Parsing, Knowledge Base, KBQA

ACM Reference Format:
Xinbang Dai, Huiying Li, Tenggou Wang, Lei Li, Yuxi Feng, and Xin Li. 2022. SPaReL: A Semantic Parsing Relation Linking Method for Knowledge Base Question Answering. In *Proceedings of The 11th International Joint Conference on Knowledge Graphs (IJCKG'22).* ACM, New York, NY, USA, 9 pages. https://doi.org/XXXXXXX.XXXXXXX

*Corresponding author.

IJCKG'22, October 27–29, 2022, Hangzhou, China

ACM ISBN 978-1-4503-XXXX-X/18/06…$15.00
https://doi.org/XXXXXXX.XXXXXXX

1 INTRODUCTION

The knowledge base question answering (KBQA) task aims to provide correct answers to natural language questions. Using semantic parsing to complete the question answering task is an important direction of KBQA [1]. This approach first transforms the corresponding semantic components (including entities, relations, and various constraints) into formal queries (e.g., SPARQL), and then executes the query on a knowledge base (KB) to retrieve answers. Relation linking is a necessary subtask for KBQA. For example, in Figure 1, to transform the question "*How many famous people are born in Long Island?*" into the DBpedia SPARQL query, it is essential to get the KB relation *dbo: birthPlace* of the linked entity *dbr: Long_Island* from the sentence.

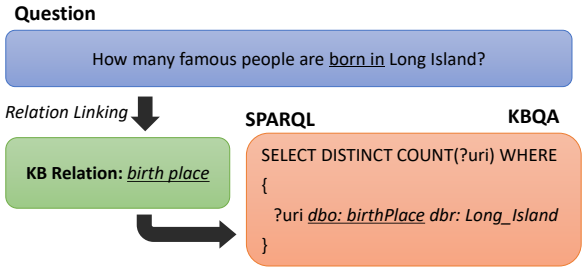

Figure 1: An example illustrating the role of relation linking in KBQA.

Recently, critical tasks of KBQA (such as relation linking) are mainly focused on surface textual information [2–4], ignoring the semantic structure information in the question. Semantic parsing tasks such as AMR have shown to be useful for the KBQA [5]. The method of semantic parsing is to convert the text into structural representations that can identify named entities and normalize relations into PropBank [6] predicates. Therefore, it can alleviate the lexical gap between different phrases with the same predicate. In addition, the sentence structure can be extracted, promoting the task of extracting multiple relations in the question text.

Some works use the AMR method to promote relation linking such as SLING [7] and SemReL [8]. However, SLING relies on a specific target KG (DBpedia) and uses a complex ensemble of different approaches, making it difficult to port it to a new KG. SemReL utilizes a path-based finding algorithm, which sometimes incorrectly predicts the relation semantic substructures of the AMR graph.

In this work, we propose a two-stage relation linking algorithm. First, to match the n-ary arguments in the Propbank framework with the binary predicates in the KB, we extract the substructure of the AMR graph as the relation representation and get all candidate relations corresponding to the linked entities. For an exacted relation representation, too many irrelevant relations will affect the retrieval efficiency of the relation linking model in the candidate relation space, so we use a cross-encoder to remove irrelevant candidate relations. Further, for relation disambiguation, we introduce question information to assist the model in selection. The source code is available at GitHub[1].

The main contributions of this work are as follows:

- Improving a rule-based approach to extract substructures containing relation information in the semantic parsing graph generated from an AMR parser[9].
- Adopting a cross-encoder model, which reduces the irrelevant candidate relations. So that the simple, knowledge graph agnostic neural model can also achieve good performance without large training data.
- Utilizing dual-encoder with different parameters to extract question features and relation features respectively, and fuse the information of two parts to predict the final relation.
- Experimental evaluations using four datasets based on DBpedia [10] and Wikidata [11]. We show that the method outperforms existing systems on all datasets.

2 RELATED WORK

In the KBQA, relation linking has been shown to be the main error propagation subtask and needs significant improvement. SIBKB [12] uses PATTY [13] as the underlying knowledge source, and performs relation linking based on the semantic similarity of the words with DBpedia predicates. EERL [14] proposes using question entities to support relation linking tasks on DBpedia, it logically connects properties to the target entities, and uses this property list to expand the set of relation candidates which can be used for the construction of SPARQL queries in the QA pipeline. The ReMatch [15] models KB relations with their underlying parts of speech, and then uses the additional attributes obtained from Wordnet [16] and Dependency parsing features to enhance the model. They all generate candidates by running a textual similarity-based method over dictionaries, which is constructed by analyzing the natural language patterns contained in massive text corpora through frequent item mining or crowdsourcing.

Falcon [3] and Falcon 2.0 [17] uses a series of steps to jointly link entities and relations in a question, they enhance entity and relation linking through cross-KG entity and relation alignment and basic principles of English morphology. EARL [18] is also a method for joint entity and relation linking, which adopts approximate generalized traveling salesman problem (GTSP) solvers and machine learning methods. KB-Pearl [4] is another system that performs joint entity and relation linking with Wikidata. It uses OpenIE to create a semantic graph of text and maps both entities and relations to a given KB.

SLING [7] leverages semantic parsing methods to understand the question and integrates some approaches (e.g. distantly supervised learning, statistical mapping, word embedding) to achieve state-of-the-art performance on various DBpedia datasets. SemReL [8] employs a simple transformer-based neural model for relation linking that leverages the AMR semantic parse of a sentence.

