EPERM: An Evidence Path Enhanced Reasoning Model for Knowledge Graph Question and Answering

Xiao Long[1], Liansheng Zhuang[1*], Aodi Li[1], Minghong Yao[1], Shafei Wang[2]

[1]University of Science and Technology of China, Hefei 230026, China
[2]Peng Cheng Laboratory, Shenzhen 518000, China
longxiao1997@mail.ustc.edu.cn; lszhuang@ustc.edu.cn

Abstract

Due to the remarkable reasoning ability, Large language models (LLMs) have demonstrated impressive performance in knowledge graph question answering (KGQA) tasks, which find answers to natural language questions over knowledge graphs (KGs). To alleviate the hallucinations and lack of knowledge issues of LLMs, existing methods often retrieve the question-related information from KGs to enrich the input context. However, most methods focus on retrieving the relevant information while ignoring the importance of different types of knowledge in reasoning, which degrades their performance. To this end, this paper reformulates the KGQA problem as a graphical model and proposes a three-stage framework named the Evidence Path Enhanced Reasoning Model (EPERM) for KGQA. In the first stage, EPERM uses the fine-tuned LLM to retrieve a subgraph related to the question from the original knowledge graph. In the second stage, EPERM filters out the evidence paths that faithfully support the reasoning of the questions, and score their importance in reasoning. Finally, EPERM uses the weighted evidence paths to reason the final answer. Since considering the importance of different structural information in KGs for reasoning, EPERM can improve the reasoning ability of LLMs in KGQA tasks. Extensive experiments on benchmark datasets demonstrate that EPERM achieves superior performances in KGQA tasks.

Introduction

Question answering over knowledge graph (KGQA) has garnered significant attention in recent years. It aims to find answers for natural language questions based on knowledge graphs (KGs), such as Freebase (Bollacker et al. 2008) and Wikidata (Vrandečić and Krötzsch 2014), which are built from a large number of triplets consisting of (head entity, relation, tail entity). Since the natural language questions often contain compositional semantics (Lan et al. 2022), exactly understanding the semantic information in the question and identifying the structured knowledge in KGs is very important for KGQA tasks.

Recently, as large language models (LLMs) (OpenAI 2023; Hadi et al. 2023) have demonstrated impressive ability to understand natural language and reasoning abilities in many NLP tasks, LLMs have also shown impressive performance in knowledge graph question answering tasks (Jiang et al. 2022). Currently, retrieval-augmented methods (Wu et al. 2023; Sun et al. 2023; Ding et al. 2024) are popular ones that combine LLMs and KGs for KGQA tasks. Usually, they involve two steps. First, they retrieve the question-related triplets or paths as contextual knowledge from the raw KGs. Then, they leverage these contexts for the LLM to generate the answers. Although retrieval-augmented methods exploit the ability of LLMs for reasoning and have achieved promising results in KGQA tasks (Wu et al. 2023; Sun et al. 2023), they still suffer from the following issues. First, they usually treat the different retrieval information equally to reason the answer, ignoring the differences between retrieved information. Second, the retrieval-augmented generation methods usually treat the retrieval and reasoning processes separately in model learning. The coupling between the retrieval and reasoning processes of the model is low, and there is a lack of a unified framework to model KGQA tasks.

Inspired by the above insights, this paper reformulates the knowledge graph question answering task as a probabilistic graphical model (Koller and Friedman 2009), and proposes a novel framework named Evidence Path Enhanced Reasoning Model (EPERM), which considers the importance of different structural information when reasoning the question answers. Our EPERM involves the subgraph retrieval stage, the evidence path finding stage, and the answer prediction stage. Specifically, in the first stage, the subgraph retriever is proposed to retrieve a subgraph related to the question from the original knowledge graphs. In the second stage, the proposed evidence path finder first generates a series of weighted plans that faithfully support the reasoning of the questions. Then it scores and filters out the weighted evidence path in the subgraph based on the weighted plans. In the final stage, the answer predictor is proposed to use the weighted evidence path to reason the final answer. Since the weight of each evidence path represents the importance of the structural information for reasoning the problem, EPERM can better leverage them to reason the answer. In addition, we design joint fine-tuning strategies to learn the parameters in the retrieval and reasoning processes. Finally, since the entire question-answering process is described as a probabilistic graphical model, EPERM exhibits greater coupling between the retrieval and reasoning stages. Our contributions can be summarized as follows.

*Corresponding author.
Copyright © 2025, Association for the Advancement of Artificial Intelligence (www.aaai.org). All rights reserved.

- This paper reformulates the Knowledge Graph Question Answering (KGQA) problem as a graphical model and proposes a novel framework named the Evidence Path Enhanced Reasoning Model (EPERM), which leverages the reasoning capabilities of Large Language Models (LLMs) and the structure information in KGs. By considering the varying importance of the structural information, our EPERM can achieve better reasoning abilities for KGQA tasks.
- A joint fine-tuning strategy is proposed to improve the reasoning abilities of LLMs guided by the graphical model for KGQA. Compared with previous works on KQGA tasks, EPERM is more unified and exhibits greater coupling between the retrieval and reasoning stages.
- Extensive experiments on two benchmark datasets demonstrate that EPERM achieves superior performances in KGQA tasks and significantly outperforms all types of state-of-the-art methods on these two datasets. Especially on WebQSP, EPERM achieves a 3.6 % relative improvement in Hit@1 score compared to the state-of-the-art methods.

