Accelerating KBQA via Logical-Question Bidirectional Reranking

Anonymous ACL submission

Abstract

Semantic parsing based on large language models aims to transform natural language questions into logical forms to support the generation of answers. Although beam search-based decoding strategies are widely adopted to ensure the golden logical form appears in candidate lists, the golden logical form often fails to rank first, which raises execution time and answer error rate. To solve this problem, we propose RankKBQA, a flexible plugin that optimizes for speed and effectiveness in KBQA via a logical-question bidirectional reranking framework. Specifically, RankKBQA first converts generated logical forms into corresponding questions via a fine-tuned PLM-based transcriber, and then measures question similarity with the original input, which obtains the second logical form sorting list. Finally, we utilize a bidirectional reranking algorithm to merge the original sorting with the new sorting. Through the above steps, the proposed framework raises the golden logical form ranking list, simultaneously improving execution efficiency (most +42.1 speedup) and QA accuracy (most +2.9 F1) by reducing the candidate search space. Our code is available at https://anonymous.4open.science/r/RankKBQA-F702/.

1 Introduction

Knowledge base question answering (KBQA) (Yih et al., 2015) has been used to explain and respond to users' queries with a large amount of stored knowledge (Bollacker et al., 2008; Vrandečić and Krötzsch, 2014) for a long time, which has potential applications in many fields, and has once become the focus of academic and industrial research. As one of the core methods of knowledge-based question answering, semantic parsing aims to effectively transform natural language questions into logical forms to support the generation of answers, and has achieved great results in recent years. The initial work is to translate the question into an intermediate logical form before execution (Yih et al., 2015), such as SPARQL or S-expressions.

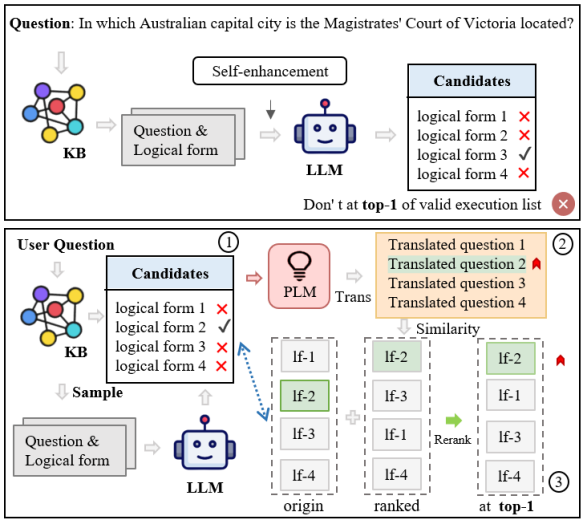

Figure 1: Comparison of our proposed method with existing methods, in brief, our method addresses the high accuracy and efficiency requirements of KBQA systems for top-1 results by transcription and reranking.

Recently, large language models (LLMs), such as Codex (Chen et al., 2021) and GPT-4 (Achiam et al., 2023), have demonstrated powerful in-context learning capabilities, which can complete complex reasoning tasks based on target questions after learning from few-shot pairs of examples ⟨question, logical form⟩ (Gao et al., 2023; Chen et al., 2022). The previous methods are usually divided into two phases, as shown in Figure 1 Top. First, a list of candidate logical forms is generated through LLMs, and then an executable logical expression is obtained through entity linking and relation linking. The self-consistency principle (Wang et al., 2022) is used in answer selection; that is, the majority voting strategy is used to determine the final answer after all candidate expressions are executed. Or use the principle of first validity (Ye et al., 2022; Luo et al., 2024a), which means that candi-

date expressions are executed sequentially with the first valid answer as the final answer. The high accuracy of most voting strategies is in exchange for a large number of unreliable candidates, resulting in increased runtime, which typically requires querying thousands of SPARQL queries and taking a few minutes to get an answer. Applications that use a LLM as a base model cannot afford such a waste of resources and response speed; Although the first validity principle reduces the time cost caused by the self-consistency principle to some extent, it is inevitable that the golden logical form in the list of candidate logical forms is not in the first place of the effective execution list. This reduces the real power of LLMs to some extent.

In this work, we propose RankKBQA(Figure 1 Bottom), a new plug-in that addresses the high accuracy and efficiency requirements of KBQA systems for top-1 results by transcribing generated logical forms into natural language questions, serializing structured knowledge, and assisting in reranking candidate lists. Specifically, RankKBQA consists of three phases. In the first phase, we generate a list of candidate logical forms for the target question. Here we adopt two approaches: the fine-tuning approach and the in-context learning (ICL) approach. We generate candidate logical forms for the target question by fine-tuning the LLM based on the question and its corresponding logical forms (i.e., S-expressions) or as example pairs of LLM for the ICL approach. The results of the beam search show that more than 70% of the test questions match gold facts when converted into logical form. In the second phase, we introduce the reverse transcription-assisted methods (RTAM), which use a fine-tuned lightweight pre-trained language model (PLM) with the ability to perceive structured knowledge to generate natural language questions. Then, a semantic similarity algorithm is used to rerank the list of candidate logical forms according to the target question and the list of transcribed natural language questions. Considering that the preliminary generated list also contributes a lot, we take it as another clue and adopt the bidirectional reranking algorithm for comprehensive ranking. Finally, RankKBQA gets an executable SPARQL query after entity linking and relation linking. RankKBQA addresses the high-precision requirement of top-1 results for KBQA systems and further taps into the faster inference capabilities of LLMs.

To evaluate the validity of RankKBQA, we conducted experiments on two standard KBQA datasets, WebQSP (Yih et al., 2016) and complex webquestions (CWQ) (Talmor and Berant, 2018). The experimental results show that RankKBQA not only achieves competitive performance in the KBQA task, but can reduce the run time of RankKBQA by more than 40% compared to the most advanced model. These results demonstrate the reliability and efficiency of our approach.

2 Related Work

2.1 LLM-based Agent method for KBQA

Agents built on the basis of LLMs demonstrate impressive reasoning abilities in a variety of downstream tasks. KD-CoT (Wang et al., 2023) overcomes illusion and error propagation by verifying and modifying inference trajectories in CoT through interaction with external knowledge; Interactive-KBQA (Xiong et al., 2024) develops three common APIs for the interaction between LLM and KBs to directly generate logical forms; QueryAgent (Huang et al., 2024) utilizes rich environmental feedback in intermediate steps to perform selective and differentiated self-correction.

