G-HiRel: Enhancing the Adaption to Knowledge Updating for Large Language Model Reasoning

Anonymous ACL submission

Abstract

Large language models (LLMs) have achieved good performance in multiple reasoning tasks. However, they are limited to adapt the rapid knowledge updates in the real-world scenario without retraining the entire LLM or modifying the model weights. Excluding these consuming methods, knowledge graphs (KGs) are used as external memory under knowledge updating because of their structural knowledge and efficient updating ability, which is yet limited by the gap between structural KG and LLM, and the deficient entity-independent semantics. To the end, we propose an LLM reasoning framework with hierarchical relational retrieval for large-scale knowledge updating, named G-HiRel. To integrate the structural edited KG into continuous LLMs, G-HiRel generates hierarchical instructions based on natural language questions. In order to handle the knowledge inconsistency between the KG and LLM and obtain the entity independence, G-HiRel utilizes a designed hierarchical relational retrieval for relational path candidates, which are selected by a designed semantics-based strategy. Finally, top entity-independent relational paths are instantiated and integrated into LLMs to generate the answer, in order to verify the reasoning performance under knowledge edits. Extensive experiments of G-HiRel on three benchmarks show that G-HiRel achieves superiority in terms of accuracy and interpretability.

1 Introduction

Large language models (LLMs) (Achiam et al., 2023; Thoppilan et al., 2022; Chowdhery et al., 2023) have obtained outstanding performance in multiple tasks, such as information extraction (Ma et al., 2023; Qi et al., 2024), information retrieval (Tang et al., 2024; Feng et al., 2024), question answering and reasoning (Monteiro et al., 2024; Dong et al., 2024), etc. However, LLMs are still limited by their static knowledge and struggle with real-world knowledge updating (Scialom et al., 2022; Luo et al., 2023), especially in the reasoning task. The rapid and large-scale knowledge editing (Wang et al., 2024a,c) will cause model parameters to be updated, which is consuming and lasks scalability.

Therefore, although some approaches by in-context learning (Wang et al., 2024b) are proposed to inject knowledge into LLMs without modifying parameters, they still have difficulties in solving questions involving complex reasoning process and massive background knowledge updates, which is denoted as LLM reasoning under knowledge editing (Zhong et al., 2023). For example, in Fig. 1 (a), after updating the knowledge of *Sikhism* (i.e. *Sikhism was founded by Alexander I of Russia*), the LLM still generates a wrong answer, even it is prompted to generate the answer based on this edited knowledge.

In order to avoid model retraining or fine-tuning, and handle massive knowledge edits in LLM reasoning, some approaches (Zhong et al., 2023; Gu et al., 2024; Chen et al., 2024b) attempt to store knowledge in external memory. Among these methods, knowledge graphs (KGs), like Wikidata (Vrandečić and Krötzsch, 2014), contain a large scale of knowledge and are dynamically updated with new factual triples. Moreover, compared to the natural language text, KGs represent knowledge in a structural triple form $(\text{sub}, rel, \text{obj})$, which is more refined and precise without redundancy.

Despite the ability of KGs in managing updating knowledge, introducing knowledge from KGs into the LLM still suffers from two main difficulties: **1) Structural KG integration into continuous language models.** KGs are constructed by structural triples, which represent the relations between two entities, e.g. (Sikhism, $foundedBy$, Alexander I of Russia) in Fig. 1 (b), but knowledge in LLMs is continuously stored, which gives rise to the gap of semantic. It restricts the retrieval, editing, and reasoning of LLMs, where the massive structural information in KGs is required to be introduced

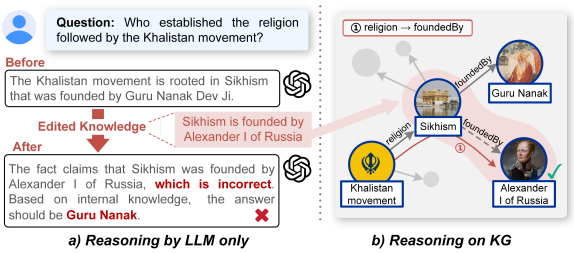

Figure 1: An example of complex multi-hop reasoning after knowledge editing with two paradigms: (a) Reasoning by LLM-only (i.e., the LLM used is OpenAI-o3) and (b) Reasoning with edited KG.

into continuous knowledge in LLMs. **2) Deficient entity-independent semantics.** There exists an inconsistency between knowledge in LLM and structural knowledge of KGs after the editing process (Xu et al., 2024). In previous knowledge-based question answering (KBQA) reasoning methods (Sun et al., 2024; Luo et al., 2024a), the high dependence on KG entities increases the harm of knowledge inconsistency. Therefore, mining the entity-independent semantics (Teru et al., 2020) in KGs benefits the LLM reasoning under knowledge editing scenario.