3 METHODS

3.1 Overview

We propose SPaReL which uses the semantic substructure of a sentence to process relevant relations retrieved from the underlying KB. Figure 2 depicts the overall architecture of our relation linking system.

Firstly, a natural language question is transformed into the AMR graph through a AMR parser [9], which also completes the entity linking task. Then we use the linked entity to search all corresponding candidate relations in the KB. According to some rules, the slot is extracted from the shortest path in the AMR graph from the entity node to the target node. The cross-encoder obtains the ranking score of the relations from the combined input of slots and candidate relations. In addition, a dictionary is generated from all slots in the training set, each unique slot is set as a key, and the value contains relations that have appeared in the training set corresponding to the slot. This dictionary can compensate for some candidate relations that the cross-encoder may ignore. The new candidate relation set is the union of these two parts. We use a pseudo-siamese network based on a dual-encoder to predict the gold KB relation. The model integrates question information to select the final result from the new relation set.

Our SPaReL system framework is shown in Figure 2. The left side shows the extraction of the slot from the AMR graph, and the combination of the slot with the question and the relations respectively, as the input of the two models. The blue words on the right represent candidate relations, where the gold relation is marked with black bold font.

3.2 Relation Slot Prediction

Semantic representations are abstracted from lexical forms and can provide more consistent structural clues than surface text. AMRs are directed acyclic graphs that capture who is doing what to whom by using PropBank frames to represent the semantic structure of a sentence. Nodes in the graph are concepts, and edges are labeled with relations between these concepts.

A path-based method is proposed in [19] to obtain the semantic substructure of AMR graphs, it finds the shortest path between the linked entity node and the *amr-unknown* node, and then deletes the irrelevant nodes, compresses the path, and extracts the slot. However, for a short path with a single predicate, after filtering out some irrelevant nodes, there is only one predicate node left in the slot. Propbank predicates are not equivalent to relations, the representation of relations requires the predicate node to combine the information of the neighbor nodes and the edges in the graph. In

[1] https://github.com/OBriennnnn/SPaReL

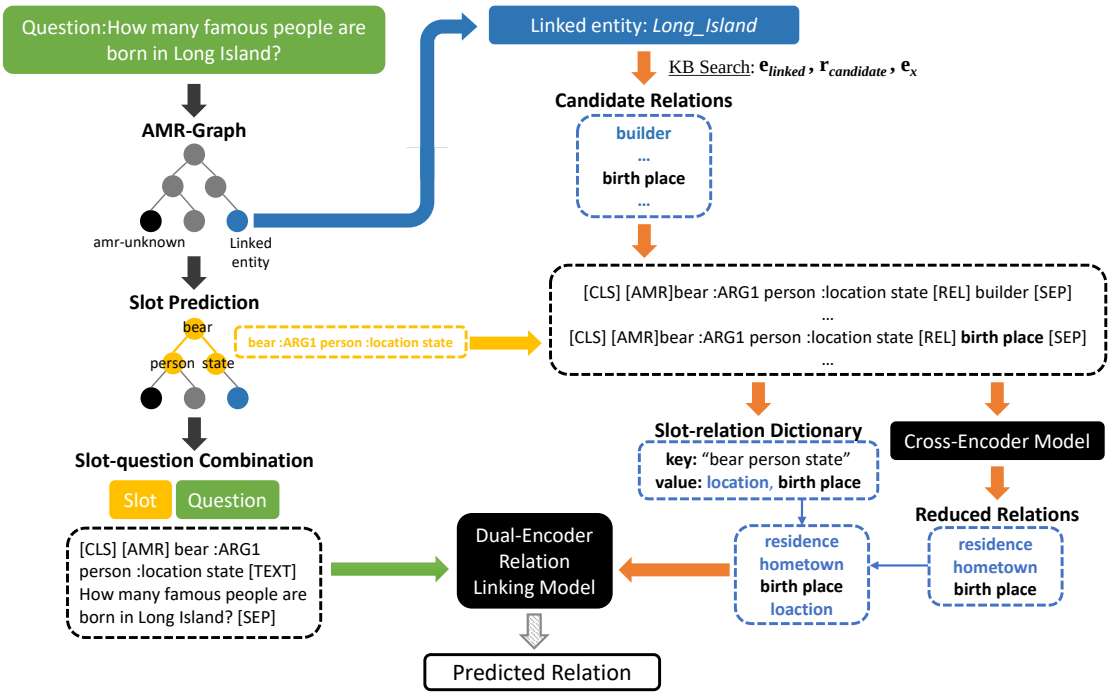

Figure 2: The architecture of the SPaReL system.

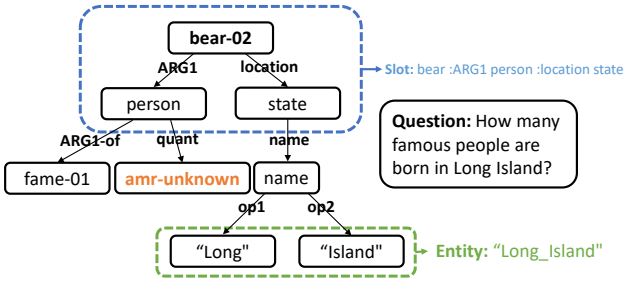

Figure 3: AMR graph of a general question.

