Decompose then Discriminate: LLMs-Grounded In-Context Learning for Few-Shot Knowledge Base Question Answering

Shengze Shi[1,2,3], Tao Ren[1,2,3] (✉), Xu Lan[1], and Jun Hu[1,2,3] (✉)

[1] State Key Laboratory of Intelligent Game, Beijing, China
[2] Institute of Software Chinese Academy of Sciences, Beijing, China
[3] University of Chinese Academy of Sciences, Beijing, China
{shishengze2023,rentao22}@iscas.ac.cn, lanxv1991@163.com, hujun@iscas.ac.cn

Abstract. Knowledge base question answering (KBQA) is an open and challenging task, which aims to provide knowledge base (KB) answers according to natural language questions. Most existing supervised KBQA methods require substantial labeled data along with a customized training framework to achieve satisfactory performance, due to the complexity of KBs and diversity of questions. Additionally, supervised KBQA methods are usually KB-specific, thus poor at generalization. Owing to the impressive in-context learning capability of large language models (LLMs), as well as their transformation ability from natural language to structured code, this paper adopts LLMs to achieve few-shot logical expression (LE) based KBQA. Considering the complexity of directly-generating LEs, we propose to decompose the direct LE-generation by LLMs into logical skeleton generation and expression component discrimination. Furthermore, we transform the generative task of LLMs into the candidate selection of skeleton and component by leveraging LLMs' discriminative instead of generative ability. To alleviate LLMs' hallucination, we further filter irrelevant candidates to the given question to enhance the qualities of in-context learning samples. We conduct extensive experiments on popular KBQA datasets and achieve substantial few-shot performance improvements over the state-of-the-art LLMs-grounded method.

Keywords: Knowledge Base Question Answering · Large Language Model · In-Context Learning · Few Shot.

1 Introduction

Knowledge base question answering (KBQA) [2], enabling non-expert users to query knowledge bases using natural language, has been a key research area in the natural language processing community [21]. However, traditional KBQA models face significant challenges: they require extensive labeled data, expensive

This work was supported in part by National Natural Science Foundation of China under Grant No. 62406330

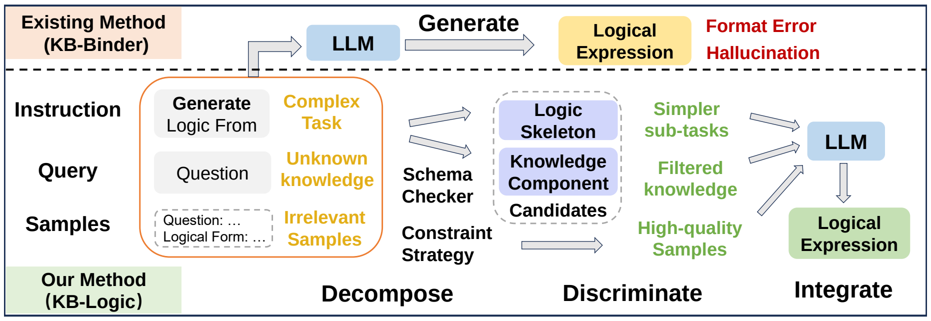


Fig. 1: Comparison of LLM-based KBQA Methods.

training frameworks [5], and often lack generalizability across different KBs [16]. Large language models (LLMs) have demonstrated impressive generalization capabilities through few-shot in-context learning [3], particularly in transforming natural language into structured code [19].

However, directly generating complete LEs is challenging for LLMs, as these expressions contain both logical structures and knowledge components. Research shows that decomposing complex tasks improves LLM performance [12]. Therefore, we propose splitting the LE generation task into two sub-tasks: logical skeleton generation and expression component discrimination [24].

Logical skeleton generation poses challenges for LLMs due to limited pretraining exposure, resulting in format errors [11] and weak logical reasoning capabilities [15]. To address this, this paper converts logical skeleton generation into a candidate selection task, leveraging LLMs' discriminative rather than generative abilities to enhance logical reasoning and LE executability. Additionally, LLMs' hallucination issues [8] necessitate careful knowledge component extraction from KBs for expression component discrimination. The extracted components must be both limited in quantity and highly relevant to avoid confusion and misleading results [1].