To address the above challenges in LLM reasoning after knowledge updating, we propose a **KG**-based reasoning framework by **Hi**erarchical **Rel**ational retrieval named **G-HiRel**. To obtain entity independence and avoid inconsistency between KG and LLM after knowledge editing, G-HiRel focuses on the fixed reasoning patterns consisting of relational paths rather than intermediate entities. We firstly design a hierarchical strategy to bridge the semantic gap between the structural knowledge and natural language text by relational paths. The hierarchical relational retrieval constructs the edited subgraph skeleton after large-scale editing. Then, we design a multi-layer and multi-step retrieval for an edited subgraph that contains multiple relational paths in the KG. In each step of expansion, G-HiRel selects the top relational paths based on their semantics, which helps reduce the negative impact of knowledge updating. G-HiRel finally guides the LLM to instantiate the selected top relational paths as reasoning paths and integrates the edited knowledge into the LLM. The complex multi-hop reasoning task evaluates the ability of G-HiRel in adapting large-scale knowledge edits.

Our main contributions are as following:

- We innovatively propose an LLM reasoning framework adapting large-scale knowledge editing, named G-HiRel. To the best of our knowledge, G-HiRel is the first method utilizing hierarchical relational retrieval to answer complex multi-hop questions under knowledge editing scenario.
- A hierarchical instruction strategy is proposed to bridge the gap between structural information from KG and LLM for reasoning under knowledge editing. As for the deficient entity-independent semantics, we focus on relational paths and implement a multi-layer and multi-step expansion with the hierarchical instructions. We also propose a semantics-based strategy to precisely select the top relational paths as the skeletons of the edited subgraph.
- Extensive experiments on multi-hop question answering datasets with editing facts show that G-HiRel provides the ability in adapting large-scale knowledge edits. G-HiRel achieves superior results compared to previous SOTA methods in terms of reasoning accuracy and interpretability.

2 Related Work

Knowledge Editing. The knowledge editing task on LLMs aims to edit and update the knowledge into language model. Recent knowledge editing methods (Wang et al., 2024b) can be divided into three types: *(1) Global optimization methods* modify the internal knowledge by updating model parameters. For example, KGEditor (Cheng et al., 2024a) uses a hyper-network to generate new parameters, and F-Learning (Ni et al., 2024) applies the parameter arithmetic to forget and relearn knowledge. *(2) Local optimization methods* focus on updating a small set of relevant model weights for efficient knowledge editing. MELO (Yu et al., 2024) and PMTE (Li et al., 2024) also selectively adjust parts of the model parameters to edit knowledge. AlphaEdit (Fang et al., 2024) projects the parameter perturbations from each edit into the knowledge-retention matrix and enable multi-round editing. *(3) External memorization methods* avoid modifying parameters of the model. This category is a different task to verify if the language model can adapt the knowledge editing. Representation methods include MEMPROMPT (Madaan et al., 2022) and IKE (Zheng et al., 2023), which modify outputs using in-context prompts. RAE (Shi et al., 2024a) maximizes mutual infor-

mation to extract fact chains and uses uncertainty-based pruning for accurate multi-hop reasoning. Mello (Zhong et al., 2023) stores external text and incrementally solves complex problems using updated facts, while PokeMQA (Gu et al., 2024) introduces a programmable scope detector and knowledge prompt generator to guide retrieval without parameter changes. GMello (Chen et al., 2024b) uses a KG as external memory to answer complex questions. Although following paradigm using external KG, G-HiRel design a hierarchical strategy to integrate KG into LLM adapting knowledge editing and obtain more accurate retrieval for LLM reasoning.