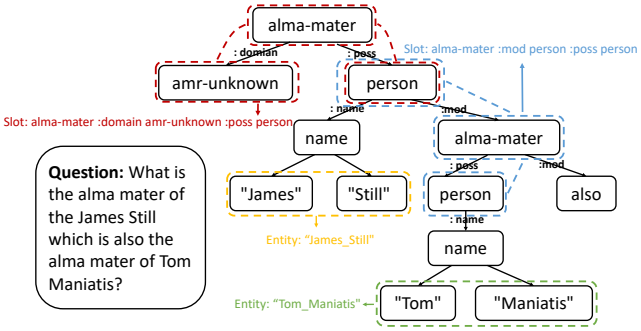

Figure 4: AMR graph of a multiple entity question.

addition, the results of AMR parsing are not always correct, namely, the underlying relations are not always represented in the form of predicates. To better represent relation information in AMR graphs, we propose two new rules:

- For a single predicate in the shortest path of the AMR graph, we consider the *amr-unknown* node as a special placeholder, which also has an implicit relation meaning and can be included in the slot. After AMR parsing, the predicate node needs to fuse edge information and neighbor node information to enrich the meaning of the relation.
- For a path that does not contain any predicates, we select a node with two non-core role edges as the center for slot generation. If there are multiple nodes in a path that satisfy this condition, the center of the relation semantic substructure corresponding to the entity should be the node closest to it.

Figure 3 shows an AMR graph of the general sentence "*How many famous people are born in Long Island?*" This is the simplest and most regular case. A special node, *amr-unknown*, is used to represent a placeholder for the answer to the question. AMR can mark named entity nodes, which are linked to KB entities by BLINK

entity linker [20]. We remove the sense label from the above slot representations (such as *bear-02* becomes *bear*) and convert them into the linearized representations through a top-down manner. As above, we get the entity's slot: *(bear|ARG1|person|location|state)*, which corresponds to the KB relations: *dbo: birthPlace*.

Figure 4 shows an AMR graph of the sentence "*What is the alma mater of the James Still which is also the alma mater of Tom Maniatis?*" with multiple entities. This sentence is a good illustration of our rules. The slot generated by the first rule is: *(alma-mater|domain|amr-unknown|poss|person)*. This slot contains the *amr-unknown* node because there are only two valid nodes between the entity *James_Still* and the target node *amr-unknown* (the *name* node is used as the aggregate representation of the entity nodes and is not included). It can be seen that the edge from the *alma-mater* node to the *amr-unknown* node is *:domain*, which implies the attribute information contained in the *amr-unknown* node.

The slot generated by the second rule is: *(alma-mater|mod|person| poss|person)*. There is no predicate node (such as *bear-02* in Figure 3) in the path: person→alma-mater→person→alma-mater→amr-unknown. According to our rules, the nodes can be used as the center node is *alma-mater* (in the first layer), *person* (in the second layer), *alma-mater*(in the third layer). We select the node *alma-mater* (in the third layer) closest to the entity as the center node to generate the slot. For nodes with the same content in the path (both are *alma-mater*), we choose the node in the third layer which is closer to the entity *Tom_Maniatis*.

Finally, we get the slots: *(alma-mater|domain|amr-unknown|poss| person)* and *(alma-mater|mod|person| poss|person)*, they both correspond to the KB relations: *dbo: almaMater*. Such representations will be concatenated with relations and sentences respectively, which are used as the input of different models.

Through processing, our rules fix slot length to 5 words, which facilitates the construction of a training set for cross-encoder learning.

3.3 Candidate Relation Reduction

For each linked entity in the AMR graph, we use a SPARQL query to find all relations associated with it. However, the number of candidate relations obtained in this way will be very large, some irregular relations have an adverse effect on the training of the model. Therefore, we use a cross-encoder to reduce irrelevant candidate relations.

We assume that slots containing PropBank predicates can represent relations, highly relevant candidate relations can be predicted by introducing deep cross attention between the slot and the relations. Our cross-encoder is similar to the ones described by [21]. The slot-relation representation $\tau_{s,r}$ is made up of word-pieces of the slot and relation. The input to our cross-encoder model is:

[CLS] [AMR] slot [REL] relation [SEP]

where [AMR] and [REL] are special tokens to separate slot and relation representation. Formally, we use $\boldsymbol{v}_{s,r}$ to denote our slot-relation embedding:

$$\boldsymbol{v}_{s,r} = \text{red}\left(T_{cross}\left(\tau_{s,r}\right)\right) \tag{1}$$

where $\tau_{s,r}$ is the input representation of slot and relation, T_{cross} is a BERT transformer model. Following the explanation in [22], we choose $red(.)$ to to get the first output in the last layer (corresponding to the special token [CLS]). To score relation candidates, a feed-forward layer W is applied to the embedding $\boldsymbol{v}_{s,r}$:

$$S_{\text{cross}}\ (s, r) = \boldsymbol{v}_{s,r}\boldsymbol{W} \tag{2}$$

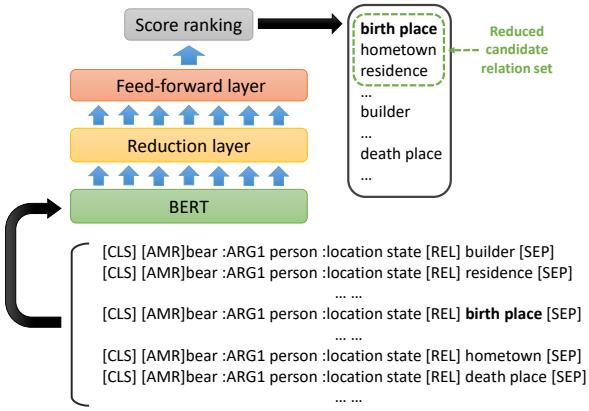

Figure 5: The overall structure of the cross-encoder ranking model.