2.2 Fine-tuning method for KBQA

Most state-of-the-art KBQA models are based on semantic parsing (Lan et al., 2021; Luo et al., 2024a), where a question is mapped onto a logical form over the KB. RNG-KBQA (Ye et al., 2022), ArcaneQA (Gu and Su, 2022), and DECAF (Yu et al., 2023) use sequence-to-sequence models to generate the complete S-expression and provide various enhancements to the semantic parsing process. Uni-Parser (Liu et al., 2022) and FC-KBQA (Zhang et al., 2023) introduce more fine-grained primitives to aid in the generation of logical forms; HGNet (Chen et al., 2023) generates query graphs by hierarchical autoregressive decoding; ChatKBQA (Luo et al., 2024a) uses a fine-tuned LLM to generate logical forms. RoG (Luo et al., 2024b) generates a relationship path based on KGs as a faithful plan.

2.3 Few-shot ICL method for KBQA

LLM has strong generalization ability, which makes the few-shot ICL method allow LLM to complete even complex inference tasks while observing a few labeled data (Cheng et al., 2023). KB-BINDER (Li et al., 2023) instructs Codex (Chen et al., 2021) to generate logical forms for the tar-

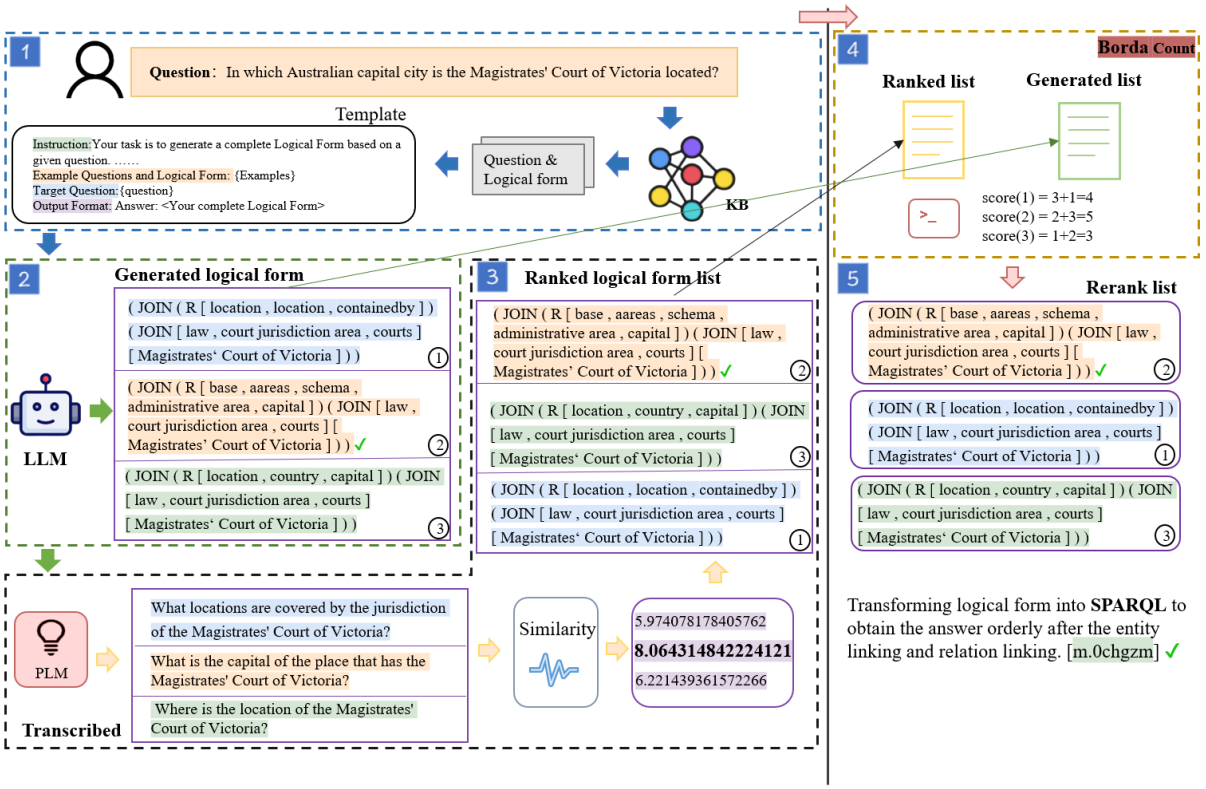

Figure 2: TRKBQA framework. Take the few-shot ICL method as an example. Firstly, based on the user's question, samples from the KB are provided to the LLM to generate multiple preliminary logical forms for the question. Next, a PLM transforms the generated logical forms into natural language questions, which are ranked by similarity to the target question. Finally, we will use the sorted list and the preliminary generated list as the two clues of Borda Count, and execute Borda Count to rerank.

get question. KB-Coder (Nie et al., 2024) finds that the LLM is more familiar with generating a code style than it is with generating structured queries, so it converts the generation process of an unfamiliar logical form into the more familiar code generation process through function calls. To further alleviate the illusion problem, ToG (Sun et al., 2024) introduces a novel approach called thinking on graphs, which guides the LLM to iteratively perform a beam search over the KG to discover the most promising inference paths. In addition, FlexKBQA (Li et al., 2024) introduces a lightweight model to further assist the generation of LLMs.

3 Preliminaries

Knowledge Base (KB). Given a KB $K = \{(s, r, o) \mid s \in \mathcal{E}, r \in \mathcal{R}, o \in \mathcal{E} \cup \mathcal{L}\}$, where $\mathcal{E}$ a set of entities, $\mathcal{L}$ a set of literals, and $\mathcal{R}$ a set of binary relations. Each entity $e \in \mathcal{E}$ in the entity set $\mathcal{E}$ has a MID (i.e., machine identifier), which is unique and has a surface name corresponding to it. (e.g., "m.08415" corresponds to "Washington Redskins"). Each relation $r \in \mathcal{R}$ in the set of relations $\mathcal{R}$ consists of multiple levels of labels, (e.g. r="sports.sports_team.location"). Besides, a literal $l \in \mathcal{L}$ is usually an integer, float, or datetime value.

Logical Form. Logical form is a structured representation of a natural language question in a KB. (e.g., SPARQL, query graph or S-expression). Following (Luo et al., 2024a), we use S-expressions to represent questions over KB. S-expression uses functions operating on set-based semantics and eliminates variable usages as in lambda DCS (Liang, 2013). This makes S-expression a suitable representation for the task of KBQA because it balances readability and compactness (Gu et al., 2021).