Taking all these considerations into account, We propose KB-Logic, a decomposed and discriminative LE-based KBQA framework (the bottom of Figure 1), that divides the LE generation task into three sub-tasks: logical skeleton parsing (LSP), knowledge component retrieval (KCR), and logical expression discrimination (LED). For LSP, we transform it into a candidate selection task by creating a logical skeleton pool from sample questions and retrieving candidates using LLM-driven logical type judgment. For KCR, we employ schema-checkers to filter inconsistent component candidates and design a constraint strategy to limit samples to those closely related to the question. For LED, we interpolate selected knowledge components into logical skeleton candidates to construct final LEs, using fine-grained components as augmented information in prompts. Experiments on GrailQA, WebQSP, and GraphQA datasets demonstrate performance improvements over state-of-the-art LLMs-grounded KBQA methods.

2 Method

We propose a decomposed and discriminative LE-based KBQA framework, KB-Logic, achieving promising generalization via LLMs' in-context learning ability. As shown in Figure 2, given a question Q, we decompose the complex task of directly constructing executable LEs into three sub-tasks: LSP, KCR and LED. For LSP and KCR, we transform the generative tasks into the selection tasks of logical skeletons (LKs) and knowledge components (KCs). For LED, we interpolate the KCs selected by KCR into the LKs selected by LSP to construct the final LEs, and leverage LLMs to discriminate the optimal one for KBQA.

2.1 Logical Skeleton Parsing

The goal of LSP is to analyze the logical structure of the target question Q and identify the most relevant LK candidates C_{ske} for logical expression construction. LSP encompasses logical type parsing (counting, superlativeness, comparison) and reasoning structure parsing (single/multiple hops).
Logical Skeleton Extraction. A LE combines knowledge factors (entities, classes, relationships from KB) and logic factors (logical types, reasoning structures), as shown in Figure 3. The logical skeleton is extracted by replacing knowledge components with placeholders (`<class>`, `<rel>`, `<entity>`, `<literal>`) while retaining logical operators (AND, JOIN) and type indicators (COUNT, ARGMAX).
Logical Skeleton Pool Construction. LKs are categorized into four major types: *general, counting, superlativeness*, and *comparison*, and further subdivided based on reasoning structures to form a comprehensive logical skeleton pool.
Logical Type Judgment. LSP encompasses logical type parsing and reasoning structure parsing. While reasoning structure analysis requires KB-specific knowledge, logical type identification can be performed using question keywords alone. Using LLMs, we extract type-specific keywords and classify questions into four major logical types, which helps narrow down and improve LK retrieval accuracy.
Logical Candidate Retrieval. For logical candidate retrieval, we opt for a retrieval-based approach over generative methods due to its superior accuracy and controllability. We implement a hybrid retrieval system combining BM25 and dense retrieval models like Contriever to find N similar questions of the same logical type. The most frequently occurring top-k logical skeletons from these examples form the preliminary candidate set C_{ske}.

2.2 Knowledge Component Retrieval

Apart from parsing C_{ske}, it is necessary to acquire KCs from the KB that align with the semantics of the question, such as entities, relationships, and classes, to fill the placeholders in LKs and thereby construct a complete LE.
Knowledge Component Extraction. Entity candidates are obtained via entity linking, relationships through hybrid retrieval from the KB, and class candidates based on KB schema definitions.