Figure 5 shows the overall structure of the cross-encoder ranking model. Slot-relation pairs are tokenized using BERT tokenizer without any additional pre-processing. The slot obtained from the AMR graph of the question: "*How many famous people are born in Long Island?*" and the candidate relations obtained from the KB entity *Long_Island* are combined, then input to the cross-encoder model to get a reduced candidate relation set.

Training. We select all slots and their corresponding gold relations as the positive instance. For the generation of the negative instance of slot-relation pairs, we choose $\boldsymbol{n}$ non-gold relations randomly rather than all negative relations, in this work we choose $\boldsymbol{n} = 5$ for LC-QuAD 1.0 and LC-QuAD 2.0, and $\boldsymbol{n} = 15$ for QALD-7 and QALD-9. The scale of the QALD dataset is small, so it is necessary to set a larger $\boldsymbol{n}$. We find that too many negative slot-relation pairs $\boldsymbol{n} \geq 20$ cause data imbalance , which is not conducive to the training of the model. The label of the positive instance is 1, and the negative is 0, the training objective is to minimize the MSE loss between the predicted vector output and the true label.

Inference. As shown in Figure 5, in inference we connect the slot with all the relations of the linked entity, then the model ranks the scores for each combination, and finally selects the top $\boldsymbol{k}$ relations with the highest score. The choice of the hyperparameter $\boldsymbol{k}$ is introduced in Section 4.2.

3.4 Dual-encoder Relation Linking Model

In this section, we propose a pseudo-siamese neural network that does not share parameters for relation linking tasks. We use the pre-trained BERT model to initialize two encoder parameters, and the inputs of the two encoders are:

[CLS] [AMR] slot [TEXT] question [SEP]

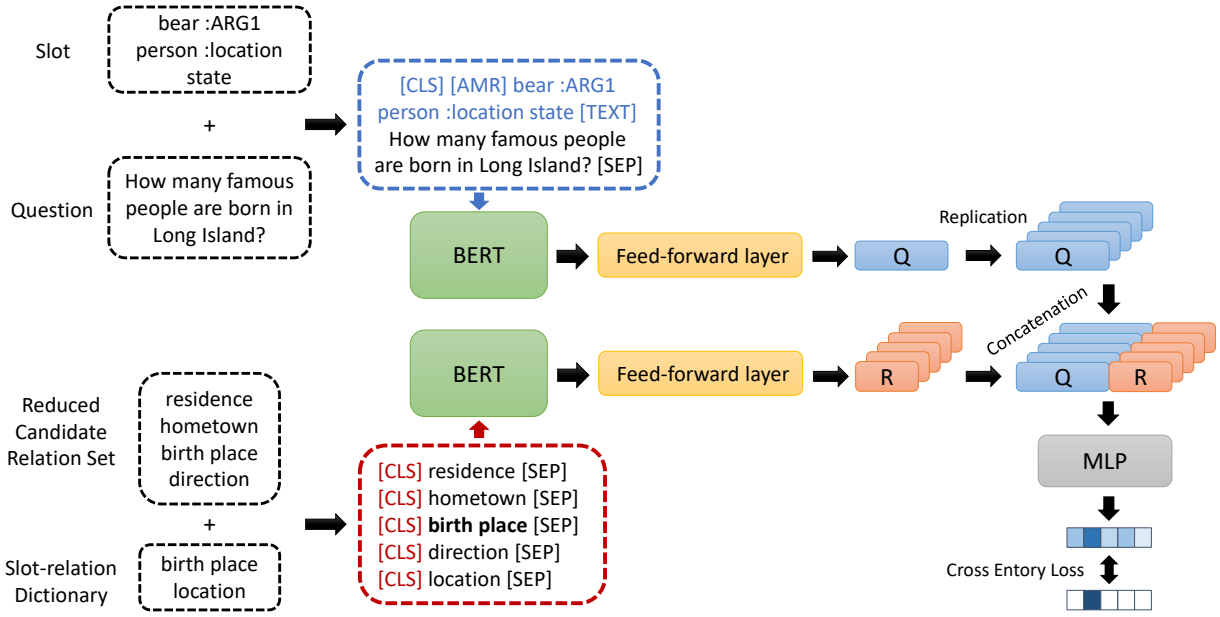

Figure 6: The architecture of the relation linking model.

[CLS] relation [SEP]

where [TEXT] is a new special token to separate slot and question representation. The question embedding and relation embedding are denoted by:

$$v_q = \text{red}\left(T_{\text{question}}\ (\tau_q)\right) \tag{3}$$

$$v_r = \text{red}\left(T_{\text{relation}}\ (\tau_r)\right) \tag{4}$$

where τ_q and τ_r are the input representation of slot-question sequence and relation sequence. For the question encoder, similar to in Section 3.3, $T_{question}$ and $T_{relation}$ are the BERT transformer model. $red(.)$ also get the first output in the last layer.

After obtaining the final vector representation of the two encoders, we concatenate the two vectors together. For each relation that aggregates sentence information, we use MLP to calculate the relation similarity score. Finally, the relation with the highest score is selected as the prediction. Due to the difference in the input of question and relation, the parameters between the two encoders are not shared. The structure of the model is shown in Figure 6.

