An interpretable logic KBQA method based on open-source large language models

Bicheng Xu[a] Rong Peng[a*] Yongchang Ding[a] Lin Fang[a]

***Abstract*—Knowledge Base Question Answering (KBQA) aims to find correct answers to natural language questions by reasoning over large-scale knowledge bases. The main challenge is multi-hop reasoning, which requires inferring answers through multiple edges and nodes. Due to high data annotation costs, most datasets only provide final answer annotations, leaving the reasoning process unknown. The model cannot provide the reasoning path along with the answer, making the answer uninterpretable. To address this, we propose a generation-retrieval multi-hop KBQA method combining large language models (LLMs) and the logic programming language Prolog. In the generation phase, we fine-tune the open-source LLMs to generate the logical form of the corresponding prolog representation for the natural language question. In the retrieval phase, the model uses prolog query to reasoning over the KB to get the final answer and the reasoning path. A transparent reasoning path also helps identify and correct errors, enhancing the model's reliability and practicality. Experimental results on two standard KBQA datasets, MetaQA and WebQSP, demonstrate that our method outperforms existing models in both performance and interpretability.**

***Index Terms*—knowledge base question answering, large language model, logic programming**

I. Introduction

A Knowledge Base (KB) is a structured data collection organized as triples of (head entity, relation, tail entity). Large KBs like Freebase, DBPedia, and Wikidata have been widely used in downstream NLP tasks. Knowledge Base Question Answering (KBQA) is a significant research area within this context. Early KBQA focused on answering simple questions, where the head entity of the question is connected to the answer entity by a single relation, known as 1-hop KBQA. In recent years, more research has focused on complex questions involving multi-step reasoning, also known as multi-hop KBQA. As shown in Fig. 1, the question requires a 3-hop reasoning path from the question's topic entity to the answer entity . Existing KBQA methods can be broadly classified into two categories: semantic parsing-based methods and information retrieval-based methods. Retrieval-based methods [1, 2, 16, 19] directly retrieve answers from the knowledge base based on the information provided in the question. Semantic parsing-based methods [8–13]focus on translating questions into intermediate logical forms, such as SPARQL, which are then executed on the knowledge base to obtain the final answer. With the advent of end-to-end neural networks, researchers have increasingly turned to end-to-end deep neural network models for KBQA tasks [1, 2, 12, 16]. These models are popular for their ability to autonomously learn knowledge base representations and network parameters. With the introduction of ChatGPT[23], LLMs have gained recognition for their capabilities in the NLP field. As shown in Fig. 1, for the question "*What types are the films starred by actors in 'The Nine Lives of Fritz the Cat'?*", there are multiple paths from the head entity "*The Nine Lives of Fritz the Cat*" to the answer entity "*Comedy*". However, only the path marked in red is the correct one. If the model only provides the answer entity, it is impossible to know which path was used to retrieve the answer. Therefore, even if the answer is correctly identified, the reasoning process may still be considered unreliable. To overcome these challenges, our approach uses Prolog queries to represent natural language questions as inferable logical forms. Prolog queries, which are essentially first-order logic, offer strong interpretability and reasoning capabilities. We address the challenge of interpretable KBQA by leveraging the strengths of both LLMs and logic programming. We propose a method that fine-tunes an LLM to generate Prolog query statements to represent questions and obtain answers through logical reasoning. The interpretability of the Prolog solving process provides interpretability for KBQA. Specifically, the contributions of this paper are reflected in the following two aspects:

1. Our model is interpretable. We incorporate the logic programming language Prolog into the KBQA task and employ topic entity masking strategy before generating logical forms. This approach not only improves the accuracy of the reasoning process but also ensures traceability and visualization of each step of the reasoning path.
2. Our research explores the performance of open-source LLMs, such as ChatGLM2 and Baichuan2, in addressing KBQA questions. To demonstrate the capabilities of the LLM model, we compare it to the traditional sequence-to-sequence T5 model.Our model achieves state-of-the-art performance on benchmark datasets WebQSP and MetaQA.

II. Related Work

Knowledge Base Question Answering has seen rapid development over the past decade, with various methods developed for knowledge base quality analysis tasks. Most existing KBQA methods can be divided into two categories: retrieval-based methods and semantic parsing-based methods.