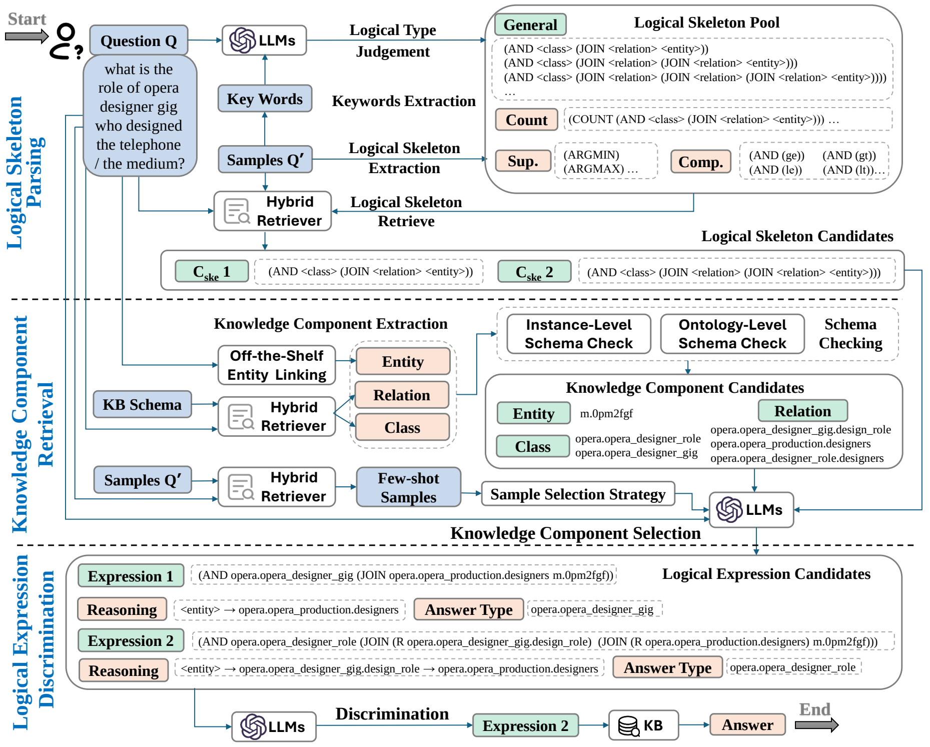

Start
Question Q
what is the role of opera designer gig who designed the telephone / the medium?
LLMs
Logical Type Judgement
Key Words
Keywords Extraction
Samples Q′
Logical Skeleton Extraction
Hybrid Retriever
Logical Skeleton Retrieve
Logical Skeleton Pool
General
(AND <class> (JOIN <relation> <entity>))
(AND <class> (JOIN <relation> (JOIN <relation> <entity>)))
(AND <class> (JOIN <relation> (JOIN <relation> (JOIN <relation> <entity>))))
Count
(COUNT (AND <class> (JOIN <relation> <entity>))) …
Sup.
(ARGMIN) (ARGMAX) …
Comp.
(AND (ge)) (AND (le)) (AND (gt)) (AND (lt))…
Logical Skeleton Candidates
C_ske 1
(AND <class> (JOIN <relation> <entity>))
C_ske 2
(AND <class> (JOIN <relation> (JOIN <relation> <entity>)))
Logical Skeleton Parsing
Knowledge Component Extraction
Off-the-Shelf Entity Linking
KB Schema
Hybrid Retriever
Entity
Relation
Class
Instance-Level Schema Check
Ontology-Level Schema Check
Schema Checking
Knowledge Component Candidates
Entity
m.0pm2fgf
Relation
opera.opera_designer_gig.design_role
opera.opera_production.designers
opera.opera_designer_role.designers
Class
opera.opera_designer_role
opera.opera_designer_gig
Samples Q′
Hybrid Retriever
Few-shot Samples
Sample Selection Strategy
LLMs
Knowledge Component Selection
Knowledge Component Retrieval
Logical Expression Candidates
Expression 1
(AND opera.opera_designer_gig (JOIN opera.opera_production.designers m.0pm2fgf))
Reasoning
<entity> → opera.opera_production.designers
Answer Type
opera.opera_designer_gig
Expression 2
(AND opera.opera_designer_role (JOIN (R opera.opera_designer_gig.design_role) (JOIN (R opera.opera_production.designers) m.0pm2fgf)))
Reasoning
<entity> → opera.opera_designer_gig.design_role → opera.opera_production.designers
Answer Type
opera.opera_designer_role
LLMs
Discrimination
Expression 2
KB
Answer
End
Logical Expression Discrimination

Fig. 2: KB-Logic framework: given a quesiton, we decompose the task into logical skeleton parsing and knowledge component retrieval, and integrate these results to construct a list of logical expression candidates, from which we discriminate the optimal logical expression for KBQA.