Slot-relation Dictionary Generation. To supplement the cross-encoder model ignoring some high-correlation relations, we use the slots of the training set and their corresponding gold relations to generate a dictionary, with the slots as keys and the relations as values. As shown in Figure 6, for slot "*bear :ARG1 person :location state*" there are two relations: *birth place* and *location*. We merge these two relations into a reduced candidate relation set obtained by the cross-encoder.

Training. For a question q with vector representation v_q and a relation r with vector representation v_r, the predicted score is:

$$S(q, r) = MLP\left(v_q, v_r\right) \tag{5}$$

The training objective is to minimize cross-entropy loss between the one-hot gold truth and the vector of predicted scores.

Inference. For an input question, the corresponding set of candidate relations consists of two parts: the reduced candidate relations with high relevance provided by the cross-encoder and the relations generated by the slot-relation dictionary. We feed each question and its corresponding candidate relation set to two encoders separately, which output the final highest relation prediction score derived by the MLP.

4 EVALUATION

4.1 Datasets and Experiment Setup

Datasets. Our model is evaluated on four datasets based on two KBs, DBpedia and Wikidata. Each question in these datasets comes with a SPARQL query, which allows us to extract entities and corresponding gold relations. The four datasets are: QALD-7 [23], QALD-9 [24], LC-QuAD 1.0 [25], LC-QuAD 2.0 [26]. The QALD-7 dataset is based on Wikidata with 215 train questions and 50 test questions in natural language. The QALD-9 dataset is based on DBpedia with 413 train questions and 150 test questions in natural language. The LC-QuAD 1.0 dataset is based on DBpedia with 4000

Table 1: HR for each dataset with different k values

	QALD-7	QALD-9	LC-QuAD 1.0	LC-QuAD 2.0
k=5	0.52	0.48	0.86	0.88
k=10	0.63	0.55	0.94	0.94
k=15	0.78	0.63	0.96	0.96
k=20	0.92	0.72	0.97	0.97

train questions and 1000 test questions. The LC-QuAD 2.0 dataset is based on Wikidata with 24k train questions and 6046 test questions.

Experiment Setup. We use the BERT tokenizer to tokenize relation names without any additional preprocessing. Moreover, special tokens [AMR], [REL], [TEXT], and AMR relation labels are added to the BERT vocabulary. We use pre-trained BERT to encode different input sequences, and the parameters in each encoder are different.For the tokenizer of the cross-encoder, the length of the input sequence is limited to 15 tokens since only slots and relations are involved, and for the dual-encoder, the length of question encoder is set to 50 tokens, and the length of relation encoder is set to 15 tokens.

For the cross-encoder, the batch size is set to be 50, the initial learning rate is tried {5e-4, 2e-4, 5e-5, 2e-5}, and the learning rate decays every 5 epochs. The multiplicative factor gamma for updating the learning rate is set to be 0.2. For dual-encoders, the learning rate is updated similarly.

4.2 Cross-encoder Model Performance

Cross-encoder can capture the rich interactions between slots and relations well. We employ Hit Ratio (HR) as the evaluation metric, we consider a hit if the gold relation is among the top k candidate relations predicted, and HR is the proportion of the total number of hits in the total number of questions. For each gold relation, we fetch a list of top candidates with a k value of 5, 10, 15, or 20, where k is the number of the generated candidates. Table 1 presents our experimental results. A higher HR means selecting too many candidate relations, which may lead to lower performance of relation linking. For different datasets, we balance the relationship between HR and the number of candidate relations input into the relation linking model to achieve optimal results.

For dataset QALD-7, we choose k=20, for dataset QALD-9, we choose k=15, for dataset LC-QuAD 1.0 and LC-QuAD 2.0, we choose k=10. The method of k value selection is: for LC-QuAD 1.0 and LC-QuAD 2.0, the scale of the training set is large, and the model can be well trained, so k does not need to be large. However, for QALD-7 and QALD-9, the scale of the training sets is small, to appropriately expand the likelihood that the gold relation is included in the reduced candidate relations, the k value is set to be larger.

4.3 Results and Analysis

We choose precision, recall, and F1 score to evaluate the results. The settings of P, R, and F1 refer to [7], for each question:

$$P = \frac{|\,gold\ relations\,| \cap |\,predicted\ relations\,|}{|\,predicted\ relations\,|} \tag{6}$$

Table 2: Relation linking results on QALD-7, QALD-9, and LC-QuAD 1.0

	QALD-7			QALD-9			LC-QuAD 1.0		
	P	R	F1	P	R	F1	P	R	F1
SIBKB	0.29	0.31	0.30	-	-	-	0.13	0.15	0.14
ReMatch	0.31	0.34	0.33	-	-	-	0.15	0.17	0.16
EARL	0.27	0.28	0.27	-	-	-	0.17	0.21	0.18
Falcon	0.58	0.61	0.59	0.23	0.23	0.23	0.42	0.44	0.43
SLING	0.57	**0.76**	0.65	0.39	**0.50**	0.44	0.41	0.55	0.47
SemReL	-	-	-	0.46	0.44	0.45	0.51	0.51	0.51
SPaReL	**0.70**	0.67	**0.68**	**0.53**	0.45	**0.49**	**0.68**	**0.58**	**0.62**

$$R = \frac{|\,gold\ relations\,| \cap |\,predicted\ relations\,|}{|\,gold\ relations\,|} \tag{7}$$

$$F1 = \frac{2 \times P \times R}{P + R} \tag{8}$$

and the final result is the average of P, R, F1 for all the questions.