Program Execution. For the KBQA task, given a natural language question Q and a knowledge base $\mathcal{K}$, we first convert Q to the logical form $LF = SP(Q)$, where $SP(\cdot)$ is a semantic parsing function.Then, we convert the logical form into a SPARQL query $q = COV(LF)$ that can be executed on the KB, where $COV(\cdot)$ is a conversion function. Finally the set of answers $A = Exec(q \mid$

$\mathcal{K}$) is obtained by executing q against $\mathcal{K}$, where $Exec(\cdot)$ is a query execution function.

4 Methodology

4.1 Overview Of RankKBQA

RankKBQA is a three-stage KBQA framework (2), first generating, then sorting, and finally retrieving. Specifically, RankKBQA uses fine-tuning methods or few-shot ICL methods to generate candidate logical forms for the target question. Meanwhile, the pre-trained Language Model (PLM) was fine-tuned to enable it to transcribe logical forms into natural language questions, and obtain a new ranked list by semantic similarity matching. Finally, the bidirectional re-ranking algorithm was used to merge the original ranking and the new ranking. Finally, the logical form is transformed into a SPARQL query by knowledge base pattern matching and executed to obtain the final answer.

4.2 Candidates Generation

We employ two methods: fine-tuning and few-shot ICL, to generate candidate logical forms. For fine-tuning, we utilize LoRA (Hu et al., 2022), which employs low-rank matrix decomposition to optimize parameter updates, thereby significantly reducing computational and memory costs. Using this method, we fine-tune Llama-3-8B (Dubey et al., 2024) to endow it with natural-language-to-logical-form translation capabilities.

It is worth noting that the logical form in the dataset uses MID representation entities such as "m.08415", which is not conducive to reasoning and learning. Therefore, we replace MID with the surface name of the entity. Furthermore, relation labels are often multi-level (e.g., "sport.sports_team.location"), which adds to the difficulty of generation for LLMs, which we format as "sport, sport team, location". Finally, we wrap the entity and relationship in brackets, e.g.,"(join (R[sport, sport team, location])[Redskins])".

For the few-shot ICL method, we extract similar top-k questions from the dataset according to the target question, in the form of $\langle$question, logical form$\rangle$ pairs. Finally, we use generation instructions $Inst$, examples Ex, and target question Tq, building Prompt P to guide the LLM in generating a list of candidate logical forms C for the target question. The format is as follows:

$$C = \text{LLM}(P \mid Inst, Ex, Tq) \quad (1)$$

When we use beam search, more than 70% of the candidate logical forms match the golden facts, and these questions, when converted into SPARQL queries, can be correctly answered without large-scale retrieval replacement against the relationship.

4.3 Logical-question Bidirectional Reranking

After obtaining the list of candidate logical forms, we find that more than 21% of the lists of candidate logical forms in which golden logical forms appear do not appear at the top of the candidate list, leading to the occurrence of errors. After analyzing the data, we conclude that the golden logical form contains specific pattern information about the KB, and using it as an auxiliary clue may solve our problem. Based on this, we introduce the reverse transcription-assisted method (RTAM), which is based on the $\langle$logical form, question$\rangle$ pair in KB to fine-tune a PLM to have the perception ability from logical forms to natural language questions. Next, we use the model to transcribe the logical forms in the candidate list into natural language questions. Finally, we use the transcribed natural language question with the KB pattern information and the target question to calculate the similarity score s_i, formulated as follows:

$$s_i = \text{Reranker}(Tq, d_i) \quad (2)$$

According to the score, rank the candidate list so that the position of the gold logical form rises. formulated as follows:

$$C_{\text{sorted}} = \text{argsort}\left(\{s_1, \ldots, s_n\}, \text{desc=True}\right) \quad (3)$$

In addition, considering that the order in the preliminary candidate list is also of great value, we propose a logical-question bidirectional reranking strategy to balance the influence of the preliminary candidate list and the transcription list on the final answer. We use the Borda Count algorithm, which is a rank-based voting mechanism where the ranking in the list expresses the preference order for each candidate, and then the candidates are scored from highest to lowest and summed to obtain the final list of candidates based on the Borda Count score.

Specifically, as shown in Algorithm 1[1], the inputs are the preliminary candidate list P_{LF} and the transcription list $T_{LF} = \{(q_i, lf_i, s_i)\}$. First, we take the total length of the candidate list as the full

[1]https://en.wikipedia.org/wiki/Borda_count

Algorithm 1 Borda Count

Require: Predicted logical form list P_{LF}; Scored Transcribed list $T_{LF} = \{(q_i, lf_i, s_i)\}$

Ensure: Sorted list LF_{borda} based on Borda Count scores

$S_{borda} \leftarrow$ defaultdict(int)
for $lf_i \in P_{LF}$ **do**
  $rank \leftarrow$ index of lf_i in P_{LF}
  $S_{borda}[p_i] \leftarrow S_{borda}[p_i] + (|P_{LF}| - rank)$
end for
$S_{orted} \leftarrow$ sorted$(T_{LF}, s_i, \text{True})$
for $(q_i, lf_i, s_i)\} \in S_{orted}$ **do**
  $rank \leftarrow$ index of (q_i, lf_i, s_i) in S_{orted}
  $S_{borda}[p_i] \leftarrow S_{borda}[p_i] + (|P_{LF}| - rank)$
end for
$LF_{borda} \leftarrow$ sorted$(T_{LF}, S_{borda}[lf_i], \text{True})$
Return LF_{borda}

score $|P_{LF}|$, and then we traverse the candidate list in order to contribute a part of the Borda Count score S_{borda} with $|P_{LF}|$ minus the index of the candidate logical formP^i_{LF}. formulated as follows:

$$S_{borda}(P_{LF}) = \sum_{i=1}^{k} \left(|P_{LF}| - \text{rank}(P^i_{LF})\right) \quad (4)$$

For the transcription list, we rank it according to the similarity score s_i. We then traverse the sorted list S_{orted} in order, and the length of the candidate list is still used as the full score $|P_{LF}|$, which we use minus the index of the logical formS^i_{orted} to contribute another part of the S_{borda}. formulated as follows:

$$S_{borda}(T_{LF}) = \sum_{i=1}^{k} \left(|P_{LF}| - \text{rank}(S^i_{orted})\right) \quad (5)$$

Finally, we sort according to the S_{borda} to obtain the final list of candidate logical forms LF_{borda}.