Knowledge Components Checking. Schema validation occurs at both instance and ontology levels, as shown in Figure 3. For one-hop relationships, instance-level checking queries direct relationships from entities. Multi-hop relationships undergo ontology-level checking based on KB schema definitions to verify connectivity between hops. The results are merged to form final relationship candidates.
Knowledge Component Selection. Large Language Models perform selection through in-context learning, where $D_{\text{comp}} = f_{\text{LLM}}(I; S; O)$, with f_{LLM} representing a specific LLM. The process uses instructions I, samples S, and objectives O, with two sample selection strategies and constraint approaches detailed in Appendix.

2.3 Logical Expression Discrimination

During the above LSP phase, we have already made a preliminary judgment on the logical type of the LEs corresponding to Q. At this point, our focus shifts

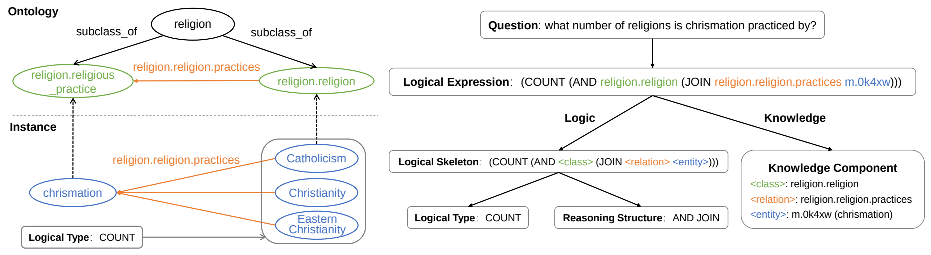

Ontology
religion
subclass_of
subclass_of
religion.religious_practice
religion.religion.practices
religion.religion
Instance
religion.religion.practices
Catholicism
chrismation
Christianity
Eastern Christianity
Logical Type: COUNT
Question: what number of religions is chrismation practiced by?
Logical Expression: (COUNT (AND religion.religion (JOIN religion.religion.practices m.0k4xw)))
Logic
Knowledge
Logical Skeleton: (COUNT (AND <class> (JOIN <relation> <entity>)))
Knowledge Component
<class>: religion.religion
<relation>: religion.religion.practices
<entity>: m.0k4xw (chrismation)
Logical Type: COUNT
Reasoning Structure: AND JOIN

Fig. 3: Illustration of a KB subgraph and the decomposition of logical expression.

to the analysis of the reasoning structure, specifically, the determination of the reasoning steps required to answer Q.

For each LK in C_{ske}, placeholders are filled through knowledge component selection, generating a final set of complete-LE candidates, C_{le}. Since each LE in C_{le} combines logical structure with knowledge semantics, LLMs face challenges in directly discriminating them. To address this, we propose a fine-grained, component-based LE discrimination method, where KC information is incorporated into prompts to guide LLMs in analyzing the roles of different KC types in reasoning.

The method extracts class and relationship components from each LE. LLMs evaluate relationship components based on their ability to form a complete reasoning chain that satisfies Q. For class components, LLMs assess their accuracy as the answer type for Q. This approach ensures coherence in reasoning structure and sufficiency in knowledge semantics, improving LLMs' logical analysis and semantic understanding of complex LEs, thus enhancing their ability to discriminate LEs.