Tables 2 and 3 compare SPaReL with existing approaches. In Table 2, We compare our work with other methods, all of them support relation linking over DBpedia [10]. SIBKB [12] and Rematch [15] build a dictionary by analyzing the natural language patterns contained in massive text corpora through frequent item mining or crowdsourcing and then generate candidates through textual similarity-based method over this dictionary. EARL [18] leverages the connection density of KG for ranking, Falcon [3] enhances relation linking through basic principles of English morphology. Both SLING [7] and SemReL [8] leverage AMR for preprocessing, train a Transformer-based model. Falcon and SLING differ from the results of QALD-9 in their paper because they are evaluated using both training and test sets, and we re-evaluate them only for the test set. SPaReL outperforms all baselines in all benchmarks with respect to the F1 score. Most of the current relation linking evaluation datasets are based on the KB of DBpedia, so most of the current methods are evaluated on this KB.

LC-QuAD 2.0 builds SPARQL queries on Wikidata [11], Falcon 2.0 [17] uses several fundamental principles of English morphology to obtain auxiliary information, and exploits an alignment model for linking, SemReL [8] is also evaluated on this dataset.

From Table 2, it can be seen that the precision of the model has been significantly improved, but on the datasets QALD-7 and QALD-9, the recall of the model does not surpass the previous model. The reason for this phenomenon is that to evaluate the performance of the model for different data sets, we only use their own training sets to the model, instead of introducing a large amount of other data to assist model training like SLING and SemReL. Therefore, for the QALD-7 and QALD-9 datasets with less training data, the model does not learn many corresponding slot representations for some gold relations, so the inference ability is not significantly improved. This problem is solved when the scale of training data is getting larger. For example, LC-QuAD 1.0 in Table 2 and LC-QuAD 2.0 in Table 3, the recall surpasses other models. It shows that the model has improved the judgment ability of some gold relations.

We analyze some errors in the experiments for future research. Since the shortest paths are all from the entity node to the *amr-unknown* node, the correctness of the node position and entity

Table 3: Relation linking results on LC-QuAD 2.0.

	LC-QuAD 2.0		
	P	R	F1
Falcon 2.0	0.44	0.37	0.40
SemReL	0.59	0.38	0.46
SPaReL	**0.69**	**0.46**	**0.55**

Table 4: Ablation study on LC-QuAD 1.0 and QALD-9 test set

	LC-QuAD 1.0	QALD-9
SPaReL	0.62	0.49
w/o slot-relation dictionary relations	0.59	0.37
w/o AMR slot	0.57	0.44
w/o candidate relations reduction	0.45	0.41

linking results are both crucial. In addition, datasets have some problems. For example, in LC-QuAD 1.0, some candidate relations are overlapping (such as *death place* and *place of death*, *home town* and *hometown*), which is also difficult for humans to distinguish these relations. The coexistence of singular and plural formats of some candidate relations (such as *region* and *regions*) also affects the model's inference ability. In LC-QUAD 2.0, we notice that some of the questions do not match their SPARQL queries. Wikidata has evolved significantly since 2019, the time LC-QuAD 2.0 was created.

4.4 Ablation Study

Table 4 shows the F1 scores of ablation experiments on the LC-QuAD 1.0 and QALD-9 test sets. Slots can improve the model's attention to relations, reducing low-relevance candidate relations and supplementing slot-relation dictionary relations both have an impact on the performance of the system.

W/o slot-relation dictionary relations. The inference result of the cross-encoder model does not include all the gold relations, so we establish a slot-relation dictionary to complement candidate relations. However, if the scale of the training set is large enough, the improvement effect of this method is not obvious (such as LC-QuAD 1.0).

W/o AMR slot. For the relation linking model based on dual-encoder, we do not concatenate AMR slot and question when inputting, but only input a single question to test the model's inference ability from the surface text. The performance has declined, because the sentence may contain multiple relations. It is difficult for the model to determine what relation the sentence implies without an AMR slot.

W/o candidate relations reduction. If candidate relation filtering is not performed, for the LC-QUAD 1.0 test set, 48.36 candidate relations are corresponding to each entity on average, and 95.44 for the QALD-9 test set. Too many candidate relations will lead to the sparse distribution of the gold relation, which has a negative impact on the training and prediction of the model. After filtering by cross-encoder, 10.56 candidate relations are corresponding to each entity on average for the LC-QuAD 1.0 test set and 15.21 for the QALD-9 test set.

4.5 Relation Linking on Single-Relation and Multi-Relation Questions

Our slot extraction method can get a more comprehensive relation representation. To illustrate the improvement in relation reasoning in both single-relation questions and multi-relation questions, we construct two small-scale test sets for evaluation using the test dataset of LC-QuAD 1.0. Each single-relation sentence contains only one relation, and the length of the sentence is less than 8 words, each multi-relation question contains more than two relations (can be the same or different), and the length of the question is more than 12 words. The single-relation test set has 70 questions, and the multi-relation test set has 100 questions.

For single-relation questions, as described in Section 3.2, we use a slot containing an *amr-unknown* node to concatenate the sentences as input to the question encoder. In SemReL, the slot generation method simply concatenates the predicate with the question.