KG-based LLM Reasoning. LLM-based reasoning methods have already achieved significant progress in optimizing the retrieval and interaction process (Glass et al., 2022; Shao et al., 2023; Cheng et al., 2024b). Since the structure of KGs helps enhance the reasoning ability of LLMs (Luo et al., 2024b), researchers have further explored KGs to improve reasoning performance. Several studies reduce noise in KG retrieval to improve reasoning by pruning subgraphs. Especially, ToG (Sun et al., 2024) utilizes external KGs to improve the multi-hop reasoning performance by retrieval of reasoning paths on graph. Some studies enhance coverage through subgraph expansion. KG^{2}RAG (Zhu et al., 2025) uses semantic search to obtain seed blocks and performs multi-hop retrieval via graph structure by a maximum spanning tree. Other methods directly optimize the retrieval-and-reasoning pipeline. PoG (Chen et al., 2024a) introduces an adaptive Retrieve-Reflect-Revise planner that iteratively extends and backtracks along reasoning paths for accurate reasoning. Despite these reasoning methods, the knowledge inconsistency is still an issue for large-scale knowledge editing scenario. Inspired by the works of retrieval-augmented generation (RAG), we design a hierarchical relational retrieval, which provides entity-independent semantics during reasoning.

3 Preliminary

3.1 Task Definition

In the knowledge editing for LLM, a constructed KG consisting of triples ($\mathcal{K} \subseteq \mathcal{E} \times \mathcal{R} \times \mathcal{E}$) is utilized for knowledge updating, where $\mathcal{E}$ is the set of entities and $\mathcal{R}$ is the set of relations connecting two entities. The knowledge editing process can be represented as $(e_h, r, e_t) \rightarrow (e_h, r, e_t') \in \mathcal{K}$, where $e_h, e_t, e_t' \in \mathcal{E}$, $r \in \mathcal{R}$. Specifically, e_h denotes the head entity of the triple, r is the relation, while e_t and e_t' are the original and updated tail entities respectively. Complex multi-hop reasoning is to evaluate the effectiveness of knowledge editing on LLM, which is required to output the answer under editing without updating the parameters of LLMs.

3.2 Relational Path and Reasoning Path

Reasoning Path. In the multi-hop reasoning, the reasoning path $e_s \xrightarrow{r_1} e_1 \xrightarrow{r_2} e_2 \cdots \xrightarrow{r_n} e_n$ consisting of entities and the relations provides an explicit reasoning process. For example, in Fig. 1, (Khalistan movement $\xrightarrow{religion}$ Sikhism $\xrightarrow{foundedBy}$ Alexander I) is a reasoning path to answer the complex multi-hop question.

Relational Path. A relational path is often considered as the skeleton of the reasoning path. It is a path consisting of adjacent relations from the start entity e_s, which is denoted as $r_1 \rightarrow r_2 \rightarrow \cdots \rightarrow r_n$, in which $r_1, r_2, \cdots, r_n \in \mathcal{R}$. The relational path does not contain entities, which possesses an entity independence in KG. For example, in Fig. 1, $(religion \rightarrow foundedBy)$ is a relational path. Under the editing scenario, the relational paths in KG are consistent during reasoning.

4 Methodology

4.1 KG Extraction and Editing

As shown in Fig. 2, we firstly employ the LLM to extract the KG $\mathcal{G}$ as a repository of editing. It should be noted that this operation is based on an assumption that the answer to evaluate the editing is in this extracted KG (Mavromatis and Karypis, 2024). Specifically, in the knowledge editing scenario, we apply the updated triples $(e_h, r_m, e_t) \rightarrow (e_h, r_m, e_t')$ to edit $\mathcal{G}$. In detail, if there is a triple in $\mathcal{G}$ with the same head entity e_h and relation r_m, we update its tail entity e_t with e_t'. Otherwise, we add a new triple (e_h, r_m, e_t') to $\mathcal{G}$ for editing. Then, we analyze each multi-hop question and select corresponding top relations $R \subseteq \mathcal{R}$ according to their frequency of appearances to reduce the complexity by LLM. The KG $\mathcal{G}$ is extracted as the repository for editing, which will preserve the knowledge consistency. In addition, given the original question $q \in \mathcal{Q}$, we use LLM with prompt in Appendix E.4 to identify the start entity e_s.