*Corresponding Author.
[a]School of Computer Science, Wuhan University, Wuhan, Hubei, China.
{2018302110271, rongpeng, cs_din.yc, 2017301500043}@whu.edu.cn

DOI reference number: 10.18293/SEKE2024-138

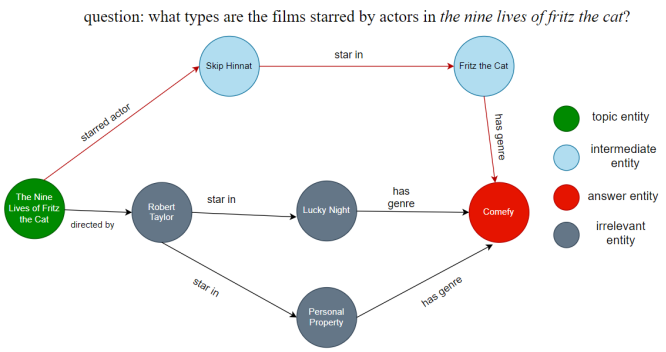

Fig. 1. Instances from the MetaQA-3hop dataset. We use green, red, blue, and gray circles to represent subject entities, accurate responses, intermediate entities, and irrelevant entities, respectively. Red arrows indicate the correct reasoning paths.

Retrieval-based approaches [1, 2, 16, 19, 21] retrieve answers directly from the KB based on the information provided in the question. Several studies [2, 16, 34] utilize classical approaches (e.g., key-value memory networks, variational inference networks, and graph convolutional networks) to perform multi-hop inference in knowledge bases.Alexander et al [16]store ternaries of the knowledge base in a key-value structured memory cache and maintain a memory table for retrieval, which is then used for inference in order to predict the answer to complete the QA. In addition, studies [1, 2] used external texts to deal with knowledge base sparsity, while [5–7] used knowledge base embedding methods to map knowledge bases and questions to vectors for reasoning over knowledge bases. Sun et al [1] proposed GRAFT-Net for dealing with knowledge base sparsity, which combines a KB with a corpus of external entity-linked texts. The approach proposed in the [2] builds a question-specific subgraph from the KB and then augments it with relevant text documents; however, commonly used publicly available datasets do not provide a text corpus about the question. Saxena et al [5] utilized ComplEx Embeddings and RoBERTa models to transform questions and triples in the knowledge base into vector space and designed a special relation matching strategy to solve the problem of incomplete knowledge base. However they both lack the utilization of intermediate signals in the process of reasoning about answers. Approaches based on semantic parsing [8–13, 31] translates questions into intermediate logical forms, like SPARQL or S-expression, and execute on the KB. Some semantic parsing-based approaches (e.g., [8, 11, 13, 18]) use the generated query subgraphs as logical forms. [14, 15] attempt to train sequence-to-sequence models to generate S-expressions as logical forms and use various strategies to enhance the reasoning process. The current challenges for semantic understanding based approaches lie in two main areas, firstly, it is difficult for logical forms to accurately express the semantics of a question when the question is complex. Second, logical parsing must cover a variety of query types for complex questions.Yih et al [30] proposed query graphs as expressive parsing targets, and they showed strong expressiveness in the complex KBQA task. However, they can only be generated by predefined manual rules, which are not applicable to large-scale datasets and complex question types. Gu et al[32] proposed S-expressions, which define functions to transform SPARQL queries on questions, making the logical form shorter, but then need to be converted to SPARQL for querying when reasoning on knowledge bases. Unlike traditional approaches, we chose the logic programming language Prolog[27] as the logical form because it is concise and logical. For example Given the question, "*What is the name of justin bieber brother*". the corresponding Prolog query is "*people.person.sibling_s (Justin Bieber, X), people.sibling_relationship.sibling(X,Y), people.person.gender(Y,Male)*". The simplicity of the Prolog language can effectively improve the accuracy of LLM, and this logical form provides a structured representation of the meaning of the question, without defining any other functions.