4.4 Knowledge base pattern matching

Since the generated logical form cannot be executed directly, we perform knowledge base schema matching to align entities and relations.

To determine the accurate MID of entities in the question, we first extract their surface names from the candidate logical forms. In KBQA tasks, ELQ (Li et al., 2020) and FACC1 (Gabrilovich et al., 2013) are commonly used entity retrieval methods. We filter all candidate MIDs from FACC1 that exceed a certain threshold. If no candidate MID exceeds the threshold, the first MID from FACC1 is selected as the entity's MID. Additionally, if multiple surface names are detected in the candidate logical forms, all permutations of their combinations are considered.

For relations, if a valid answer can be obtained after execution, we use it, otherwise, we first obtain all relations of all entities in the candidate logical form within 2 hops in the knowledge base, and then use SimCSE (Gao et al., 2021) to obtain top-k most likely matching relation entries for each relation entry in the logical form. Moreover, if more than one relation term of logical form is detected, all permutations of their combination are considered.

5 Experiments

5.1 Dataset

We evaluate RankKBQA on two public standard KBQA datasets as follows:

WebQuestionsSP(WebQSP) (Yih et al., 2016) is a widely recognized KBQA dataset, containing 4,737 natural language questions. The main objective of this dataset is to evaluate the generalization capability in an i.i.d. setting, as the training and testing data share common entities and relations.

ComplexWebQuestions(CWQ) (Talmor and Berant, 2018) extends WebQSP by incorporating four types of complex questions: conjunction (Conj), composition (Compo), comparative (Compa) and superlative (Super). This dataset is used similarly to evaluate generalization ability in an i.i.d. setting.

5.2 Baselines

To comprehensively evaluate our approach, we select a set of state-of-the-art (SOTA) baseline models comprising three categories: LLM-based agent method, fine-tuning method, and few-shot ICL method.

LLM-based agent method include KD-CoT (Wang et al., 2023), Interactive-KBQA (Xiong et al., 2024) and Queryagent (Huang et al., 2024).

Fine-tuned method include RnG-KBQA (Ye et al., 2022), ArcaneQA (Gu and Su, 2022), DECAF(Yu et al., 2023), Uni-Parser (Liu et al., 2022), FC-KBQA (Zhang et al., 2023), HGNet (Chen et al., 2023), ChatKBQA (Luo et al., 2024a) and ROG (Luo et al., 2024b).

Few shot ICL method include KB-BINDER (Li et al., 2023), KB-Coder (Nie et al., 2024), TOG (Sun et al., 2024) and FlexKBQA (Li et al., 2024).

Table 1: Performance Comparison of RankKBQA and Different Baselines on the Two KBQA Datasets

Type	Methods	WebQSP		CWQ	
		F1↑	Hits@1↑	F1↑	Hits@1↑
LLM-based Agent	KD-CoT	52.5	68.8	-	55.7
	Interactive-KBQA	71.2	-	49.1	-
	QueryAgent	69.0	-	-	-
Few-shot ICL	KB-BINDER	74.4	-	-	-
	KB-Coder	75.6	-	-	-
	FlexKBQA	60.6	-	-	-
	ToG	-	75.8	-	58.9
	Ours w/Llama-3-8b	73.4	76.4	59.7	64.4
	Ours w/ChatGPT	**78.5**	**82.0**	**64.3**	**69.8**
Fine-tuning	RnG-KBQA	75.6	-	-	-
	ArcaneQA	75.6	-	-	-
	DecAF	78.8	82.1	-	70.4
	Uni-Parser	75.8	-	-	-
	HGNet	76.6	76.9	68.5	68.9
	FC-KBQA	76.9	-	56.5	-
	ChatKBQA w/Llama-3-8b	78.9	82.7	75.5	79.7
	ROG	70.8	**85.7**	56.2	62.2
	Ours w/Llama-3-8b	**79.6**	83.5	**75.9**	**80.2**

5.3 Evaluation Metrics

Consistent with previous work (Chen et al., 2023; Luo et al., 2024a,b), we use F1 and Hits@1 as personality measures on WebQSP and CWQ. In addition, in the ablation study, we also report the average time cost (ATC) on each dataset.

5.4 Implementation details

For the generation phase, in the fine-tuning method, we chose Llama-3-8B[2], and the beam size was set to 15 for WebQSP, while it was set to 8 for CWQ. In this environment, we replicated ChatKBQA. Among the few-shot ICL methods, we chose Llama-3-8B and ChatGPT[3], and we set the shot to 100 for WebQSP and 40 for CWQ. In the transcription stage, we adopt the T5 family of models, and for WebQSP, we use T5-base. For CWQ, we use T5-large. For computational semantic similarity computation, we use the bge-reranker-v2-m3[4] model without fine-tuning. In knowledge base pattern matching, we adopt a threshold to control the number of candidate entities, as well as K=15 relations as candidates. See Appendix A.1 for details.

[2]https://huggingface.co/meta-llama/Meta-Llama-3-8B
[3]https://openai.com/api
[4]https://huggingface.co/BAAI/bge-reranker-v2-m3

5.5 Main Result

For the baseline methods, except for ChatKBQA, where we use the Llama-3-8B recapitulation, the other baseline methods directly adopt the results reported in the corresponding original papers. As shown in TABLE 1, since few-shot ICL method and LLM-based agents have natural disadvantages compared to fine-tuning methods. Therefore, in our comparison process, RankKBQA based on the few-shot ICL method is not compared with the baseline based on the fine-tuning method. In addition, after analyzing the results, we conduct A specific case analysis in Appendix A.2.

Few-shot ICL method. With the few-shot ICL method, RankKBQA achieves an F1 of 78.5 using ChatGPT on WebQSP, surpassing the state of the art 2.9 (KB-Coder), where KB-Coder employs the same base model ChatGPT as ours. Even outperforming KB-BINDER using Codex and Interactive-KBQA and QueryAgent using GPT-4. For Hits@1, RankKBQA (82.0) outperforms ToG using ChatGPT 6.2. Among them, ToG uses oracle entity annotation, which is obviously better than the FACC1 preference retrieval adopted by RankKBQA. Excitingly, on the CWQ dataset, RankKBQA beats all baselines, including LLM-based Agent and few-

Table 2: Comparison of F1 and ATC in RankKBQA with and without RTAM.