3 Experiment

3.1 Experiment Setup

Dataset. The evaluation was conducted on four KBQA datasets: GrailQA [5], which tests I.I.D., compositional, and zero-shot generalization with complex 4-hop relations and diverse aggregation functions; WebQSP [22], focusing on I.I.D. generalization with 2-hop relations and constraints; GraphQA [16], challenging compositional generalization through syntactic complexity; and MetaQA [25], a movie-domain dataset with over 400,000 questions testing multi-hop reasoning capabilities.

Metrics. According to the distinct features of the above datasets, we adopt corresponding metrics, implemented using official scripts, to evaluate the KBQA performance on each dataset. For GrailQA, we use the exact match (EM) to measure the accuracy of constructed LE and compute the F1 Score based on the

Table 1: Results on GrailQA.

Method	I.I.D.		Compositional		Zero-Shot		Overall	
	EM	**F1**	**EM**	**F1**	**EM**	**F1**	**EM**	**F1**
Full Supervised on the Entire Training Set								
RnG-KBQA [20]	86.7	89.0	61.7	68.9	68.8	74.7	69.5	76.9
DecAF [23]	88.7	92.4	71.5	79.8	65.9	77.3	72.5	81.4
TIARA [14]	88.4	91.2	66.4	74.8	73.3	80.7	75.3	81.9
LLM-only Methods (Training-Free)								
ChatGPT	18.6	19.6	16.1	17.0	29.6	31.2	24.0	25.3
Davinci-003	22.3	23.5	20.8	22.0	34.5	36.4	28.5	30.1
GPT-4	23.7	25.0	19.5	20.6	37.1	39.2	30.1	31.7
LLM-based Reasoning Framework (Training-Free)								
StructGPT [9]	66.0	70.4	39.9	44.3	45.2	50.5	49.8	54.6
KB-BINDER [10]	75.8	80.9	48.3	53.6	45.4	50.7	53.2	58.5
KB-Coder [11]	76.9	81.0	52.7	57.8	48.9	54.1	56.3	61.3
Pangu [4]	54.2	62.3	54.1	63.2	51.5	61.4	52.5	61.9
KB-Logic	62.7	65.2	**55.6**	58.0	**61.8**	**64.7**	**60.5**	**63.2**

predicted and gold answers. We report the F1 score for WebQSP and GraphQA, and use Hits@1 for MetaQA following standard evaluation methods.
Baselines. We consider the following three types of baseline methods for performance comparison: (1) Fully-supervised methods, such as RnG-KBQA [20], DecAF [23], and TIARA [14], provide strong performance benchmarks. (2) LLM-only methods, such as ChatGPT, Davinci-003, and GPT-4, offer a reliable baseline for comparison without training. (3) LLM-based reasoning frameworks, such as StructGPT [9], KB-BINDER [10], KB-Coder [11], and Pangu [4], demonstrate competitive performance under training-free setups, all of which use GPT-3.5-Turbo to ensure a fair comparison.

3.2 Main Results

We evaluate KB-Logic's performance across various datasets, demonstrating competitive results compared to other LLM-based reasoning frameworks, particularly excelling in zero-shot scenarios.On GrailQA, as shown in Table 1, KB-Logic shows superior generalization ability in the zero-shot category for both EM and F1 scores, with a slight advantage in compositional queries' EM scores, though there remains room for improvement in I.I.D. performance. For WebQSP, as shown in Table 2, KB-Logic outperforms all LLM-only methods but ranks slightly behind KB-Coder and KB-BINDER in the I.I.D. scenario, consistent with GrailQA observations, while still surpassing Pangu and StructGPT due to better integration of structured LKs and KCs. In GraphQA, ss shown in Table 3, KB-Logic achieves the highest F1 score (43.8) among all evaluated methods, attributable to its ef-

Table 2: WebQSP.

Method	**F1**
Full Supervised	
ArcaneQA [6]	75.6
TIARA [14]	76.7
DecAF [23]	78.7
LLM-only Methods	
ChatGPT	59.3
Davinci-003	63.9
GPT-4	62.3
LLM-based Framework	
StructGPT [9]	63.7
KB-BINDER [10]	71.1
KB-Coder [11]	75.2
Pangu [4]	53.9
KB-Logic	68.6

Table 3: GraphQA.