III. The Framework of LLMProKBQA

A. Overview

The model adopts a "generation-retrieval" framework, as shown in Fig. 2. It uses fine-tuned LLMs to generate logical forms of questions and retrieves relevant information from the KB, thereby facilitating accurate and natural language understanding and answer generation in KBQA. To fine-tune the open-source LLMs, we adjust the$< question, answer >$pairs from the KBQA datasets to $< question, logical form >$pairs, using the logic programming language Prolog for the logical forms. The fine-tuned LLMs are then used to translate new NL questions into corresponding candidate logical forms through semantic parsing. The question answering module utilizes these logical forms to reason over the knowledge base to obtain the final answer, providing an interpretable reasoning path for the NL question.

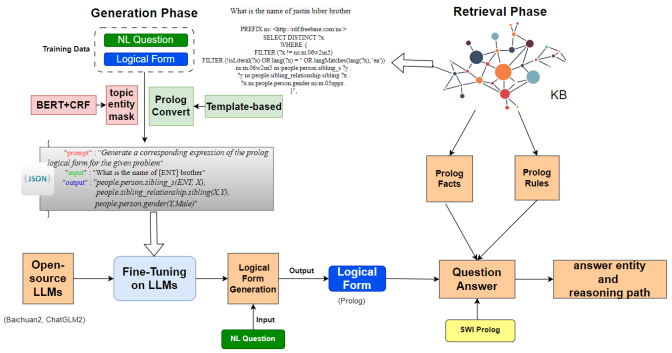

Fig. 2. The Framework of LLMProKBQA.

B. Logical Form Generation

Unlike traditional models that use$< question, answer >$ training data, our model uses $< question, logical form >$ pairs as input for the LLM. To create the training data for fine-tuning the LLMs, we need to generate Prolog reasoning forms for the questions in the training dataset. This process involves two main steps: Topic entity mask and Prolog expression generation.

1) Topic Entity Mask: To ensure the model focuses on the representation of the question and reduces the influence of the topic entity on question modeling, a strategy called "**topic entity masking**" is employed. The reasoning process should prioritize the semantic structure of the question, particularly the predicates, over the topic entity of the given question. The key to visualizing reasoning in KBQA is mapping the information from the NL question to the relations in the KB, which often appear as predicates in the natural language question and do not involve the subject entity. Therefore, to make the model focus on the semantic structure of the question itself, we replace the subject entity in each question with a special token "ENT". We use a BERT model combined with CRF for named entity recognition in questions. The *BERT_tokenizer* function tokenizes the input question $q = \{w1, w2, ..., wn\}$ using BERT's tokenizer, splitting the text into subword units.Each subword is assigned a unique identifier, and special start and end markers ([CLS] and [SEP]) are added. The tokenized sequence is then input into the pre-trained BERT model, which generates contextual embeddings for each token, capturing the token's context information.

$$embeddings = BERT([CLS], tokens, [SEP]) \quad (1)$$

The output from BERT's final layer is extracted as the feature representation for each token. This representation is passed through a fully connected layer and then input into the CRF layer. The CRF layer learns the transition probabilities between labels, enabling it to consider label sequence information during prediction and generate the globally optimal label sequence.

$$CRF\_output = CRF(FullyConnectedLayer(embeddings)) \quad (2)$$

Finally, the model replaces the labeled subject entity with the special token "ENT".

2) Prolog Logical Form Generation: To transform each natural language question into a logical form expressed in Prolog, we need to utilize additional information from the dataset, such as SPARQL queries and question types, to construct corresponding logical predicate mappings for the questions. Since SPARQL queries and question types have fixed structures and specific keywords, we employ a Template-based Matching algorithm to parse and match the text using predefined patterns and rules, enabling accurate identification and extraction of key information from the text data. Based on the fixed syntax structures in SPARQL queries and question types, we design corresponding templates and then map the key information from the templates to logical predicates, thereby constructing Prolog logical expressions.

$$Prolog\_form = PredicateMapping(Template(q)) \quad (3)$$

To enhance the accuracy of template matching, we first tokenize the question, then remove noise by eliminating fixed syntactic structures that do not affect the Prolog logical form. Finally, from the key information obtained through template matching, we design mappings for different matching symbols and Prolog predicates, extracting logical reasoning predicates to construct Prolog logical expressions. By employing template matching, we achieve precise identification and extraction of key information from the text data, ultimately enabling the construction of correct Prolog logical expressions for natural language questions.