For multiple-relation questions, in SemReL, they follow the method in [19], adding all the potential predicate nodes and corresponding edges in the path to generate the slot, which is concatenated with questions for relation linking. They use special tokens [SP] and [EP] to mark the words corresponding to each predicate and use an AMR parser [27] for word-to-predicate alignment. In our work, we use improved rules to obtain more related information.

Table 5 lists two examples of the input for the two methods. For the single-relation question, after processing, there is only one node: *develop* in the path. SemReL only adds this single node, but we consider that the information attached to its neighbor nodes and edges is beneficial for inference as well. Adding neighbors and edges can provide richer information. For the multiple-relation question, the entity *dbr: C++* corresponds to the relation *programming language*. The shortest path from the entity to the *amr-unknown* node is: language→software→amr-unknown, which is no predicate node in the path (possibly due to AMR parsing errors). In SemReL, the corresponding slot cannot be generated because of no predicate. In our work, according to the description in Section 3.2, the center of the slot is formed by selecting the node *language* that satisfies the condition and is the closest to the entity node, we generate a regular slot through its neighbor nodes and corresponding edges. As can be seen, our method can still extract a slot that expresses relation information.

Incorrect AMR graphs lead to wrong alignment results, to prevent error propagation, we remove the special token used for alignment in the input. Moreover, the accuracy of alignment is difficult to evaluate on the KBQA dataset. Our method fixes the slot length to 5 words, which builds the premise for reducing candidate relations through the cross-encoder and constructing a slot-relation dictionary.

The results are shown in Table 6, our method outperforms the state-of-the-art model SemReL in F1 scores on both small datasets. For single-relation questions, the slot emphasizes the components containing relation information; for multiple-relation questions, the slot assists in abstracting and selecting relation information.

Table 5: Input of single-relation questions and multi-relation questions

	single-relation question	multiple-relation question
SemReL	**developer:** [CLS] [AMR] develop [TEXT] what has been [SP] developed [EP] by John Fanning? [SEP]	**programming language:** There is no predicate in the shortest path, no slot is generated **operating system:** [CLS] [AMR] operate :instrument system :poss company [TEXT] What is the total number of software whose programming language is C++ and [SP] operating [EP] system is Microsoft Windows? [SEP]
SPaReL	**developer:** [CLS] [AMR] develop :ARG0 person :ARG1 amr-unknown [TEXT] what has been developed by John Fanning? [SEP]	**programming language:** [CLS] [AMR] language :ARG3 program :mod software [TEXT] What is the total number of software whose programming language is C++ and operating system is Microsoft Windows? [SEP] **operating system:** [CLS] [AMR] operate :instrument system :poss company [TEXT] What is the total number of software whose programming language is C++ and operating system is Microsoft Windows? [SEP]

Table 6: Relation linking results on single-relation questions and multiple-relation questions

	single-relation questions	multiple-relation questions
SemReL	0.68	0.38
SPaReL	0.72	0.50

5 CONCLUSIONS AND FUTURE WORK

In this paper, we propose the SPaReL model that exploits the semantic structure of sentences. Compared to existing systems, the rules we adopt can better express relation information, and our regular semantic information is beneficial to the irrelevant relation filter and relation linking. Experiments show that our method can adapt to multiple KGs and achieves state-of-the-art performance.

We also notice that when the training data scale is small, due to lack of training, the cross-encoder is difficult in inferring all high-relevance relations. In the future, we will explore new methods to obtain this kind of few-shot relations. Furthermore, our model still relies on the rule-based AMR graph slot-finding algorithm, and its identification is not always correct for some complex AMR graphs. We will explore learning algorithms to identify slots from the graph in the future.

ACKNOWLEDGMENTS

The work is supported by the National Key Research and Development Program of China (No. 2019YFB2101802-02) and the Natural Science Foundation of China (No. 61502095).

REFERENCES

[1] Dennis Diefenbach, Vanessa Lopez, Kamal Singh, and Pierre Maret. Core techniques of question answering systems over knowledge bases: a survey. *Knowledge and Information systems*, 55(3):529–569, 2018.

[2] Isaiah Onando Mulang, Jennifer D'Souza, and Sören Auer. Fine-tuning bert with focus words for explanation regeneration. In *Proceedings of the Ninth Joint Conference on Lexical and Computational Semantics*, pages 125–130, 2020.

[3] Ahmad Sakor, Isaiah Onando Mulang, Kuldeep Singh, Saeedeh Shekarpour, Maria Esther Vidal, Jens Lehmann, and Sören Auer. Old is gold: linguistic driven approach for entity and relation linking of short text. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 2336–2346, 2019.

[4] Xueling Lin, Haoyang Li, Hao Xin, Zijian Li, and Lei Chen. Kbpearl: a knowledge base population system supported by joint entity and relation linking. *Proceedings of the VLDB Endowment*, 13(7):1035–1049, 2020.

[5] Jungwoo Lim, Dongsuk Oh, Yoonna Jang, Kisu Yang, and Heuiseok Lim. I know what you asked: Graph path learning using amr for commonsense reasoning. *arXiv preprint arXiv:2011.00766*, 2020.

[6] Paul R Kingsbury and Martha Palmer. From treebank to propbank. In *LREC*, pages 1989–1993. Citeseer, 2002.

[7] Nandana Mihindukulasooriya, Gaetano Rossiello, Pavan Kapanipathi, Ibrahim Abdelaziz, Srinivas Ravishankar, Mo Yu, Alfio Gliozzo, Salim Roukos, and Alexander Gray. Leveraging semantic parsing for relation linking over knowledge bases. In *International Semantic Web Conference*, pages 402–419. Springer, 2020.