Method	**F1**
Full Supervised	
SPARQA [18]	21.5
BERT+Ranking [5]	25.0
ArcaneQA [6]	31.8
LLM-only Methods	
ChatGPT	24.5
Davinci-003	25.7
GPT-4	30.8
LLM-based Framework	
StructGPT [9]	31.9
KB-BINDER [10]	32.5
KB-Coder [11]	36.6
Pangu [4]	41.7
KB-Logic	**43.8**

Table 4: MetaQA.

Method	**Hits@1**
Full Supervised	
GraphNet [17]	89.8
Emb [13]	97.2
NSM [7]	98.6
LLM-only Methods	
ChatGPT	43.4
Davinci-003	55.7
GPT-4	70.9
LLM-based Framework	
StructGPT [9]	89.0
KB-BINDER [10]	97.3
KB-Coder [11]	98.0
Pangu [4]	98.3
KB-Logic	**99.6**

fective knowledge base integration with logic reasoning, enhancing compositional generalization for handling syntactic complexity. For MetaQA, as shown in Table 4, KB-Logic achieves the highest Hits@1 of 99.6, outperforming all other methods and demonstrating excellent multi-hop reasoning capability.

3.3 Ablation Studies

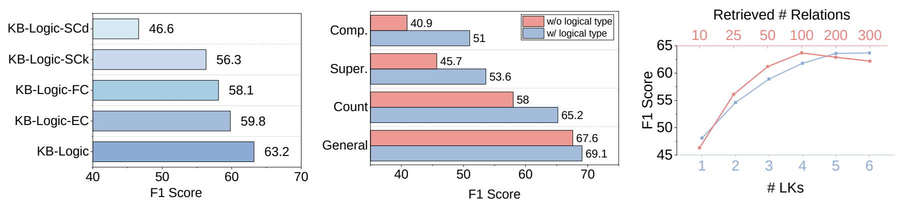
As shown in Figure 4a, we designed several model variants to investigate the impact of various components of KB-Logic on KBQA performance on GrailQA:

• w/o Skeleton Candidate: Use C_{ske} generated by LLM instead of selecting from retrieval-based candidates (abbr. KB-Logic-SCd). This leads to significant performance degradation, showing the importance of discriminating over generating C_{ske}.

• w/o Schema Checks: Use hybrid retrieval results as C_{comp} without schema checking (abbr. KB-Logic-SCk). This results in a slight performance decrease, indicating the necessity of schema checks for accurate relation selection.

• w/o Example Constraints: Use hybrid retrieval results as in-context learning samples without skeleton-based and candidate-based constraints (abbr. KB-Logic-EC). This shows a slight decline, highlighting the importance of constraints for precise knowledge learning.

• w/o Fine-grained Components: Identify C_{le} directly by LLM without fine-grained components (abbr. KB-Logic-FC). This leads to a minor performance drop, suggesting that fine-grained components aid in better logical reasoning.



(a) Ablation study. (b) Effect of logical type. (c) Effect of #candidates.

Fig. 4: Ablation study and module & parameter analysis.

3.4 Module and Parameter Analysis

To further analyze the impacts of different KB-Logic components and parameters on the KBQA performance, we conducted the following studies on GrailQA, recognized for its large scale, high quality, and most representativeness.