C. Fine-tuning large language models

Due to the large parameter size of LLMs, traditional training methods require high hardware demands, resulting in slow training speeds and low efficiency. Therefore, this paper adopts Parameter-Efficient Fine-Tuning (PEFT)[28] to fine-tune LLMs. Fine-tuning is a crucial step in large-scale pre-trained LLMs to adapt them to specific tasks. However, traditional fine-tuning methods often require significant computational resources and storage space. To address this issue, P-Tuning v2[26], LoRA[25]and QLoRA[29] techniques have been proposed, significantly reducing the parameter size and computational requirements during fine-tuning by employing low-rank decomposition and quantization techniques. They decompose the weight matrix of the model into two low-rank matrices. QLoRA quantizes the decomposed low-rank matrices after the adaptation layers, using 8-bit or 16-bit quantization techniques to represent the matrices in low-precision formats.

To focus the model on the task of generating Prolog logical forms for given questions, we adopt a prompt-based approach to fine-tune LLMs. By using prompt-based fine-tuning, no modification to the model's internal parameters is needed; only adjustments to the input prompts are required to fine-tune while maintaining the model's original capabilities. As prompts are explicit text cues, we can debug and optimize them naturally, selecting the most suitable prompt for training the Prolog logical form generation task. In this paper, we use the prompt: "Generate a corresponding expression of the prolog logical form for the given problem." To assess the capabilities of different open-source LLMs in KBQA tasks, we selected the Baichuan2 and ChatGLM2 models, and the traditional sequence-to-sequence model T5 as a comparison. Under the same training dataset conditions, we compared the results of these models to evaluate their performance on KBQA tasks.

D. Prolog-based Answer Retrieval

Due to the excellent performance of fine-tuned LLMs in generating logical forms, we can obtain the logical form and reasoning path for a given question. Traditional semantic parsing methods typically convert the logical forms generated by fine-tuned LLMs into SPARQL and then execute the query on the knowledge base to obtain the final answer. However, the conversion from logical form to SPARQL is time-consuming and unreliable. In our answer retrieval module, we utilize Prolog logical forms, eliminating the need for query conversion. Our logical forms are Prolog queries, a logic programming language executable directly through SWI-Prolog. Before executing the query, we need to import facts

and rules for our module. Each triple in the KB represents a fact. We convert each triple "$(entity1, relation, entity2)$" in the KB to the form "$relation(entity1, entity2)$", with the relation as the Prolog query predicate and the entities as variables to construct the facts. To address the incompleteness of relations in the KB, we define new rules for completing relations after the conversion to Prolog queries. Since Prolog queries are essentially first-order logic expressions, we use first-order logic for reasoning, defining rules to complete certain relations. Our module can discover and construct new inference rules, such as reversibility and transitivity rule, based on the triples. By adding rules, the model automatically adds new triples to the knowledge base to enhance the knowledge graph.For example, when constructing facts in Prolog queries from the triples in the knowledge base, the model discovers that for a relation r, if the pairs of triples$(e1, r, e2)$ and $(e2, r, e1)$ reach a threshold, the model adds reversibility rules for r to rule set. It then constructs corresponding reverse triples for all triples in the knowledge base related to r. After importing facts and rules, the model executes Prolog queries using SWI-Prolog. Since Prolog queries are generated with entity masking, to make the logical query executable, we replace the special token "ENT" with the corresponding question topic entity. Due to the inherent reasoning capability of Prolog, by executing the query, we can retrieve the answer to the question and obtain the reasoning path connecting the question entity and the answer entity. This method not only improves query efficiency but also ensures the interpretability of logical forms and reasoning paths.

IV. EXPERIMENTS

A. Dataset

To evaluate our model, we conducted experiments on two benchmark datasets for the KBQA task:

- **MetaQA** [17]: This dataset consists of over 400,000 single-hop and multi-hop (up to 3 hops) questions in the movie domain, divided into three subsets: MetaQA-1hop, MetaQA-2hop, and MetaQA-3hop.
- **WebQSP** [30]: This dataset comprises 4,737 natural language questions that can be answered using Freebase as the knowledge base. Details of the datasets are shown in Table I.