[8] Tahira Naseem, Srinivas Ravishankar, Nandana Mihindukulasooriya, Ibrahim Abdelaziz, Young-Suk Lee, Pavan Kapanipathi, Salim Roukos, Alfio Gliozzo, and Alexander Gray. A semantics-aware transformer model of relation linking for knowledge base question answering. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 2: Short Papers)*, pages 256–262, 2021.

[9] Michele Bevilacqua, Rexhina Blloshmi, and Roberto Navigli. One spring to rule them both: Symmetric amr semantic parsing and generation without a complex pipeline. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 35, pages 12564–12573, 2021.

[10] Jens Lehmann, Robert Isele, Max Jakob, Anja Jentzsch, Dimitris Kontokostas, Pablo N Mendes, Sebastian Hellmann, Mohamed Morsey, Patrick Van Kleef, Sören Auer, et al. Dbpedia–a large-scale, multilingual knowledge base extracted from wikipedia. *Semantic web*, 6(2):167–195, 2015.

[11] Denny Vrandečić and Markus Krötzsch. Wikidata: a free collaborative knowledgebase. *Communications of the ACM*, 57(10):78–85, 2014.

[12] Kuldeep Singh, Isaiah Onando Mulang', Ioanna Lytra, Mohamad Yaser Jaradeh, Ahmad Sakor, Maria-Esther Vidal, Christoph Lange, and Sören Auer. Capturing knowledge in semantically-typed relational patterns to enhance relation linking. In *Proceedings of the Knowledge Capture Conference*, pages 1–8, 2017.

[13] Ndapandula Nakashole, Gerhard Weikum, and Fabian Suchanek. Patty: A taxonomy of relational patterns with semantic types. In *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, pages 1135–1145, 2012.

[14] Jeff Z Pan, Mei Zhang, Kuldeep Singh, Frank van Harmelen, Jinguang Gu, and Zhi Zhang. Entity enabled relation linking. In *International Semantic Web Conference*, pages 523–538. Springer, 2019.

[15] Isaiah Onando Mulang, Kuldeep Singh, and Fabrizio Orlandi. Matching natural language relations to knowledge graph properties for question answering. In *Proceedings of the 13th International Conference on Semantic Systems*, pages 89–96, 2017.

[16] Christiane Fellbaum. Wordnet: An electronic lexical database cambridge. *MA: MIT Press*, 1998.

[17] Ahmad Sakor, Kuldeep Singh, Anery Patel, and Maria-Esther Vidal. Falcon 2.0: An entity and relation linking tool over wikidata. In *Proceedings of the 29th ACM International Conference on Information & Knowledge Management*, pages 3141–3148, 2020.

[18] Mohnish Dubey, Debayan Banerjee, Debanjan Chaudhuri, and Jens Lehmann. Earl: joint entity and relation linking for question answering over knowledge graphs. In *International Semantic Web Conference*, pages 108–126. Springer, 2018.

[19] Pavan Kapanipathi, Ibrahim Abdelaziz, Srinivas Ravishankar, Salim Roukos, Alexander Gray, Ramon Astudillo, Maria Chang, Cristina Cornelio, Saswati Dana, Achille Fokoue, et al. Leveraging abstract meaning representation for knowledge base question answering. *arXiv preprint arXiv:2012.01707*, 2020.

[20] Ledell Wu, Fabio Petroni, Martin Josifoski, Sebastian Riedel, and Luke Zettlemoyer. Scalable zero-shot entity linking with dense entity retrieval. *arXiv preprint arXiv:1911.03814*, 2019.

[21] Lajanugen Logeswaran, Ming-Wei Chang, Kenton Lee, Kristina Toutanova, Jacob Devlin, and Honglak Lee. Zero-shot entity linking by reading entity descriptions. *arXiv preprint arXiv:1906.07348*, 2019.

[22] Samuel Humeau, Kurt Shuster, Marie-Anne Lachaux, and Jason Weston. Poly-encoders: Transformer architectures and pre-training strategies for fast and accurate multi-sentence scoring. *arXiv preprint arXiv:1905.01969*, 2019.

[23] Ricardo Usbeck, Axel-Cyrille Ngonga Ngomo, Bastian Haarmann, Anastasia Krithara, Michael Röder, and Giulio Napolitano. 7th open challenge on question answering over linked data (qald-7). In *Semantic web evaluation challenge*, pages 59–69. Springer, 2017.

[24] Ngonga Ngomo. 9th challenge on question answering over linked data (qald-9). *language*, 7(1):58–64, 2018.

[25] Priyansh Trivedi, Gaurav Maheshwari, Mohnish Dubey, and Jens Lehmann. Lc-quad: A corpus for complex question answering over knowledge graphs. In *International Semantic Web Conference*, pages 210–218. Springer, 2017.

[26] Mohnish Dubey, Debayan Banerjee, Abdelrahman Abdelkawi, and Jens Lehmann. Lc-quad 2.0: A large dataset for complex question answering over wikidata and dbpedia. In *International semantic web conference*, pages 69–78. Springer, 2019.

[27] Ramón Fernandez Astudillo, Miguel Ballesteros, Tahira Naseem, Austin Blodgett, and Radu Florian. Transition-based parsing with stack-transformers. *arXiv preprint arXiv:2010.10669*, 2020.