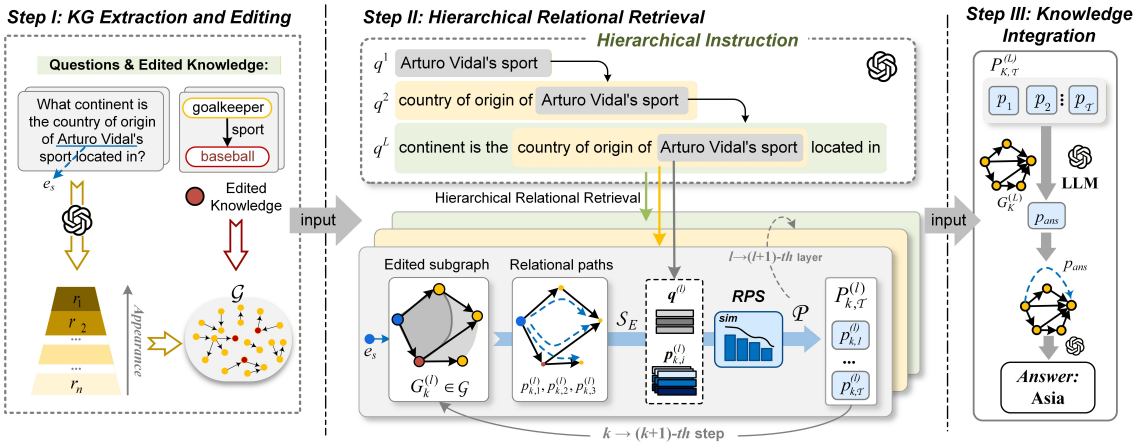

Figure 2: Overview of G-HiRel. G-HiRel firstly extracts a KG and injects edited knowledge into it. Secondly, G-HiRel retrieves the relational paths with the hierarchical instructions, which construct edited subgraph by a multi-layer and multi-step expansion. Finally, G-HiRel guides the LLM to integrate knowledge in edited subgraph, and generate an answer.

4.2 Hierarchical Relational Retrieval

4.2.1 Hierarchical Instruction

In this process, the complex question is transferred to a sequence of hierarchical instructions to match the structural topology of $\mathcal{G}$. The LLM iteratively generates hierarchical instruction $q^{(l)}$ on l-th layer based on the relational predicate information of q, starting from e_s and gradually rebuilding the full question with L layers, until $q^{(L)}$ is textually the same as q. Each $q^{(l+1)}$ inherits the complete textual content of $q^{(l)}$ and extends $q^{(l)}$, which is denoted as $q^{(l)} \sqsubseteq q^{(l+1)}$. These L instructions construct a hierarchical instruction set $S_q = \{q^1, q^2, \cdots, q^{(l)}, \cdots, q^{(L)}\}$, which is shown in the gray, yellow and green box of Fig. 2. Each item in S_q instructs each layer of subgraph consisting of relational paths. Specifically, in Fig. 2, the original question *'What continent is the country of origin of Arturo Vidal's sport located in?'* involves three relation predicates: '*sport*', '*country of origin*', and '*continent*', which generates hierarchical instructions as q^1, q^2, q^3, respectively.

4.2.2 Multi-layer and Multi-step Expansion

We apply an edited subgraph to store the updated knowledge and extracted relational paths. As illustrated in Section 4.2.1, the hierarchical instructions provide guidance for the L-layer subgraph expansion. In every layer, there will be up to K steps, adding more relations to expand the subgraph. The subgraph in the l-th layer and k-th expansion is denoted as $G_k^{(l)} \in \mathcal{G}$, where $0 \leq k \leq K$. We use the start entity e_s to initialize the edited subgraph, which is denoted as $G_0^{(0)}$. In order to obtain entity independence under knowledge editing, a relational path is constructed under the guidance of the l-th hierarchical instruction $q^{(l)}$.