TABLE I
STATISTICS OF ALL DATASETS.THE *ONE-SHOT DATASET IS CONSTRUCTED BY RANDOMLY SAMPLING ONE DATA INSTANCE FOR EACH QUESTION TEMPLATE FROM THE METAQA TRAINING SET.

datasets	train	dev	test	*one-shot
MetaQA-1hop	96,106	9,992	9,947	161
MetaQA-2hop	118,980	14,872	14,872	210
MetaQA-3hop	114,196	14,274	14,274	150
WebQSP	3,098	-	1,639	-

B. Baselines

For comparison purposes, we evaluate the hit@1 metric of our model on the multi-hop test dataset.The baseline models compared in this experiment are as follows:

- **KV-Mem**[16]: Maintains a memory table for retrieval, storing KB facts encoded as key-value pairs.
- **GraftNet**[1]: Utilizes a variant of graph convolutional networks for multi-hop reasoning on heterogeneous graphs.
- **PullNet**[2]: Trains a graph retrieval module with shortest paths as supervision and performs multi-hop reasoning on retrieved subgraphs along with GraftNet.
- **SRN**[20]: A multi-hop reasoning model in an RL environment, extending reasoning paths on the knowledge base to address multi-hop question answering.
- **EmbedKGQA**[5]: Conducts multi-hop reasoning by matching pre-trained entity embeddings and question embeddings obtained from RoBERTa.
- **NSM**[6]: Utilizes teacher-student networks for intermediate entity representation and answer prediction separately, enhancing model accuracy through bidirectional reasoning.

C. Experimental Results

For LLM, we fine-tuned 100 epochs on WebQSP and 50 epochs on MetaQA, with a batch size of 4. We optimized all models using page-wise AdamW for memory optimization, with a learning rate set to 1e-4. All experiments were conducted on an NVIDIA A30 GPU (24GB), and results are from experiments with random seeds.

1) Results: The results are shown in Table II. The results of the baseline models are taken from their original papers. Experimental results on WebQSP and MetaQA datasets show that our LLM model combined with Prolog performs exceptionally well in KBQA tasks, significantly outperforming all baseline models. On the WebQSP dataset, our model improves the hit@1 metric by approximately 5.6% compared to baseline models.On the MetaQA dataset, our model shows a 2.8% improvement on MetaQA-1hop compared to the second best models. Our overall performance on MetaQA achieves 100%, which previous models could not attain, demonstrating that our model possesses a high level of comprehension and reasoning ability for template-generated questions. When we removed the tpoic entity masking strategy from the model, the hit@1 scores on WebQSP, MetaQA-1hop, and MetaQA-2hop all declined to varying degrees. This demonstrates that our tpoic entity masking strategy helps maintain the semantic structure of the question while enabling the LLM to focus on generating logical forms. our model outperforms previous embedding-based methods (EmbedKGQA) and subgraph-based methods (GraftNet and PullNet) across all benchmark datasets. These significant performance improvements are mainly attributed to the LLM's ability to understand NL questions and the simplicity and logical structure of the Prolog.LLMs excel in NL understanding, enabling more accurate handling and interpretation of questions.

TABLE II
PERFORMANCE COMPARISON OF DIFFERENT METHODS FOR KBQA (HITS@1 IN PERCENT).BOLD FONT DENOTES THE BEST METHODS. LLMPROKBQA* REFERS THE RESULT WITHOUT TOPIC ENTITY MASKING STRATEGY.