Methods	WebQSP		CWQ	
	F1↑	ATC↓	F1↑	ATC↓
Few-shot ICL Method				
RankKBQA(LLama)	**73.4**	**40.1**	**59.7**	**26.1**
w/o RTAM	73.3	41.9	59.6	45.1
RankKBQA(ChatGPT)	78.3	**10.6**	**64.4**	**14.7**
w/o RTAM	**78.5**	12.9	64.3	19.4
Fine-tuning Method				
RankKBQA(LLama)	**79.6**	**11.4**	**75.9**	**18.1**
w/o RTAM	78.9	11.5	75.5	20.3

shot ICL for F1 and Hit@1, achieving 64.3 for F1 and 69.8 for Hit@1. It surpasses the state-of-the-art (Interactive-KBQA) 15.2 and (ToG) 10.9, respectively. Moreover, RankKBQA with Llama-3-8b achieves the same impressive performance, with an F1 score (73.4) only 2.2 lower than that of the state of the art (KB-Coder) on WebQSP, and outperforms ChatGPT's ToG by 0.6 on Hit@1. Moreover, on CWQ, the new SOTA is also implemented, which is only inferior to RankKBQA, which uses ChatGPT. This also suggests that a more powerful pedestal model will bring us more substantial gains.

Fine-tuning method. In addition, RankKBQA achieves a new F1-score performance (79.6) on WebQSP and also obtains a competitive performance (83.5) on Hit@1, just 2.2 below the state of the art (RoG). However, on the CWQ dataset, RankKBQA achieves a brand new performance (75.9, 80.2) for F1 and Hit@1.

5.6 Ablation Study

We conduct an ablation study to investigate the impact of RTAM on F1 score and average time cost (ATC), tested on few-shot ICL and fine-tuning methods, respectively. Llama and ChatGPT are selected as the basic models in the experiment. The performance improvement of RTAM based on the first validity principle is shown by the gap between the first hit rate and the overall hit rate. The results show that the first hit rate, the overall hit rate, and their difference are significantly improved as the model's ability increases (fig 3a). In addition, Table 2 contrasts the enabled/disabled cases of RTAM and reports the F1 score and ATC score achieved in each case.

In the few-shot ICL method, RankKBQA using Llama as the base model, whether it is WebQSP or CWQ, our method has a certain improvement in F1, and for ATC, it is increased by 1.8s on WebQSP. ATC achieved an astonishing improvement of 19 seconds on CWQ, representing a remarkable speed increase of up to 42.1%. For RankKBQA using ChatGPT, although the performance decreases by 0.2 on WebQSP, it improves by 2.3 seconds in ATC. Also it is improved by 4.7s on CWQ. In addition, in the fine-tuning method, RankKBQA using RTAM has a certain degree of improvement in F1 and ATC.

In addition, in order to verify the robustness of RTAM, we replaced different transcription models and semantic similarity methods. Due to the long inference time to complete all test problems, we randomly selected 200 and 500 from WebQSP and CWQ, respectively. In addition, we find in the experiments that in the Borda Count of two clues, it may occur that two candidates received the same score, resulting in no change in order, and we guess that the preliminary generated candidate list is too weighted; therefore, we consider a kind of weighted Borda Count. In the following, we will expand the analysis separately.

Transcription models may have some influence on the final candidate list order. We chose Flan-T5-Large, which is more powerful than T5-base and T5-Large, for question transcription. As shown in Figure 3b, the more powerful transcription model does achieve better performance. We analyze that this is due to the fact that the powerful transcription model is able to better capture the relation and entity patterns in the logical form, thus transcribing questions that are more consistent with the logical form. However, the better performance increases the ATC, we believe analytically that it is caused during the pattern matching process of the KB. The performance improvement is not a result of preliminary logical form execution, where RankKBQA only improves the ranking of candidates that are closer to the golden fact.

Semantic Similarity Methods may be another factor affecting the order of the candidate list. Besides bge-reranker-v2-m3, we choose BM25 and SimCSE dense retrieval, which are more general. As shown in Figure 3c, different semantic similarity methods can further improve the model performance and ensure good ATC. On WebQSP, BM25 and SimCSE achieve the same F1 values and outperform bge-reranker-v2-m3, and in ATC, all three are approximately the same. On CWQ, BM25

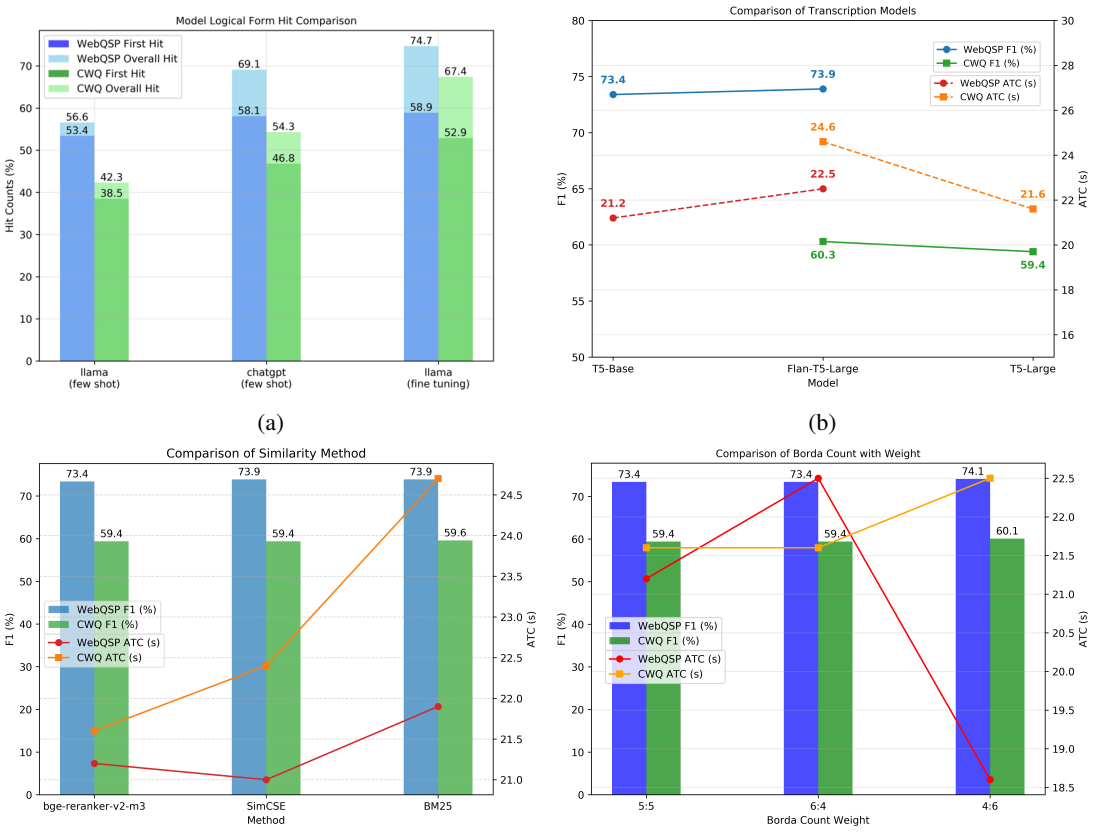

Figure 3: (a) Model Logical Form Hit Comparison. (b) Comparison of Transcription Models. (c) Comparison of Semantic Similarity Methods. (d) Comparison of Borda Count with Weight.