Related Work

Knowledge Graph Question Answering. Conventional KBQA solutions can be categorized into three types: Semantic Parsing-based (SP-based) methods, Information Retrieval-based (IR-based) methods, and Embedding-based methods. SP-based methods parse the question into a structural query (e.g., SPARQL) which can be executed by a query engine to get answers (Lan et al. 2022). ArcaneQA (Gu and Su 2022) dynamically generates the query based on results from previous steps. RnG-KBQA (Ye et al. 2021) first enumerate all possible queries and then rank them to get the final output. These methods heavily rely on the quality of generated queries. If the query is not executable, no answers will be generated. DECAF (Donahue et al. 2014) combines semantic parsing and LLMs reasoning to jointly generate answers, which also reach salient performance on KGQA tasks. However, these methods need to annotate expensive logic forms as supervision or are limited to narrow domains with a few logical predicates (Lan et al. 2022). KG embedding, which aims to encode entities and relations into a continuous vector space (Bordes et al. 2013; Long et al. 2022; Sun et al. 2019; Long et al. 2024a), and its effectiveness has been validated in knowledge graph question answering (KGQA) tasks. Embedding-based methods model the entities and relations in embedding space and design special model architectures to reason answers. KV-Mem (Miller et al. 2016) adopts a Key-Value memory network to store triples for reasoning. EmbedKGQA (Saxena, Tripathi, and Talukdar 2020) and NSM (He et al. 2021) utilize the sequential model to mimic the multi-hop reasoning process. IR-based methods primarily retrieve relevant factual triples or text from Knowledge Graphs (KGs) based on natural language questions and then design special model architectures to reason answers. Early works adopt the page rank or random walk algorithm to retrieve subgraphs from KGs for reasoning (Sun et al. 2018). Recently, to integrate LLMs for KGQA, retrieval augmented methods (Jiang et al. 2022; Luo et al. 2023) aim to leverage the LLMs to reason on the retrieved facts from the KGs to improve the reasoning performance. For example, UniKGQA (Jiang et al. 2022) unifies the graph retrieval and reasoning process into a single model with LLMs. ToG (Sun et al. 2023) uses LLM as an agent to iteratively perform beam search on knowledge graphs to find answers. RoG (Luo et al. 2023) uses LLM to generate relation plans, which are used to retrieve the relative facts from raw KGs for LLMs to conduct faithful reasoning. However, these methods treat the different retrieval information equally to reason the answer, ignoring the differences between retrieved information. EPERM proposes to retrieve and score the evidence paths, which consider the different importance of the structural information for better reasoning the answers.

Large Language Models. With the launch of ChatGPT and GPT-4 (OpenAI 2023), displaying the prowess of decoder-only large language models (LLMs) with a vast number of parameters that exhibit emergent phenomena, many traditional NLP tasks are becoming simplified (Hadi et al. 2023). Subsequently, open-source models like Llama-2-7B (Touvron et al. 2023), ChatGLM2-6B (Zeng et al. 2022) and Qwen-Chat (Bai et al. 2023) emerged and can be supervised fine-tuned (SFT) using instruction-tuning technologies (Zhang et al. 2023) such as LoRA (Hu et al. 2021), QLoRA (Dettmers et al. 2024), P-Tuning v2 (Liu et al. 2021), and Freeze (Geva et al. 2020), enhancing the capabilities of LLMs for specific tasks. Additionally, Chain-of-Thought (CoT) (Wei et al. 2022) has been shown to be effective in enhancing LLM reasoning. It creates a series of prompt instances according to reasoning logic under a few-shot learning paradigm in order to improve LLM's performance on complex tasks. In this paper, EPERM employs the instruction-tuning technique to fine-tune open-source LLMs, which consists of the subgraph retriever, evidence path finder, and answer predictor. All the modules in EPERM are joint fine-tuning to learn the parameters.

Methodology

Overall, the framework of the Evidence Path Enhanced Reasoning Model (EPERM) is shown in Figure 2, which contains the subgraph retriever module, the evidence path finder module and the answer predictor module. In the first stage, EPERM first uses the fine-tuned LLM to retrieve a subgraph related to the question from the original knowledge graph. In the second stage, the proposed evidence path finder first generates a series of weighted plans that faithfully support the reasoning of the questions. Then it scores and filters out the weighted evidence path in the subgraph based on the weighted plans. Finally, EPERM uses the evidence paths with their importance score to reason the final answer. In the following subsections, we first formally define the KGQA task. Then, we introduce the details of the proposed method.

Problem Definition

A knowledge graph typically consists of a set of triples, denoted by $\mathcal{G} = \{(e, r, e^{'})|e, e^{'} \in E, r \in R\}$, where E and R denote the entity set and relation set, respectively.

Knowledge Graph Question Answering (KGQA) is a typical reasoning task based on KGs. Given a natural language question Q_n and a KG $\mathcal{G}$, the task aims to design a function f to predict answers $\mathcal{A}_n$ based on knowledge graph $\mathcal{G}$, i.e., $\mathcal{A}_n = f(Q_n, \mathcal{G})$. Following previous works (Zhang et al. 2022) , we assume the topic entities $\mathcal{T}_n$ mentioned in Q_n and answers $\mathcal{A}_n$ are labeled and linked to the corresponding entities in $\mathcal{G}$, i.e., $\mathcal{T}_n, \mathcal{A}_n \subseteq E$. Additionally, given a question Q_n and an answer A_n, the i-th evidence path instance connecting e_{Q_n} and e_{A_n} in KGs is defined as $P_{\mathcal{P}_i(e_{Q_n}, e_{A_n})} = e_{Q_n} \xrightarrow{r_1^i} e_1 \xrightarrow{r_2^i} \cdots \xrightarrow{r_l^i} e_{A_n}$. The corresponding i-th plan $p^i = \{r_1^i, r_2^i, \cdots r_l^i\}$ can be considered a faithful plan for reasoning the question Q_n. And $\mathcal{P}_i(e_{Q_n}, e_{A_n}) = \{p^i | i = 1, \cdots, s\}$ is a set of plans to the question Q_n, which s is the number of plans.

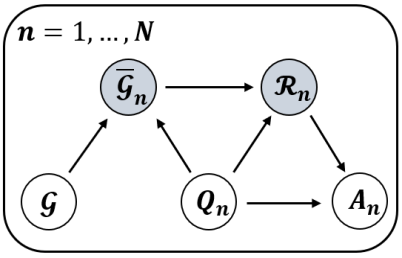


Figure 1: The directed graphical model of KGQA tasks.