Models	WebQSP	MetaQA-1	MetaQA-2	MetaQA-3
KV-Mem	46.7	96.2	82.7	48.9
GraftNet	66.4	97.0	94.8	77.7
PullNet	68.1	97.0	99.9	91.4
SRN	-	97.0	95.1	75.2
EmbedKGQA	66.6	97.5	98.8	94.8
NSM	68.7	97.1	99.9	98.9
LLMProKBQA*	75.3	98.7	99.9	**100**
LLMProKBQA	**79.9**	**100**	**100**	**100**

2) Evaluation on one-shot: From the table, we observe that for the MetaQA dataset, all baseline models achieved excellent results in 1-hop, 2-hop, and 3-hop tasks. Models like EmbedKGQA, NSM, and NSM+h all surpassed 95%, while our model achieved 100%. However, the performance on WebQSP is significantly lower.Upon analysis, questions in MetaQA are generated in batches using templates, with ample training data available for MetaQA, where each dataset contains no more than 300 templates with 100 training examples per template. This abundance of similar questions likely leads to the model's outstanding performance on the original dataset, as it can learn stable patterns from a large number of similar questions. To further analyze the model's performance on MetaQA, we created a new one-shot training dataset by randomly selecting one training example for each question template from the original dataset, shown in TableIII. Each question template in the one-shot training dataset has only one training example, which poses a greater challenge to the model's generalization ability. As seen in TableIII, both the Baichuan and ChatGLM models achieved good results, consistent with the original performance. However, the NSM model experienced a sharp decline in hit@1 scores for 1-hop and 3-hop tasks, with an 8.3% drop in 3-hop, indicating poor generalization ability in handling multi-hop questions. Despite the significant reduction in training data, our model's performance remains excellent, demonstrating strong robustness and generalization ability.

TABLE III
EVALUATION ON ONE-SHOT.

Models	MetaQA-1	MetaQA-2	MetaQA-3
NSM	93.3	97.7	90.6
ChatGLM2+prolog	97.9	**99.8**	**100**
Baichuan+prolog	**98.5**	99.3	**100**

3) Comparison of results from different LLMs: To evaluate the effectiveness of different LLMs in generating final answers for KBQA tasks, we compared the performance of T5, Baichuan2-7B, and ChatGLM2-6B on WebQSP. Results are shown in Table IV. Among all models, Baichuan2-7B outperformed others, indicating its significant advantage in generating Prolog logical forms. The performance of the T5 model gradually improved with the increase in parameter size, with T5-Large outperforming the T5-Base model by 5.6% in generating logical forms. Fine-tuned open-source LLMs (Baichuan2-7B and ChatGLM2-6B) demonstrated stronger semantic parsing abilities, capable of generating more accurate logical forms for given questions.

TABLE IV
RESULTS OF DIFFERENT LLMS IN WEBQSP.

Models	Model size	WebQSP
T5-small	80M	52.1
T5-base	250M	65.6
T5-Large	780M	71.2
Flan-T5-3B	3B	73.4
ChatGLM2-6B	6B	74.8
Baichuan2-7B	7B	**79.9**

4) Interpretability: As shown in Fig. 3, our model exhibits interpretability by generating not only answers to questions but also the reasoning paths behind them, as illustrated in the figure. For a given question "What is the nationality of Barack Obama?", our model first identifies the head entity "*Barack Obama*", then generates intermediate query statements, and finally performs answer retrieval on the KB. The model not only provides the answer "*USA*" but also outputs the reasoning path, presenting all the inferred triples during the reasoning process. From the figure, it can be observed that there are multiple paths from the head entity to the answer entity. The transparency of reasoning paths not only helps verify the answer's reliability but also allows people to clearly see changes in the flow of information during the reasoning process.

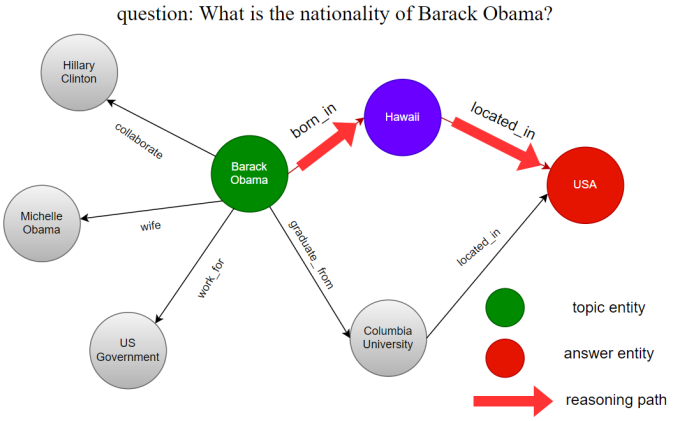

Fig. 3. A case study on 2-hop queation, we use green, blue,red circles denote the topic entity, intermediate entity, answer entity. The red edges are the correct reasoning path.