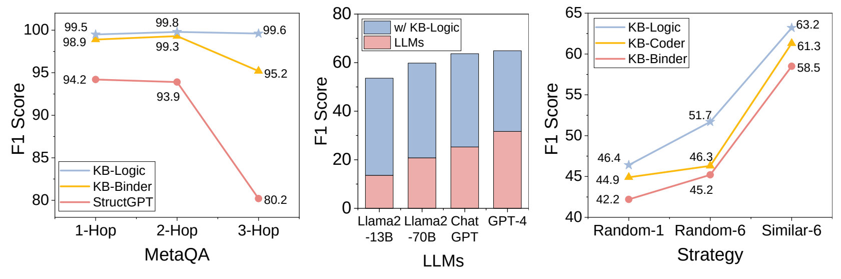
Logical Type Judgment Component. As shown in Figure 4b, we analyzed the impact of the logical type judgment component on KBQA performance by extracting various logical types of questions from the GrailQA validation set. Focusing on one-hop questions to avoid reasoning structure influence, we found significant performance improvements in counting, superlativeness, and comparison types of questions, with slight improvement in general type questions.

Number of Candidates. As shown in Figure 4c, we investigated the performance impact of varying numbers of Logical Skeleton (LK) and Knowledge Component (KC) candidates. For LK candidates, increasing the number of candidates (N = 1, 2, 3, 4, 5, 6) improved performance up to $N = 5$, after which it stabilized. For KC candidates, increasing the number of candidates (M = 10, 25, 50, 100, 200, 300) improved performance until a certain point, beyond which performance declined due to difficulty in distinguishing optimal candidates.

3.5 Performance Comparison

Multi-hop Reasoning Performance Comparison. As shown in Figure 5a, KB-Logic consistently achieves high performance across all reasoning tasks in the MetaQA dataset. Notably, KB-Logic maintains stable accuracy even in multi-hop questions, whereas other methods like StructGPT and KB-Binder exhibit significant declines as the number of hops increases to 3. This robustness may stem from its structured reasoning framework, which ensures accurate knowledge information transfer across deeper reasoning paths.

Performance Comparison of LLMs. As shown in Figure 5b, our experiments on GrailQA indicate that the base performance of various LLMs is generally modest, with significant disparities among different models. However, integrating KB-Logic significantly enhances performance, narrowing the performance gap between models and enabling all models to achieve competitive results. This improvement is attributed to the effectiveness of the decomposition strategy and the generalizability of KB-Logic.



(a) Multi-Hops. (b) LLMs. (c) Model Variants.

Fig. 5: Performance comparison.

Model Variants Analysis. As shown in Figure 5c, our experiments on GrailQA reveal that increasing the number of skeletons consistently enhances model performance. This trend is evident across all methods, with KB-Logic showing the most significant improvement. Additionally, transitioning from a random to a similar strategy markedly boosts F1 scores, indicating the efficacy of strategy alignment. Notably, KB-Logic consistently outperforms KB-Binder and KB-Coder, suggesting superior handling of complex structures and logical reasoning.

4 Conclusion

In this paper, we present KB-Logic, a training-free KBQA framework that utilizes LLMs' few-shot in-context learning ability. KB-Logic breaks down the task of generating logical expressions into simpler sub-tasks, including logical skeleton parsing, knowledge component retrieval, and logical expression discrimination, relying on LLMs' discriminative capabilities rather than generative ones. Additionally, schema checking and sample selection strategies ensure the knowledge provided to LLMs is both concise and high-quality. Experiments show that KB-Logic outperforms state-of-the-art LLM-based methods on KBQA datasets.

References

1. Baek, J., Aji, A.F., Saffari, A.: Knowledge-augmented language model prompting for zero-shot knowledge graph question answering. arXiv:2306.04136 (2023)
2. Berant, J., Chou, A., Frostig, R., et al.: Semantic parsing on freebase from question-answer pairs. In: Proceedings of the 2013 EMNLP. pp. 1533–1544 (2013)
3. Brown, T., Mann, B., Ryder, N., et al.: Language models are few-shot learners. In: Proceedings of the 34th NIPS (2020)
4. Gu, Y., Deng, X., Su, Y.: Don't generate, discriminate: A proposal for grounding language models to real-world environments. In: Proceedings of the 61st ACL. pp. 4928–4949 (2023)
5. Gu, Y., Kase, S., Vanni, M., et al.: Beyond i.i.d.: Three levels of generalization for question answering on knowledge bases. In: Proceedings of the 30th WWW. pp. 3477–3488 (2020)