The EPERM Framework

We start by formalizing the knowledge graph question answering in a probabilistic way, Given a question Q_n and its answers $\mathcal{A}_n$, we formalize the KBQA problem as to model the probabilistic distribution $P(\mathcal{A}_n|\mathcal{G}, Q_n)$. We introduce two latent variables: a question-related subgraph $\overline{\mathcal{G}}$ and a series of evidence paths $\mathcal{R}_n$ help to reason the question Q_n. Then the KGQA task can be reformulated as a probabilistic graphical model in Figure 1. Based on the independence among the variables in the directed graph and following the d-separation principle (Pearl 2009), the proposed model (objective distribution) $P(\mathcal{A}_n|\mathcal{G}, Q_n)$ can be reformulated as below (we include these details in the Appendix):

$$P_\theta(\mathcal{A}_n|\mathcal{G}, Q_n) = \Sigma_{\mathcal{R}_n}\Sigma_{\overline{\mathcal{G}}} P_\theta(\mathcal{A}_n|\mathcal{R}_n, Q_n) P_\theta(\mathcal{R}_n|\overline{\mathcal{G}}, Q_n) P_\theta(\overline{\mathcal{G}}|\mathcal{G}, Q_n). \quad (1)$$

In the above equation, the proposed EPERM can be divided into three parts. The first part is the subgraph retriever module, which is described by $P_\theta(\overline{\mathcal{G}}|\mathcal{G}, Q_n)$. It aims to retrieve a question-related subgraph from the KGs. The second part is the evidence path finder module, which is described by $P_\theta(\mathcal{R}_n|\overline{\mathcal{G}}, Q_n)$. Its aim is to find and evaluate evidence paths for the subsequent reasoning. Finally, the answer predictor module is formulated by $P_\theta(\mathcal{A}_n|\mathcal{R}_n, Q_n)$, which leverages the weighted evidence paths to reason the final answer. The following sections will provide a detailed introduction to the three modules of EPERM and its training objectives.

Subgraph retriever module. The subgraph retriever module aims to calculate $P_\theta(\overline{\mathcal{G}}|\mathcal{G}, Q_n)$ for any $\overline{\mathcal{G}}$, which is intractable as the latent variable $\overline{\mathcal{G}}$ is combinatorial in nature. To avoid enumerating $\overline{\mathcal{G}}$, we propose to expand top-K paths relevant to Q_n from the topic entities. Specifically, path expansion starts from a topic entity $\mathcal{T}_n$ and follows a sequential decision process. At the beginning of the iteration, the relation expansion phase first searches out all relations $\{r_i^0\}_{i=1}^N$ linked to the topic entity $\mathcal{T}_n$. Then, we select the top K relations $\{r_i^0\}_{i=1}^K$ by using the fine-tuned LLM to score the relevance of each $\{r_i^0\}_{i=0}^N$ to the question Q_n.

$$S(Q_n, r) = LLM_\theta(r, Q_n). \quad (2)$$

The scoring procedure is completed by executing a predefined formal query shown in the Appendix. Then, we retrieve the corresponding tail entities $\{E_j\}_{j=1}^K$ connected to corresponding K relations. Normally, at D-th iteration, we still perform a top-K beam search from current entities to get the K relations. In this way, we can get the subgraph $\overline{\mathcal{G}}$ related to question Q_n.

Evidence path finder module. This module aims to score and filter out a series of weighted evidence paths that faithfully support the reasoning of the questions. First, it generates a series of latent weighted plans $\mathcal{P}_n = \{p^1, \cdots, p^s\}$ for answering the question Q_n based on the subgraph $\overline{\mathcal{G}}$ in the previous stage. This process is defined by the distribution $P_\theta(\mathcal{P}_n|\overline{\mathcal{G}}, Q_n)$, which specifically modeled by a fine-tuned LLM called $Generator_\theta$. Specifically, the $Generator_\theta$ generates compositional plans by only considering the question Q_n and the subgraph $\overline{\mathcal{G}}$, which allows these plans to generalize across entities in KGs. The methods for fine-tuning the generator will be explained in detail in the following sections. To utilize the instruction-following ability of LLMs (Zhang et al. 2023), we design a simple instruction template that prompts LLMs to generate the compositional plan in $\mathcal{P}_n = \{p^1, \cdots, p^s\}$ and their scores. For each compositional plan p^j in the $\mathcal{P}_n$, it can be viewed as a sequence of relations $\{r_1^j, r_2^j \cdots, r_l^j\}$. The $\{r_i^j\}_{i=1}^l$ is the

Algorithm 1: Inference Stage

Input: KG $\mathcal{G}$, question Q_n, topic entities $\mathcal{T}_n$;
Output: Answer $\mathcal{A}_n$;
$\overline{\mathcal{G}} \leftarrow SubgraphRetriever(\mathcal{G}, Q_n)$
Evidence path $\mathcal{R}_n \leftarrow []$
$\mathcal{P}_n = \{p^1, \cdots, p^s\} \leftarrow Generator(Q_n)$
foreach $e_T \in \mathcal{T}_n$ **do**
  $E_0^{e_T} \leftarrow [e_T]$
  foreach $p^j \in \mathcal{P}_n$ **do**
    for $i \leftarrow 1$ **to** $\text{length}(p^j) + 1$ **do**
      $E_i^{e_T} \leftarrow SearchAdj(e_T, r_{i-1}^j, \overline{\mathcal{G}})$
      $E_i^S \leftarrow S(Q_n,\ E_i^{e_T})$
      $E_i^{filter} \leftarrow topK(E_i^S)$
      $\mathcal{R}_n$.append($[E_{i-1}^{e_T}, p_i^j, E_i^{filter}]$)
      $E_i^{e_T} \leftarrow E_i^{filter}$
    end
  end
end
$\mathcal{A}_n \leftarrow AnswerPredictor(\mathcal{R}_n)$