V. CONCLUSION AND FUTURE RESEARCH

In this paper, we propose an innovative approach for KBQA tasks. We integrate different open-source LLMs into the KBQA task, leveraging their unique understanding of natural language to deeply analyze given NL questions and

extract more semantic information. Unlike traditional semantic parsing-based methods, we focus on generating logical forms before answer inference, selecting the logical programming language Prolog as the intermediate logical form for questions, and employing topic entity masking strategy to keep the LLM focused on the semantic structure. We fine-tune open-source LLMs to generate intermediate representations of questions. Subsequently, we utilize the generated Prolog representation as the logical form for inference. Our answer inference module is capable of reasoning based on the intermediate representation of questions and ultimately outputs answers.We evaluated our model on MetaQA and WebQSP, and the results demonstrate that our model achieves state-of-the-art performance on both standard KBQA benchmark datasets.

REFERENCES

[1] H. Sun, B. Dhingra, M. Zaheer, K. Mazaitis, R. Salakhutdinov, and W. W. Cohen, "Open Domain Question Answering Using Early Fusion of Knowledge Bases and Text," in Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, pp. 4231–4242.

[2] H. Sun, T. Bedrax-Weiss, and W. W. Cohen, "PullNet: Open Domain Question Answering with Iterative Retrieval on Knowledge Bases and Text," in Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing, EMNLP-IJCNLP 2019, pp. 2380–2390.

[3] W. Xiong, M. Yu, S. Chang, X. Guo, and W. Y. Wang, "Improving Question Answering over Incomplete KBs with Knowledge-Aware Reader," in Proceedings of the 57th Conference of the Association for Computational Linguistics, ACL 2019, Florence, Italy, July 28- August 2, 2019, Volume 1: Long Papers, 2019, pp. 4258–4264.

[4] Y. Deng et al., Multi-Task Learning with Multi-View Attention for Answer Selection and Knowledge Base Question Answering. 2018.

[5] A. Saxena, A. Tripathi, and P. P. Talukdar, "Improving Multi-hop Question Answering over Knowledge Graphs using Knowledge Base Embeddings," in Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics, ACL 2020, pp. 4498–4507

[6] G. He, Y. Lan, J. Jiang, W. X. Zhao, and J.-R. Wen, "Improving Multi-hop Knowledge Base Question Answering by Learning Intermediate Supervision Signals," in WSDM '21, The Fourteenth ACM International Conference on Web Search and Data Mining, pp. 553–561.

[7] Y. Cao et al., "Pay More Attention to Relation Exploration for Knowledge Base Question Answering," in Findings of the Association for Computational Linguistics: ACL 2023, pp. 2119–2136.

[8] W. Yih, M. Richardson, C. Meek, M.-W. Chang, and J. Suh, "The Value of Semantic Parse Labeling for Knowledge Base Question Answering," 2016.

[9] J. Berant, A. Chou, R. Frostig, and P. Liang, "Semantic Parsing on Freebase from Question-Answer Pairs," in Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing, EMNLP 2013, pp. 1533–1544.

[10] Y. Lan and J. Jiang, "Query Graph Generation for Answering Multi-hop Complex Questions from Knowledge Bases," in Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics, Jul. 2020, pp. 969–974.

[11] Z.-Y. Chen, C.-H. Chang, Y.-P. Chen, J. Nayak, and L.-W. Ku, "UHop: An Unrestricted-Hop Relation Extraction Framework for Knowledge-Based Question Answering," in Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL-HLT 2019, pp. 345–356.

[12] C. Liang, J. Berant, Q. V. Le, K. D. Forbus, and N. Lao, "Neural Symbolic Machines: Learning Semantic Parsers on Freebase with Weak Supervision," in Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics, ACL 2017, pp. 23–33.