Each layer reserves the top-$\mathcal{T}$ relational paths, formally defined as $P_{k,\mathcal{T}}^{(l)} = \{p_{k,1}^{(l)}, p_{k,2}^{(l)}, \cdots, p_{k,\mathcal{T}}^{(l)}\}$, where each $p_{k,i}^{(l)}$ represents the i-th relational path after k steps of expansion in the l-th layer. Specifically, the relational path $p_{k,i}^{(l)}$ is denoted as:

$$p_{k,i}^{(l)} = (\overbrace{r_{1,i}^{(1)} \rightarrow r_{2,i}^{(1)} \cdots \rightarrow r_{K',i}^{(1)}}^{1st \text{ layer}} \cdots \overbrace{r_{1,i}^{(l)} \rightarrow r_{2,i}^{(l)} \cdots \rightarrow r_{k,i}^{(l)}}^{l\text{-th layer}}). \quad (1)$$

where $K' \leq K$. Therefore, even with edited knowledge, the relational paths instructed by S_q will not be affected by the knowledge inconsistency and available to preserve the knowledge consistency of KG and LLM. After k steps of expansion in the l-th layer, we employ a discriminator $\mathcal{P}$ to determine whether the relational paths from $P_{k,\mathcal{T}}^{(l)}$ are adequate for the current $q^{(l)}$, whose prompt is in Appendix E.2. If they are, the relational paths will transfer to $(l+1)$-th layer instructed by $q^{(l+1)}$ until the $q^{(L)} \in S_q$ has been solved; otherwise it continues the $(k+1)$-th step of expansion.

We retrieve the passing entities of relational paths in $P_{k,\mathcal{T}}^{(l)}$ by $\mathcal{G}$ and extract the subgraph, which is actually $G_k^{(l)}$. Specifically, candidate paths are derived from the one-hop neighbors of the tail entities in the previous step. Note that although the entities are edited, we still focus on the relational paths in $G_k^{(l)}$, which are used for knowledge preservation. Let $p_{k,i}^{(l)}$ be a relational path in $G_k^{(l)}$ and i indicate the i-th path, the expansion is to explore each rela-

tional path $p_{k,i}^{(l)}$ by retrieving relations from the tail entities $E_{k,i}^{(l)}$ in $\mathcal{G}$. Specifically, for each $e \in E_{k,i}^{(l)}$, we collect all the relations in its neighborhoods and add them to the sequence of relations in $p_{k,i}^{(l)}$ to make the $k-1$-th to k-th step of expansion, which is denoted as:

$$p_{k,j}^{(l)} = (p_{k-1,i}^{(l)} \to r_{k,j}^{(l)}), r_{k,j}^{(l)} \in \mathcal{N}_r(r_{k-1,i}^{(l)}), \quad (2)$$

$$\mathcal{N}_r(r_{k,i}^{(l)}) = \{r \mid \forall r \in \mathcal{N}(e), e \in E_{k,i}^{(l)}\}, \quad (3)$$

$$E_{k,i}^{(l)} = \{e \mid (e_s, p_{k,i}^{(l)}, e) \in \mathcal{G}\} = \{e \mid (e_s, r_{1,i}^{(l)}, \cdots, r_{k,i}^{(l)}, e) \in \mathcal{G}\} \quad (4)$$

where $E_{k,i}^{(l)}$ contains all entities in $\mathcal{G}$ that can be reached through $p_{k,i}^{(l)}$. $\mathcal{N}(\cdot)$ refers to the neighbor relations of an entity. After the k-th expansion in the (l)-th level, a set of candidate relational paths is obtained, which can be denoted as:

$$\overline{P}_{k,i}^{(l)} = \{(p_{k-1,i}^{(l)} \to r_{k,j}^{(l)}) \mid \forall r_{k,j}^{(l)} \in \mathcal{N}_r(r_{k-1,i}^{(l)})\}, \quad (5)$$

$$\overline{P}_{k}^{(l)} = \{\overline{P}_{k,1}^{(l)}, \cdots, \overline{P}_{k,i}^{(l)}, \cdots, \overline{P}_{k,\mathcal{T}}^{(l)}\} \quad (6)$$

For example, as shown in Fig. 2, given a relational path $p_{1,1}^{(1)}$ = *(position of team)*, we will obtain a i-th candidate paths set after a one-step expansion: $\overline{P}_{2,1}^{(1)}$ = { *(position of team, sport), (position of team, location)*}.

4.2.3 Relational Path Selection (RPS)

Obtaining candidate relational paths set $\overline{P}_{k}^{(l)}$ after k-step exploration in (l)-th layer, we select top-$\mathcal{T}$ relational paths in the candidates to derive the edited subgraph $G_k^{(l)}$ for the next layer. To relieve the influence of the knowledge updating of entities in $\mathcal{G}$, we efficiently select the top-$\mathcal{T}$ relational paths by the relational semantics.