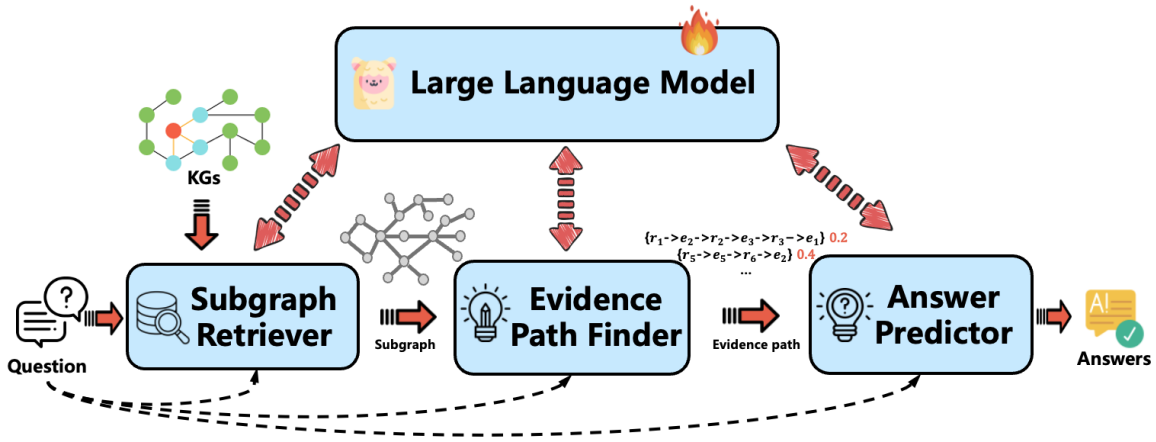


Figure 2: Overview of the proposed EPERM framework. The subgraph retriever module aims to retrieve the question-related subgraph. The evidence path finder module aims to find and score the importance of evidence reasoning paths. The answer predictor module aims to reason the final answer based on the weighted evidence paths.

body of the plan, where l is the max hops for the plan, and s is the total number of the weighted plans for one question Q_n. After obtaining the weighted plans $\mathcal{P}_n$, we need to further retrieve and score the importance of evidence reasoning paths $\mathcal{R}_n$. For each plan $p^j = \{r_i^j\}_{i=1}^l$, a path tree can be induced by filling in the intermediate entities along the plan, i.e., $T(j) = (\mathcal{T}_n, r_1^j, E_1, \cdots, r_l^j, E_l)$. In general, given a head entity, the multi-semantics of the relations (Long et al. 2024b) often lead to multiple tail entities. Therefore, each $E_i \in \{E_1, \cdots, E_l\}$ can be represented $E_i = \{e_i^m\}_{m=1}^V$, which is an entity set and it leads to a situation where a plan connects multiple paths. However, not all entities in the path help answer the question, and we further score the relevance between question Q_n and the entities by using the information surrounding them. The prompt of scoring entities shown in the Appendix.

$$E_i^S = S(Q_n, SearchAdj(e_{i-1}^m, r_{i-1}^j, \overline{\mathcal{G}})). \tag{3}$$

The $SearchAdj(e_{i-1}^m, r_{i-1}^j, \overline{\mathcal{G}})$ is used to get all the adjacent entities of e_{i-1}^m given the pre-relation r_{i-1}^j in $\overline{\mathcal{G}}$. After obtaining the scores for all adjacent entities, we filter the top S score entities to form corresponding top S paths in every hop. Finally, by multiplying the scores on the paths, we get the evidence path $\mathcal{R}_n$.

Answer predictor module. The answer predictor module takes the question Q_n and the evidence paths $\mathcal{R}_n$ to generate answers $\mathcal{A}_n$, which defined by $P_\theta(\mathcal{A}_n|\mathcal{R}_n, Q_n)$. Similarly, we design a reasoning instruction prompt to guide the fine-tuned LLMs to conduct reasoning based on the evidence path $\mathcal{R}_n$. The details of the prompts can be found in the Appendix. Finally, given the input of the question Q_n and a raw knowledge graph $\mathcal{G}$, the pseudocode for the complete inference process is presented in Algorithm 1.

Optimization Framework

Next, we introduce how to optimize the EPERM framework. Since the LLMs have zero knowledge of the relations contained in KGs. Therefore, LLMs cannot directly generate weighted plans $\mathcal{P}_n$ and the evidence paths $\mathcal{R}_n$ grounded by KGs. Moreover, LLMs might not understand the evidence paths correctly and conduct reasoning based on them. To address these issues, we design a joint instruction tuning task. The objective function in equation 1 can be optimized by maximizing the evidence lower bound (ELBO) (Hoffman and Johnson 2016), which is formulated as:

$$\begin{aligned} logP(\mathcal{A}_n|\mathcal{G}, Q_n) \geq\ & \mathbb{E}_{\mathcal{R}_n \sim q_1}[logP(\mathcal{A}_n|\mathcal{R}_n) \\ & - D_{KL}(q_2(\overline{\mathcal{G}}|\mathcal{G}, Q_n)||P(\overline{\mathcal{G}}|\mathcal{G}, Q_n))] \\ & - \mathbb{E}_{\overline{\mathcal{G}} \sim q_2}[D_{KL}(q_1(\mathcal{R}_n|A_n)||P(\mathcal{R}_n|\overline{\mathcal{G}}, Q_n))]. \end{aligned} \tag{4}$$

where $q_1(\mathcal{R}_n|A_n)$ denotes the posterior distribution of faithful evidence paths grounded by KGs and $q_2(\overline{\mathcal{G}}|\mathcal{G}, Q_n)$ denotes the posterior distribution of subgraph. Since we define how to retrieve the subgraph, the posterior distribution of the subgraph is known and the parameters in equation 2 can be learned in the evidence path finding stage. So, we need to optimize the first and the third items in equation 4, which represent to evidence path finder module and the answer predictor module, respectively. We will provide a detailed introduction to these two parts.

Evidence path finder module optimization. To optimize the evidence path finder module, we aim to distill the knowledge from KGs into LLMs to generate faithful evidence paths. This can be achieved by minimizing the KL divergence with the posterior distribution of faithful evidence paths $q_1(\mathcal{R}_n)$, which can be approximated by the valid plans $\mathcal{P}_n$ in KGs. Specifically, given a question Q_n and plans $\mathcal{P}_n$, we may retrieve many candidate answer entities $f_{Q_n}(\mathcal{P}_n)$ in the knowledge graph $\overline{\mathcal{G}}$. We select the plan, in which the ratio of the answer entity to all candidate entities is greater than the threshold value. The posterior distribution distribution

$q_1(\mathcal{R}_n)$ can be formally approximated as:

$$q_1(\mathcal{R}_n) \simeq q_1(\mathcal{R}_n|A_n, Q_n) = \begin{cases} 1, & \frac{N(A_n \in f_{Q_n}(\mathcal{P}_n))}{||f_{Q_n}(\mathcal{P}_n)||} \geq t \\ 0, & else. \end{cases} \tag{5}$$

Therefore, the KL divergence can be calculated as

$$\mathcal{L}_{find} = -\sum_{\mathcal{R}_n \in q_1(\mathcal{R}_n)} logP_\theta(\mathcal{R}_n|Q_n). \tag{6}$$

By optimizing $\mathcal{L}_{find}$, we maximize the probability of LLMs generating faithful plans $\mathcal{P}_n$ and the corresponding evidence paths $\mathcal{R}_n$ by distilling the knowledge from KGs.