[13] Y. Gu, X. Deng, and Y. Su, "Don't Generate, Discriminate: A Proposal for Grounding Language Models to Real-World Environments," in Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), ACL 2023, pp. 4928–4949.

[14] R. Das et al., "Case-based Reasoning for Natural Language Queries over Knowledge Bases," in Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing, EMNLP 2021, pp. 9594–9611.

[15] X. Ye, S. Yavuz, K. Hashimoto, Y. Zhou, and C. Xiong, "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), ACL 2022, pp. 6032–6043.

[16] A. H. Miller, A. Fisch, J. Dodge, A.-H. Karimi, A. Bordes, and J. Weston, "Key-Value Memory Networks for Directly Reading Documents," in Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing, EMNLP 2016, pp. 1400–1409.

[17] Y. Zhang, H. Dai, Z. Kozareva, A. J. Smola, and L. Song, "Variational Reasoning for Question Answering With Knowledge Graph," in Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence, (AAAI-18), 2018, pp. 6069–6076.

[18] S. Min, V. Zhong, L. Zettlemoyer, and H. Hajishirzi, "Multi-hop Reading Comprehension through Question Decomposition and Rescoring," in Proceedings of the 57th Conference of the Association for Computational Linguistics, ACL 2019, pp. 6097–6109.

[19] A. Hudson and C. D. Manning, "Compositional Attention Networks for Machine Reasoning," 2018.

[20] Y. Qiu, Y. Wang, X. Jin, and K. Zhang, "Stepwise Reasoning for Multi-Relation Question Answering over Knowledge Graph with Weak Supervision," in WSDM '20, 2020, pp. 474–482.

[21] T. N. Kipf and M. Welling, "Semi-Supervised Classification with Graph Convolutional Networks," 2017.

[22] P. Velickovic, G. Cucurull, A. Casanova, A. Romero, P. Liò, and Y. Bengio, "Graph Attention Networks," 2018.

[23] OpenAI. Gpt-4 technical report, 2023

[24] A. Yang et al., Baichuan 2: Open Large-scale Language Models. 2023.

[25] E. J. Hu et al., "LoRA: Low-Rank Adaptation of Large Language Models," 2022.

[26] X. Liu et al., P-Tuning v2: Prompt Tuning Can Be Comparable to Fine-tuning Universally Across Scales and Tasks. 2022.

[27] N. Madani, R. K. Srihari, and K. Joseph, Domain Specific Question Answering Over Knowledge Graphs Using Logical Programming and Large Language Models. 2023.

[28] S. Mangrulkar, S. Gugger, L. Debut, Y. Belkada, and S. Paul.Peft: State-of-the-art parameter-efficient fine-tuning methods. 2022.

[29] T. Dettmers, A. Pagnoni, A. Holtzman, and L. Zettlemoyer, QLoRA: Efficient Finetuning of Quantized LLMs. 2023.

[30] W. Yih, M.-W. Chang, X. He, and J. Gao, "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 of the Asian Federation of Natural Language Processing, ACL 2015, July 26-31, 2015, Beijing, China, Volume 1: Long Papers, 2015, pp. 1321–1331.

[31] K. Xu, Y. Lai, Y. Feng, and Z. Wang, "Enhancing Key-Value Memory Neural Networks for Knowledge Based Question Answering," in Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL-HLT 2019, Minneapolis, MN, USA, June 2-7, 2019, Volume 1 (Long and Short Papers), 2019, pp. 2937–2947.

[32] Y. Gu et al., "Beyond I.I.D.: Three Levels of Generalization for Question Answering on Knowledge Bases," CoRR, vol. abs/2011.07743, 2020.

[33] H. Bast and E. Haussmann, "More Accurate Question Answering on Freebase," in Proceedings of the 24th ACM International on Conference on Information and Knowledge Management, 2015, pp. 1431–1440.

[34] R. Das et al., "Go for a Walk and Arrive at the Answer: Reasoning Over Paths in Knowledge Bases using Reinforcement Learning," 2018.

[35] X. V. Lin, R. Socher, and C. Xiong, "Multi-Hop Knowledge Graph Reasoning with Reward Shaping," in Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, Brussels, Belgium, October 31 - November 4, 2018, 2018, pp. 3243–3253.