achieves unexpectedly high F1 scores, but with some increase in ATC. We believe analytically that it is caused by pattern matching in the knowledge base. The performance improvement is not a result of preliminary logical form execution, RankKBQA only improves the ranking of candidates that are closer to the golden fact.

Borda Count with weight is introduced to address the issue that may arise in the original Borda Count method, where two candidates might receive identical scores, resulting in no change in their ranking. To verify this, we assign different weight ratios to the preliminary candidate list and the sorted candidate list—specifically 5:5, 6:4, and 4:6. As illustrated in Figure 3d, the model yields nearly identical F1 and ATC performance under the 5:5 and 6:4 settings, which aligns with our expectations. However, when using a 4:6 weight ratio, the model shows improved performance on WebQSP, accompanied by a reduction in ATC. This supports our hypothesis that assigning a greater weight to the sorted candidate list improves the ranking of golden facts on WebQSP. In contrast, on CWQ, due to the relatively low accuracy of preliminary candidates and the dependency on knowledge base pattern matching, RankKBQA promotes the ranking of candidates closer to the golden truth, which will always lead to the rise of ATC.

6 Conclusion

In conclusion, this paper introduces RankKBQA, a novel framework that can further optimize the capabilities of KBQA systems based on LLMs through a transcribe-then-rank approach. Extensive experiments on different datasets demonstrate the effectiveness of RankKBQA, which outperforms all baseline methods both under the few-shot ICL method and under the fine-tuning method. And, in terms of the average time cost, our method can reduce the average time cost considerably compared to other baseline methods. In the era of post-LLMs, this is sufficient to meet the user requirements of KBQA systems, both in terms of response speed and performance.

Limitations

Although RankKBQA has made significant progress in improving the accuracy and inference efficiency of KBQA systems, there are still some limitations. On the one hand, the performance of RankKBQA depends on the generation quality of candidate logical forms. If the generation phase fails to cover the golden logical form, it is difficult to improve the accuracy of the final answer, even if the subsequent bidirectional reranking strategy is excellent. On the other hand, RankKBQA relies on the transcription process from logical form to natural language, and its effect is limited by the generalization ability of the pre-trained model. Therefore, future research should consider further optimizing the generation coverage of logical forms and improving the robustness of transcription quality, so as to further promote the landing and expansion of RankKBQA in practical applications.

References

OpenAI Josh Achiam, Steven Adler, Sandhini Agarwal, Lama Ahmad, Ilge Akkaya, Florencia Leoni Aleman, Diogo Almeida, Janko Altenschmidt, Sam Altman, Shyamal Anadkat, Red Avila, Igor Babuschkin, Suchir Balaji, Valerie Balcom, Paul Baltescu, Haim ing Bao, Mo Bavarian, Jeff Belgum, Irwan Bello, and 260 others. 2023. Gpt-4 technical report.

Kurt Bollacker, Colin Evans, Praveen Paritosh, Tim Sturge, and Jamie Taylor. 2008. Freebase: a collaboratively created graph database for structuring human knowledge. In *Proceedings of the 2008 ACM SIGMOD International Conference on Management of Data*, SIGMOD '08, page 1247–1250, New York, NY, USA. Association for Computing Machinery.

Shuang Chen, Qian Liu, Zhiwei Yu, Chin-Yew Lin, Jian-Guang Lou, and Feng Jiang. 2021. ReTraCk: A flexible and efficient framework 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: System Demonstrations*, pages 325–336, Online. Association for Computational Linguistics.

Wenhu Chen, Xueguang Ma, Xinyi Wang, and William W. Cohen. 2022. Program of thoughts prompting: Disentangling computation from reasoning for numerical reasoning tasks. *Trans. Mach. Learn. Res.*, 2023.

Yongrui Chen, Huiying Li, Guilin Qi, Tianxing Wu, and Tenggou Wang. 2023. Outlining and filling: Hierarchical query graph generation for answering complex questions over knowledge graphs. *IEEE Trans. on Knowl. and Data Eng.*, 35(8):8343–8357.

Zhoujun Cheng, Tianbao Xie, Peng Shi, Chengzu Li, Rahul Nadkarni, Yushi Hu, Caiming Xiong, Dragomir Radev, Mari Ostendorf, Luke Zettlemoyer, Noah A. Smith, and Tao Yu. 2023. Binding language models in symbolic languages. In *The Eleventh International Conference on Learning Representations*.

Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Amy Yang, Angela Fan, and 1 others. 2024. The llama 3 herd of models. *arXiv preprint arXiv:2407.21783*.

Evgeniy Gabrilovich, Michael Ringgaard, and Amarnag Subramanya. 2013. Facc1: Freebase annotation of clueweb corpora, version 1 (release date 2013-06-26, format version 1, correction level 0).

Luyu Gao, Aman Madaan, Shuyan Zhou, Uri Alon, Pengfei Liu, Yiming Yang, Jamie Callan, and Graham Neubig. 2023. Pal: program-aided language models. In *Proceedings of the 40th International Conference on Machine Learning*, ICML'23. JMLR.org.

Tianyu Gao, Xingcheng Yao, and Danqi Chen. 2021. SimCSE: Simple contrastive learning of sentence embeddings. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 6894–6910, Online and Punta Cana, Dominican Republic. Association for Computational Linguistics.

Yu Gu, Sue Kase, Michelle Vanni, Brian Sadler, Percy Liang, Xifeng Yan, and Yu Su. 2021. Beyond i.i.d.: Three levels of generalization for question answering on knowledge bases. In *Proceedings of the Web Conference 2021*, WWW '21, page 3477–3488, New York, NY, USA. Association for Computing Machinery.