Answer predictor module optimization. To optimize the answer predictor module, we aim to enable LLMs to conduct the final answer based on the evidence paths $\mathcal{R}_n$. By utilizing the evidence paths $\mathcal{R}_N$ formed from the N sampled evidence paths to approximate the expectation, the objective function of reasoning optimization can be written as follows:

$$\mathcal{L}_{reasoning} = logP_\theta(\mathcal{A}_n|Q_n, \mathcal{R}_N). \tag{7}$$

The final objective function of EPERM is the combination of the finding and reasoning optimization, which can be formulated as:

$$\mathcal{L} = \mathcal{L}_{find} + \mathcal{L}_{reasoning}. \tag{8}$$

We use the same LLM for both the evidence path finder module and the answer predictor module, which are jointly trained on two instruction-tuning tasks. In this way, EPERM can better generate more accurate evidence reasoning paths and derive the final answers based on these evidence paths and their importance scores.

Experiment

In this section, we first introduce the experiment settings including datasets, baselines, and evaluation protocols. Secondly, we compare EPERM with competitive models and demonstrate its superiority. Thirdly, we conduct a series of ablation studies to analyze the importance of the three modules in the EPERM. Then, we analyze the impact of two important parameters on the proposed model. Finally, we do the case study to exploit how the EPERM finds the evidence paths and reasons the answers based on them.

Experiment Setup

Datasets. We evaluate the proposed EPERM on two benchmarks, WebQuestionSP (WebQSP) (Yih et al. 2016) and Complex WebQuestion (CWQ) (Talmor and Berant 2018), which contain up to 4-hop questions. The statistics of the datasets are given in Table 1. Freebase (Bollacker et al. 2008) is the background knowledge graph for both datasets, which contains around 88 million entities, 20 thousand relations, and 126 million triples.

Evaluation Metrics. Following previous works (Luo et al. 2023), we use Hits@1 and F1 as the evaluation metrics. Hits@1 measures the proportion of questions whose top-1 predicted answer is correct. Since a question may correspond to multiple answers, F1 considers the coverage of all answers, which balances the precision and recall of the predicted answers.

Datasets	#Train	#Test	Max #hop
WebQSP	2826	1628	2
CWQ	27639	3531	4

Table 1: Statistics of datasets.

Baseline Models. We compare EPERM with the three types of KGQA methods. **Embedding based methods:** KV-Mem (Miller et al. 2016), EmbedKGQA (Saxena, Tripathi, and Talukdar 2020), NSM (He et al. 2021), TransferNet (Shi et al. 2021), KGT5 (Saxena, Kochsiek, and Gemulla 2022) and BAMnet (Chen, Wu, and Zaki 2019). **Retrieval based methods:** GraftNet (Sun et al. 2018), GrailQA Ranking (Gu et al. 2021) PullNet (Sun, Bedrax-Weiss, and Cohen 2019), SR+NSM (Zhang et al. 2022), SR+NSM+E2E (Zhang et al. 2022), BeamQA (Atif, El Khatib, and Difallah 2023). **LLM based methods:** LLaMA2-Chat-7B (Touvron et al. 2023), ChatGPT+CoT (Luo et al. 2023), EPR+NSM (Ding et al. 2024), UniKGQA (Jiang et al. 2022), KD-CoT (Wang et al. 2023), RoG (Luo et al. 2023), StructGPT (Jiang et al. 2023), ToG+ChatGPT (Sun et al. 2023)

Implementations details. For EPERM, we use LLaMA2-Chat-7B (Touvron et al. 2023) as the LLM backbone, which is instruction finetuned on the training split of WebQSP and CWQ as well as Freebase for 5 epochs. We generate the top-6 and top-5 weighted plans using beam-search for each question in WebQSP and CWQ respectively. Then it scores and filters out weighted evidence plans. For LLMs, we use zero-shot prompting to conduct KGQA. Our model is trained on 8 Nvidia A40 GPUs.

Main Results

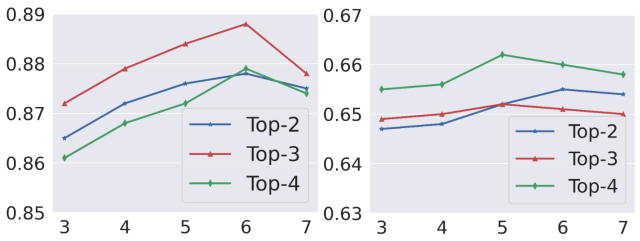
We present the main results on two KGQA datasets(CWQ, WebQSP) in Table 2. Our observations based on the results are as follows. First, retrieval-based approaches outperform embedding-based methods by retrieving relevant subgraphs from KGs, which reduces reasoning complexity. Furthermore, SR+NSM and SR+NSM+E2E adopt relation paths-based retrieval which achieves better performance, highlighting the importance of relation paths. Compared to these two types of traditional methods, EPERM achieves remarkable improvement across all metrics on two datasets. Specifically, it achieves a 19.3% (27.7% relative), and 16.0% (31.8% relative) increase in Hits@1 on the WebQSP and CWQ respectively. Second, compared to the methods of jointly using KGs and LLMs, EPERM still achieves improvement across all metrics on two datasets. Specifically, it achieves a 3.1% (3.6% relative), and 3.6% (5.8% relative) increase in Hits@1 scores over the SOTA model on the WebQSP and CWQ respectively. In conclusion, these results illustrate that by decomposing the KGQA task into three stages, EPERM is able to find the more accurate evidence paths that are highly relevant to the questions and have different weights to help the reasoning stage, assisting the answer predictor to achieve better reasoning performance.