6. Gu, Y., Su, Y.: Arcaneqa: Dynamic program induction and contextualized encoding for knowledge base question answering. In: Proceedings of the 29th COLING. pp. 1718–1731 (2022)
7. He, G., Lan, Y., Jiang, J., Zhao, W.X., Wen, J.R.: Improving multi-hop knowledge base question answering by learning intermediate supervision signals. In: Proceedings of the 14th ACM international conference on web search and data mining. pp. 553–561 (2021)
8. Ji, Z., Lee, N., Frieske, R., et al.: Survey of hallucination in natural language generation. ACM Computing Surveys **55**(12), 1–38 (2023)
9. Jiang, J., et al.: Structgpt: A general framework for large language model to reason over structured data. In: Proceedings of the 2023 EMNLP. pp. 9237–9251 (2023)
10. Li, T., Ma, X., Zhuang, A., et al.: Few-shot in-context learning on knowledge base question answering. In: Proceedings of the 61st ACL. pp. 6966–6980 (2023)
11. Nie, Z., Zhang, R., Wang, Z., et al.: Code-style in-context learning for knowledge-based question answering. arXiv:2309.04695 (2023)
12. Pourreza, M., Rafiei, D.: DIN-SQL: decomposed in-context learning of text-to-sql with self-correction. In: Proceedings of the 37th NIPS (2023)
13. Saxena, A., Tripathi, A., Talukdar, P.: Improving multi-hop question answering over knowledge graphs using knowledge base embeddings. In: Proceedings of the 58th ACL (2020)
14. Shu, Y., et al.: TIARA: multi-grained retrieval for robust question answering over large knowledge base. In: Proceedings of the 2022 EMNLP. pp. 8108–8121 (2022)
15. Srivastava, A., Rastogi, A., Rao, A., et al.: Beyond the imitation game: Quantifying and extrapolating the capabilities of language models. arXiv:2206.04615 (2022)
16. Su, Y., Sun, H., Sadler, B., et al.: On generating characteristic-rich question sets for qa evaluation. In: Proceedings of the 2016 EMNLP. pp. 562–572 (2016)
17. Sun, H., Dhingra, B., Zaheer, M., et al.: Open domain question answering using early fusion of knowledge bases and text. In: Proceedings of the 2018 EMNLP. pp. 4231–4242 (2018)
18. Sun, Y., Zhang, L., Cheng, G., et al.: Sparqa: skeleton-based semantic parsing for complex questions over knowledge bases. In: Proceedings of the 34th AAAI. pp. 8952–8959 (2020)
19. Wang, X., Li, S., Ji, H.: Code4struct: Code generation for few-shot event structure prediction. In: Proceedings of the 61st ACL. pp. 3640–3663 (2023)
20. Ye, X., et al.: RNG-KBQA: generation augmented iterative ranking for knowledge base question answering. In: Proceedings of the 60th ACL. pp. 6032–6043 (2022)
21. Yih, S.W.t., Chang, M.W., He, X., et al.: Semantic parsing via staged query graph generation: Question answering with knowledge base. In: Proceedings of the 53rd ACL. pp. 1321–1331 (2015)
22. Yih, W.t., et al.: The value of semantic parse labeling for knowledge base question answering. In: Proceedings of the 54th ACL. pp. 201–206 (2016)
23. Yu, D., Zhang, S., et al.: Decaf: Joint decoding of answers and logical forms for question answering over knowledge bases. In: Proceedings of the 11th ICLR (2023)
24. Zhang, L., et al.: FC-KBQA: A fine-to-coarse composition framework for knowledge base question answering. In: Proceedings of the 61st ACL. pp. 1002–1017 (2023)
25. Zhang, Y., Dai, H., Kozareva, Z., et al.: Variational reasoning for question answering with knowledge graph. In: Proceedings of the 32nd AAAI. pp. 6069–6076 (2017)