Yu Gu and Yu Su. 2022. ArcaneQA: Dynamic program induction and contextualized encoding for knowledge base question answering. In *Proceedings of the 29th International Conference on Computational Linguistics*, pages 1718–1731, Gyeongju, Republic of Korea. International Committee on Computational Linguistics.

Edward J Hu, yelong shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. 2022. LoRA: Low-rank adaptation of large language models. In *International Conference on Learning Representations*.

Xiang Huang, Sitao Cheng, Shanshan Huang, Jiayu Shen, Yong Xu, Chaoyun Zhang, and Yuzhong Qu. 2024. QueryAgent: A reliable and efficient reasoning framework with environmental feedback based self-correction. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 5014–5035, Bangkok, Thailand. Association for Computational Linguistics.

Yunshi Lan, Gaole He, Jinhao Jiang, Jing Jiang, Wayne Xin Zhao, and Ji-Rong Wen. 2021. A survey on complex knowledge base question answering: Methods, challenges and solutions. In *Proceedings of the Thirtieth International Joint Conference on Artificial Intelligence, IJCAI-21*, pages 4483–4491. International Joint Conferences on Artificial Intelligence Organization. Survey Track.

Belinda Z. Li, Sewon Min, Srinivasan Iyer, Yashar Mehdad, and Wen-tau Yih. 2020. Efficient one-pass end-to-end entity linking for questions. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 6433–6441, Online. Association for Computational Linguistics.

Tianle Li, Xueguang Ma, Alex Zhuang, Yu Gu, Yu Su, and Wenhu Chen. 2023. Few-shot in-context learning on knowledge base question answering. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 6966–6980, Toronto, Canada. Association for Computational Linguistics.

Zhenyu Li, Sunqi Fan, Yu Gu, Xiuxing Li, Zhichao Duan, Bowen Dong, Ning Liu, and Jianyong Wang. 2024. Flexkbqa: a flexible llm-powered framework for few-shot knowledge base question answering. In *Proceedings of the Thirty-Eighth AAAI Conference on Artificial Intelligence and Thirty-Sixth Conference on Innovative Applications of Artificial Intelligence and Fourteenth Symposium on Educational Advances in Artificial Intelligence*, AAAI'24/IAAI'24/EAAI'24. AAAI Press.

Percy Liang. 2013. Lambda dependency-based compositional semantics. *arXiv preprint arXiv:1309.4408*.

Ye Liu, Semih Yavuz, Rui Meng, Dragomir Radev, Caiming Xiong, and Yingbo Zhou. 2022. Uni-parser: Unified semantic parser for question answering on knowledge base and database. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, pages 8858–8869, Abu Dhabi, United Arab Emirates. Association for Computational Linguistics.

Haoran Luo, Haihong E, Zichen Tang, Shiyao Peng, Yikai Guo, Wentai Zhang, Chenghao Ma, Guanting Dong, Meina Song, Wei Lin, Yifan Zhu, and Anh Tuan Luu. 2024a. ChatKBQA: A generate-then-retrieve framework for knowledge base question answering with fine-tuned large language models. In *Findings of the Association for Computational Linguistics: ACL 2024*, pages 2039–2056, Bangkok, Thailand. Association for Computational Linguistics.

LINHAO Luo, Yuan-Fang Li, Reza Haf, and Shirui Pan. 2024b. Reasoning on graphs: Faithful and interpretable large language model reasoning. In *The Twelfth International Conference on Learning Representations*.

Zhijie Nie, Richong Zhang, Zhongyuan Wang, and Xudong Liu. 2024. Code-style in-context learning for knowledge-based question answering. In *Proceedings of the Thirty-Eighth AAAI Conference on Artificial Intelligence and Thirty-Sixth Conference on Innovative Applications of Artificial Intelligence and Fourteenth Symposium on Educational Advances in Artificial Intelligence*, AAAI'24/IAAI'24/EAAI'24. AAAI Press.

Jiashuo Sun, Chengjin Xu, Lumingyuan Tang, Saizhuo Wang, Chen Lin, Yeyun Gong, Lionel Ni, Heung-Yeung Shum, and Jian Guo. 2024. Think-on-graph: Deep and responsible reasoning of large language model on knowledge graph. In *The Twelfth International Conference on Learning Representations*.

Alon Talmor and Jonathan Berant. 2018. The web as a knowledge-base for answering complex questions. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 641–651, New Orleans, Louisiana. Association for Computational Linguistics.

Denny Vrandečić and Markus Krötzsch. 2014. Wikidata: a free collaborative knowledgebase. *Commun. ACM*, 57(10):78–85.

Keheng Wang, Feiyu Duan, Sirui Wang, Peiguang Li, Yunsen Xian, Chuantao Yin, Wenge Rong, and Zhang Xiong. 2023. Knowledge-driven cot: Exploring faithful reasoning in llms for knowledge-intensive question answering. *ArXiv*, abs/2308.13259.

Xuezhi Wang, Jason Wei, Dale Schuurmans, Quoc Le, Ed H. Chi, and Denny Zhou. 2022. Self-consistency improves chain of thought reasoning in language models. *ArXiv*, abs/2203.11171.

Guanming Xiong, Junwei Bao, and Wen Zhao. 2024. Interactive-KBQA: Multi-turn interactions for knowledge base question answering with large language models. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 10561–10582, Bangkok, Thailand. Association for Computational Linguistics.

Xi Ye, Semih Yavuz, Kazuma Hashimoto, Yingbo Zhou, and Caiming Xiong. 2022. RNG-KBQA: Generation augmented iterative ranking for knowledge base question answering. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 6032–6043, Dublin, Ireland. Association for Computational Linguistics.

Wen-tau Yih, Ming-Wei Chang, Xiaodong He, and Jianfeng Gao. 2015. Semantic parsing via staged query graph generation: Question answering with knowledge base. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 1321–1331, Beijing, China. Association for Computational Linguistics.

Wen-tau Yih, Matthew Richardson, Chris Meek, Ming-Wei Chang, and Jina Suh. 2016. The value of semantic parse labeling for knowledge base question answering. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 201–206, Berlin, Germany. Association for Computational Linguistics.

Donghan Yu, Sheng Zhang, Patrick Ng, Henghui Zhu, Alexander Hanbo Li, Jun Wang, Yiqun Hu, William Yang Wang, Zhiguo Wang, and Bing Xiang. 2023. DecAF: Joint decoding of answers and logical forms for question answering over knowledge bases. In *The Eleventh International Conference on Learning Representations*.