Type	Methods	WebQSP		CWQ	
		Hits@1 ↑	F1 ↑	Hits@1 ↑	F1 ↑
Embedding Based	KV-Mem (Miller et al. 2016)	46.7	34.5	18.4	15.7
	EmbedKGQA (Saxena, Tripathi, and Talukdar 2020)	66.6	-	45.9	-
	NSM (He et al. 2021)	68.7	62.8	47.6	42.4
	TransferNet (Shi et al. 2021)	71.4	-	48.6	-
	KGT5 (Saxena, Kochsiek, and Gemulla 2022)	56.1	-	36.5	-
	BAMnet (Chen, Wu, and Zaki 2019)	55.6	51.8	-	-
Retrieval Based	GraftNet (Sun et al. 2018)	66.4	60.4	36.8	32.7
	GrailQA Ranking (Gu et al. 2021)	-	70.0	-	-
	BeamQA (Atif, El Khatib, and Difallah 2023)	73.3	-	-	-
	PullNet (Sun, Bedrax-Weiss, and Cohen 2019)	68.1	-	45.9	-
	SR+NSM (Zhang et al. 2022)	68.9	64.1	50.2	47.1
	SR+NSM+E2E (Zhang et al. 2022)	69.5	64.1	49.3	46.3
	EPR+NSM (Ding et al. 2024)	71.2	-	60.6	-
LLM Based	LLaMA2-Chat-7B (Touvron et al. 2023)	64.4	-	34.6	-
	UniKGQA (Jiang et al. 2022)	77.2	<u>72.2</u>	51.2	49.1
	KD-CoT (Wang et al. 2023)	68.6	52.5	55.7	-
	ChatGPT+CoT (Luo et al. 2023)	75.6	-	48.9	-
	RoG (Luo et al. 2023)	<u>85.7</u>	70.8	<u>62.6</u>	<u>56.2</u>
	StructGPT (Jiang et al. 2023)	72.6	–	–	–
	ToG+ChatGPT (Sun et al. 2023)	76.2	–	58.9	–
	EPERM (Ours)	**88.8**	**72.4**	**66.2**	**58.9**

Table 2: Performance comparison with different baselines on the two KGQA datasets. The best results are in bold and the second best results are underlined.

Ablation Study

First, we conduct a series of ablation studies to analyze the importance of the weighted evidence paths for the performance of the subsequent answer predictor. We compare three variants: 1) w/o evidence path finder, where we remove the evidence path finder and perform the answer predictor directly. 2) w/o scoring and filtering out the evidence paths in the evidence path finder, where we do not filter the evidence path by weighted plans. The results are shown in Table 3. Based on the results, it is obvious that the performance of the answer predictor will be greatly reduced if we remove the evidence path finder. This is because the input is solely the question, causing the model to degrade into LLM that directly answers the questions. Additionally, if we do not score and filter out the evidence paths during the evidence path finding stage, it will also lead to a decrease in the final performance of the answer predictor. Further scoring and filtering of evidence paths can take into account the varying contributions of different evidence paths to question reasoning. All these results demonstrate the effectiveness of weighted evidence paths for the performance of the subsequent answer predictor. Second, to analyze the importance of the answer predictor, we remove the answer predictor and use all answers from the weighted evidence paths as results. The results are shown in Table 4. From these results, it can be inferred that the answer predictor can further infer and judge from the weighted evidence paths, and obtain more accurate results. Although removing the answer predictor leads to a high recall rate due to an increased number of answers, precision significantly drops.



(a) Comparison on WebQSP (b) Comparison on CWQ

Figure 3: The Hit@1 scores of EPERM with the total number of generated plans s and the number of Top-S path filtered in every hop.

Influence of hyperparameters

In this subsection, we conduct two experiments to analyze the impact of the total number of generated plans s and the number of filtering paths S on the proposed model. Firstly, we change the number of generated plans s. From the figure 3, it can be inferred that when we fix the number of filtering paths S, as the number of plans increases, the performance of the model initially rises and then falls. This is because when s is too small, the coverage rate of the answers is low,

Methods	WebQSP		CWQ	
	Hits@1	F1	Hits@1	F1
EPERM	88.8	72.4	66.2	58.9
EPERM w/o filtering path	84.2	69.1	61.3	55.8
EPERM w/o evidence path	66.2	50.3	36.8	35.7

Table 3: Ablation on the evidence path finder module.

Methods	WebQSP		CWQ	
	Hits@1	Recall	Hits@1	Recall
EPERM	88.8	76.4	66.2	60.9
EPERM w/o answer predictor	62.3	79.8	33.1	66.2

Table 4: Ablation on the answer predictor module.

making it difficult to cover all correct answers. As s increases, more relevant information to the query is retrieved, leading to a higher answer coverage rate and improved model performance. However, as s continues to increase, it introduces unnecessary noise, which can degrade the performance of the model. In addition, an appropriate s is of great significance to the model's performance. Specifically, for the WebQSP, the best s is 6. While for the CWQ the best s is 5. Secondly, we change the number of filtering top S paths in every hop. We can see that an appropriate filtering path number S plays a crucial role in the model's performance. A smaller S potentially removes more irrelevant information but also risks discarding correct information along the way. Conversely, a larger S might introduce noise and degrade the performance of the model. Typically, in the WebQSP dataset, the suitable S is 3, whereas a value of 4 is appropriate for the CWQ dataset.

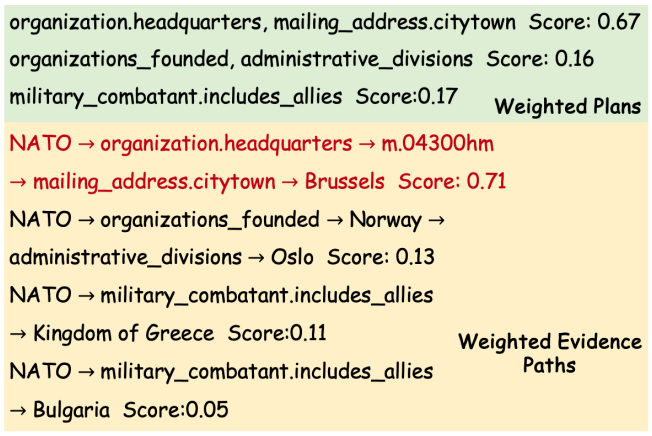


Figure 4: Example of EPERM reasoning based on weighted evidence paths.