Lingxi Zhang, Jing Zhang, Yanling Wang, Shulin Cao, Xinmei Huang, Cuiping Li, Hong Chen, and Juanzi Li. 2023. FC-KBQA: A fine-to-coarse composition framework for knowledge base question answering. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1002–1017, Toronto, Canada. Association for Computational Linguistics.

A Appendix

A.1 Implementation details

For the generation phase, in the fine-tuning method, we choose Llama-3-8b as our base LLM, where batch size is 4, learning rate is 5e-5, other parameters are kept at default values, and training is performed for 10 epochs. In addition, for WebQSP, the beam size is set to 15 while for CWQ, the beam size is set to 8. Meanwhile, based on this environment, we replicate ChatKBQA. In the few-shot ICL method, we still choose Llama-3-8B as the base LLM, where the temperature is set to 0.7, top-k is set to 4, top-p is set to 0.7, and max_token is set to 8192. In addition, num_return_sequences is set to 3 (the maximum is limited due to laboratory resources). For WebQSP, we set the shot to 100, and for CWQ, we set the shot to 40. In addition, in order to verify that a larger model can bring more substantial benefits, based on this experimental environment, we replace Llama-3-8b and use ChatGPT (gpt-3.5-turbo) under API calls to conduct experiments. During the transcription phase, we employ the T5 family of models as the PLM for transcription. For WebQSP, due to its relatively smaller dataset size, we use a base version of the T5 family, i.e., T5-base. For CWQ, which has more complex and diverse data, we utilize a larger version of the T5 family, i.e., T5-large, with a learning rate of 3e-5, a batch size of 8, and the same beam search decoding strategy, setting the beam size to 10. The model was trained for 20 epochs. Additionally, for calculating semantic similarity scores, we used the bge-reranker-v2-m3 model without fine-tuning. In the process of knowledge base schema matching, we adopt a threshold to control the number of entities to be matched. For relational schema matching, we use the SimCSE instance unsup-simcse-roberta-large to obtain a dense representation. We allow K=15 relations to be matched for each relation entry.

A.2 Case Study

By analyzing the questions where RankKBQA did not get the correct results, we present several representative correct and incorrect cases. We selected two valid questions and two failed questions from the WebQSP and CWQ test sets, respectively. For the cases where RankKBQA works, we label the correct answer as green, and for the cases where RankKBQA fails, we label the correct answer as rose red.

WebQSP
Effective case study
Question: How deep is Lake Merritt Oakland?

Preliminary generation:

- `(JOIN(R[location,location,depth]) [Lake Merritt]);`
- `(JOIN(R[location,location,contains]) [Lake Merritt]);`
- `(JOIN(R[geography, body of water,depth]) [Lake Merritt]);` ✓

RankKBQA:

- `(JOIN(R[geography,body of water,depth]) [Lake Merritt]);` ✓
- `(JOIN(R[location,location,contains]) [Lake Merritt]);`
- `(JOIN(R[location,location,depth]) [Lake Merritt]);`

WebQSP
Failed case study
Question: What other books did Charles Dickens write?

Preliminary generation:

- `(JOIN(R[book,author,works written]) [Charles Dickens]);`
- `(JOIN(R[book,author,book editions published]) [Charles Dickens]);` ✓
- `(JOIN(R[book,author,written works]) [Charles Dickens]);`

RankKBQA:

- `(JOIN(R[book,author,works written]) [Charles Dickens]);`
- `(JOIN(R[book,author,written works]) [Charles Dickens]);`
- `(JOIN(R[book,author,book editions published]) [Charles Dickens]);` ✓

CWQ
Effective case study
Question: Where is the state with the capital of Frankfort located?

Preliminary Generation:

- `(JOIN(R[base,aareas,schema,administrative area,administrative parent])(JOIN [location,us state,capital][Frankfort]));`
- `(JOIN(R[location,location,containedby])(JOIN [location,us state,capital][Frankfort]))` ✓
- `(JOIN(R[location,location,administrative parent])(JOIN[location,us state,capital] [Frankfort]))`

RankKBQA:

- `(JOIN(R[location,location,containedby])(JOIN [location,us state,capital][Frankfort]))` ✓
- `(JOIN(R[base,aareas,schema,administrative area,administrative parent])(JOIN [location,us state,capital][Frankfort]));`
- `(JOIN(R[location,location,administrative parent])(JOIN[location,us state,capital] [Frankfort]))`

CWQ

Failed case study

Question: What states does the Colorado River run through in the Mountain Time Zone?

Preliminary Generation:

- `(AND(JOIN[location,location,time zones][Mountain Time Zone])(JOIN(R[location,location,partially containedby])[Colorado River]))`
- `(AND(JOIN[location,location,time zones][Mountain Time Zone])(AND(JOIN[base,biblioness,bibs location,loc type]"State")(JOIN(R[location,location,partially containedby])[Colorado River]))) ✓`
- `(AND(JOIN[location,location,time zones][Mountain Time Zone](AND(JOIN[base,biblioness,bibs location,loc type]"State")(JOIN(R[location,location,partially contains])"[Colorado River])))`

RankKBQA:

- `(AND(JOIN[location,location,time zones][Mountain Time Zone](AND(JOIN[base,biblioness,bibs location,loc type]"State")(JOIN(R[location,location,partially contains])"[Colorado River])))`
- `(AND(JOIN[location,location,time zones][Mountain Time Zone])(AND(JOIN[base,biblioness,bibs location,loc type]"State")(JOIN(R[location,location,partially containedby])[Colorado River]))) ✓`
- `(AND(JOIN[location,location,time zones][Mountain Time Zone])(JOIN(R[location,location,partially containedby])[Colorado River]))`

For the WebQSP dataset, in the valid case, the difference between the three candidate logical forms lies in the relation pattern. RankKBQA can improve the position of the correct relation pattern by executing RTAM, thereby reducing the execution time of the first two and even the occurrence of wrong answers. In the error case, because the pattern relationship in the candidate list is too similar, the transcribed question and the original question also have high similarity, so the final ranking result does not adjust the position of the gold fact to the first place in the list. For the CWQ dataset, in the effective case, the relation patterns with relatively large differences in similarity also appear, so RankKBQA can adjust the position of gold facts well by RTAM. Similarly, RankKBQA may not work well in the error case where the relation pattern is too similar. We analyze that more complex questions and their logical forms contain more information, which also provides RankKBQA with more angles to adjust the position of candidate logical forms, which will give greater play to the performance of RankKBQA, which is also more in line with what may really happen in real scenarios.