Case Study

Finally, we explore how the EPERM reasoning the answers based on the weighted evidence paths. We illustrate a case study in Figure 4. We can see that for the question "Where are the NATO headquarters located?", EPERM can generate a series of weighted plans and then it scores and filters out the weighted evidence paths in the subgraph based on the weighted plans. Although these paths are related to the problem, they still have different confidence scores to reason the questions. If we treat each path equally, it will degrade the reasoning performance. For example, the first path "NATO $\xrightarrow{\text{organization.headquarters}}$ m.04300hm $\xrightarrow{\text{mailing\_address.citytown}}$ Brussels" is more likely to reason the final result. Because the question emphasizes where NATO's headquarters is located. The other evidence paths e.g., "NATO $\xrightarrow{\text{organizations\_founded}}$ Norway $\xrightarrow{\text{administrative\_divisions}}$ Oslo" focus on where NATO's various departments are located, which are supposed to have lower confidence in reasoning the answer. In this way, the Answer Predictor in the EPERM can better make the final choice in the reasoning stage.

Conclusion

In this paper, we propose a novel framework called the Evidence Path Enhanced Reasoning Model (EPERM) to address RAG-based knowledge graph question answering tasks. This framework explores the integration of the generative and reasoning capabilities of large language models (LLMs) with prior knowledge in knowledge graphs for faithful reasoning. We reformulate the KGQA task as a graphical model comprising three stages. In the first stage, EPERM utilizes a fine-tuned LLM to retrieve a subgraph related to the question from the original knowledge graph. In the second stage, the evidence path finder generates a series of weighted plans that reliably support the reasoning process. It then scores and filters the weighted evidence paths within the subgraph based on these plans. Finally, in the third stage, the answer predictor leverages the weighted evidence path to reason the final answer. Since the weight of each evidence path indicates the different importance of the structural information for reasoning the question, EPERM can better leverage them to reason the answer. Extensive experiments on benchmark datasets demonstrate that EPERM achieves superior performances in KGQA tasks.

Acknowledgements

This work was supported by the National Natural Science Foundation of China under Grant No.U20B2070, No.61976199.

References

Atif, F.; El Khatib, O.; and Difallah, D. 2023. Beamqa: Multi-hop knowledge graph question answering with sequence-to-sequence prediction and beam search. In *Proceedings of the 46th International ACM SIGIR Conference on Research and Development in Information Retrieval*, 781–790.

Bai, J.; Bai, S.; Chu, Y.; Cui, Z.; Dang, K.; Deng, X.; Fan, Y.; Ge, W.; Han, Y.; Huang, F.; et al. 2023. Qwen technical report. *arXiv preprint arXiv:2309.16609*.

Bollacker, K.; Evans, C.; Paritosh, P.; Sturge, T.; and Taylor, J. 2008. Freebase: a collaboratively created graph database for structuring human knowledge. In *Proceedings of the 2008 ACM SIGMOD international conference on Management of data*, 1247–1250.

Bordes, A.; Usunier, N.; Garcia-Duran, A.; Weston, J.; and Yakhnenko, O. 2013. Translating embeddings for modeling multi-relational data. *Advances in neural information processing systems*, 26.

Chen, Y.; Wu, L.; and Zaki, M. J. 2019. Bidirectional attentive memory networks for question answering over knowledge bases. *arXiv preprint arXiv:1903.02188*.

Dettmers, T.; Pagnoni, A.; Holtzman, A.; and Zettlemoyer, L. 2024. Qlora: Efficient finetuning of quantized llms. *Advances in Neural Information Processing Systems*, 36.

Ding, W.; Li, J.; Luo, L.; and Qu, Y. 2024. Enhancing complex question answering over knowledge graphs through evidence pattern retrieval. In *Proceedings of the ACM on Web Conference 2024*, 2106–2115.

Donahue, J.; Jia, Y.; Vinyals, O.; Hoffman, J.; Zhang, N.; Tzeng, E.; and Darrell, T. 2014. Decaf: A deep convolutional activation feature for generic visual recognition. In *International conference on machine learning*, 647–655. PMLR.

Geva, M.; Schuster, R.; Berant, J.; and Levy, O. 2020. Transformer feed-forward layers are key-value memories. *arXiv preprint arXiv:2012.14913*.

Gu, Y.; Kase, S.; Vanni, M.; Sadler, B.; Liang, P.; Yan, X.; and Su, Y. 2021. Beyond IID: three levels of generalization for question answering on knowledge bases. In *Proceedings of the Web Conference 2021*, 3477–3488.

Gu, Y.; and Su, Y. 2022. ArcaneQA: Dynamic program induction and contextualized encoding for knowledge base question answering. *arXiv preprint arXiv:2204.08109*.

Hadi, M. U.; Qureshi, R.; Shah, A.; Irfan, M.; Zafar, A.; Shaikh, M. B.; Akhtar, N.; Wu, J.; Mirjalili, S.; et al. 2023. A survey on large language models: Applications, challenges, limitations, and practical usage. *Authorea Preprints*.

He, G.; Lan, Y.; Jiang, J.; Zhao, W. X.; and Wen, J.-R. 2021. 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*, 553–561.

Hoffman, M. D.; and Johnson, M. J. 2016. Elbo surgery: yet another way to carve up the variational evidence lower bound. In *Workshop in Advances in Approximate Bayesian Inference, NIPS*, volume 1.

Hu, E. J.; Shen, Y.; Wallis, P.; Allen-Zhu, Z.; Li, Y.; Wang, S.; Wang, L.; and Chen, W. 2021. Lora: Low-rank adaptation of large language models. *arXiv preprint arXiv:2106.09685*.

Jiang, J.; Zhou, K.; Dong, Z.; Ye, K.; Zhao, W. X.; and Wen, J.-R. 2023. Structgpt: A general framework for large language model to reason over structured data. *arXiv preprint arXiv:2305.09645*.

Jiang, J.; Zhou, K.; Zhao, W. X.; and Wen, J.-R. 2022. Unikgqa: Unified retrieval and reasoning for solving multi-hop question answering over knowledge graph. *arXiv preprint arXiv:2212.00959*.

Koller, D.; and Friedman, N. 2009. *Probabilistic graphical models: principles and techniques*. MIT press.

Lan, Y.; He, G.; Jiang, J.; Jiang, J.; Zhao, W. X.; and Wen, J.-R. 2022. Complex knowledge base question answering: A survey. *IEEE Transactions on Knowledge and Data Engineering*, 35(11): 11196–11215.

Liu, X.; Ji, K.; Fu, Y.; Tam, W. L.; Du, Z.; Yang, Z.; and Tang, J. 2021. P-tuning v2: Prompt tuning can be comparable to fine-tuning universally across scales and tasks. *arXiv preprint arXiv:2110.07602*.

Long, X.; Zhuang, L.; Aodi, L.; Wang, S.; and Li, H. 2022. Neural-based mixture probabilistic query embedding for answering fol queries on knowledge graphs. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, 3001–3013.

Long, X.; Zhuang, L.; Li, A.; Li, H.; and Wang, S. 2024a. Fact Embedding through Diffusion Model for Knowledge Graph Completion. In *Proceedings of the ACM on Web Conference 2024*, 2020–2029.

Long, X.; Zhuang, L.; Li, A.; Wei, J.; Li, H.; and Wang, S. 2024b. KGDM: A Diffusion Model to Capture Multiple Relation Semantics for Knowledge Graph Embedding. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 38, 8850–8858.

Luo, L.; Li, Y.-F.; Haffari, G.; and Pan, S. 2023. Reasoning on graphs: Faithful and interpretable large language model reasoning. *arXiv preprint arXiv:2310.01061*.

Miller, A.; Fisch, A.; Dodge, J.; Karimi, A.-H.; Bordes, A.; and Weston, J. 2016. Key-value memory networks for directly reading documents. *arXiv preprint arXiv:1606.03126*.

OpenAI, R. 2023. Gpt-4 technical report. arxiv 2303.08774. *View in Article*, 2(5).

Pearl, J. 2009. *Causality*. Cambridge university press.

Saxena, A.; Kochsiek, A.; and Gemulla, R. 2022. Sequence-to-sequence knowledge graph completion and question answering. *arXiv preprint arXiv:2203.10321*.

Saxena, A.; Tripathi, A.; and Talukdar, P. 2020. 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*, 4498–4507.

Shi, J.; Cao, S.; Hou, L.; Li, J.; and Zhang, H. 2021. Transfernet: An effective and transparent framework for multi-hop question answering over relation graph. *arXiv preprint arXiv:2104.07302*.

Sun, H.; Bedrax-Weiss, T.; and Cohen, W. W. 2019. Pullnet: Open domain question answering with iterative retrieval on knowledge bases and text. *arXiv preprint arXiv:1904.09537*.

Sun, H.; Dhingra, B.; Zaheer, M.; Mazaitis, K.; Salakhutdinov, R.; and Cohen, W. W. 2018. Open domain question answering using early fusion of knowledge bases and text. *arXiv preprint arXiv:1809.00782*.

Sun, J.; Xu, C.; Tang, L.; Wang, S.; Lin, C.; Gong, Y.; Shum, H.-Y.; and Guo, J. 2023. Think-on-graph: Deep and responsible reasoning of large language model with knowledge graph. *arXiv preprint arXiv:2307.07697*.

Sun, Z.; Deng, Z.-H.; Nie, J.-Y.; and Tang, J. 2019. Rotate: Knowledge graph embedding by relational rotation in complex space. *arXiv preprint arXiv:1902.10197*.

Talmor, A.; and Berant, J. 2018. The web as a knowledge-base for answering complex questions. *arXiv preprint arXiv:1803.06643*.

Touvron, H.; Martin, L.; Stone, K.; Albert, P.; Almahairi, A.; Babaei, Y.; Bashlykov, N.; Batra, S.; Bhargava, P.; Bhosale, S.; et al. 2023. Llama 2: Open foundation and fine-tuned chat models. *arXiv preprint arXiv:2307.09288*.

Vrandečić, D.; and Krötzsch, M. 2014. Wikidata: a free collaborative knowledgebase. *Communications of the ACM*, 57(10): 78–85.

Wang, K.; Duan, F.; Wang, S.; Li, P.; Xian, Y.; Yin, C.; Rong, W.; and Xiong, Z. 2023. Knowledge-driven cot: Exploring faithful reasoning in llms for knowledge-intensive question answering. *arXiv preprint arXiv:2308.13259*.

Wei, J.; Wang, X.; Schuurmans, D.; Bosma, M.; Xia, F.; Chi, E.; Le, Q. V.; Zhou, D.; et al. 2022. Chain-of-thought prompting elicits reasoning in large language models. *Advances in neural information processing systems*, 35: 24824–24837.

Wu, Y.; Hu, N.; Qi, G.; Bi, S.; Ren, J.; Xie, A.; and Song, W. 2023. Retrieve-rewrite-answer: A kg-to-text enhanced llms framework for knowledge graph question answering. *arXiv preprint arXiv:2309.11206*.

Ye, X.; Yavuz, S.; Hashimoto, K.; Zhou, Y.; and Xiong, C. 2021. Rng-kbqa: Generation augmented iterative ranking for knowledge base question answering. *arXiv preprint arXiv:2109.08678*.

Yih, W.-t.; Richardson, M.; Meek, C.; Chang, M.-W.; and Suh, J. 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)*, 201–206.

Zeng, A.; Liu, X.; Du, Z.; Wang, Z.; Lai, H.; Ding, M.; Yang, Z.; Xu, Y.; Zheng, W.; Xia, X.; et al. 2022. Glm-130b: An open bilingual pre-trained model. *arXiv preprint arXiv:2210.02414*.

Zhang, J.; Zhang, X.; Yu, J.; Tang, J.; Tang, J.; Li, C.; and Chen, H. 2022. Subgraph retrieval enhanced model for multi-hop knowledge base question answering. *arXiv preprint arXiv:2202.13296*.

Zhang, L.; Zhang, J.; Wang, Y.; Cao, S.; Huang, X.; Li, C.; Chen, H.; and Li, J. 2023. FC-KBQA: A fine-to-coarse composition framework for knowledge base question answering. *arXiv preprint arXiv:2306.14722*.