In order to implement hierarchical relational retrieval, we align S_q with the relational paths in each step. The textual hierarchical instruction $q^{(l)}$ and each $p_{k,i}^{(l)} \in P_k^{(l)}$ are encoded into vectors $\boldsymbol{q}^{(l)} \in \mathbb{R}^m$ and $\boldsymbol{p}_{k,i}^{(l)} \in \mathbb{R}^m$ by $\mathcal{S}_E$, respectively, where m is the dimension of the vector. The encoding process is defined as $\boldsymbol{q}^{(l)} = \mathcal{S}_E(q^{(l)})$ and $\boldsymbol{p}_{k,i}^{(l)} = \mathcal{S}_E(p_{k,i}^{(l)})$. We then compute the semantic similarity between $\boldsymbol{q}^{(l)}$ and $\boldsymbol{p}_{k,i}^{(l)}$ to select paths that match the semantics of the hierarchical instruction:

$$P_{k,\mathcal{T}}^{(l)} = \left\{p_t = \underset{p_{k,i}^{(l)} \in P_k^{(l)}}{\arg\max} \; score(\boldsymbol{p}_{k,i}^{(l)}, \boldsymbol{q}^{(l)}) \mid t \in \{1 \cdots \mathcal{T}\}\right\}. \quad (7)$$

We obtain the top-$\mathcal{T}$ relational paths $P_{k,\mathcal{T}}^{(l)}$ ranked by the cosine similarity. $G_k^{(l)}$ is formed by the paths in $P_{k,\mathcal{T}}^{(l)}$ and the entities passed by these paths, which contains the edited entities.

4.3 Knowledge Integration

The effectiveness of methods handling knowledge editing is evaluated by multi-hop complex reasoning (Fang et al., 2025). We have constructed a subgraph $G_K^{(L)}$ and then integrate the edited knowledge into an LLM $\mathcal{M}$ to derive the final answer.

A few-shot prompt guides $\mathcal{M}$ to select the most relevant path from $P_{K,\mathcal{T}}^{(L)}$ based on the question as the final relational path p_r (Appendix E.6). In order to integrate the edited knowledge in the subgraph, we need to instantiate the relational path into reasoning paths with edited entities. p_r is instantiated on $G_K^{(L)}$ to generate a candidate set of reasoning paths P_{cand} containing edited knowledge. Then, a most suitable reasoning path p_{ans} based on the question and edited knowledge is selected from P_{cand} by $\mathcal{M}$, whose prompt is in Appendix E.6. This process integrates the structural edited knowledge into the LLM $\mathcal{M}$, whose generated answers are utilized to verify the effectiveness of editing. All the notations of G-HiRel are in Appendix B.

5 Experimental Results

We will answer the following questions to illustrate the results: **RQ1:** Does G-HiRel outperform other methods? **RQ2:** How to select reasoning LLM of G-HiRel in terms of time and cost? **RQ3:** How effective is the hierarchical relational retrieval? **RQ4:** How effective is the entity-independent semantics? **RQ5:** How do different factors influence the performance of G-HiRel?

5.1 Experiment Settings

Datasets. We comprehensively evaluate the effectiveness of our method on MQuAKE-CF-3k and MQuAKE-T datasets (Zhong et al., 2023), which contain counterfactual edits and temporal knowledge updates, respectively. These datasets consist of multi-hop questions, where each multi-hop question involves one or more editable facts. As mentioned in previous multi-hop $(2, 3, 4)$ question answering methods (Chen et al., 2024b; Sun et al., 2024), we adopt Wikidata (Vrandečić and Krötzsch, 2014) as the KG to construct the edited subgraph in G-HiRel. The edited knowledge follows the settings in MQuAKE (Zhong et al., 2023). We also implement experiments on one-hop edited dataset WikiUpdate (Wu et al., 2024). The detailed statistics of the datasets are in Appendix C.2.

Baselines. To comprehensively evaluate G-HiRel, we compare it with typical baselines. We se-

Table 1: Comparison of reasoning performance (%) on MQuAKE-CF-3k and MQuAKE-T. The optimal and suboptimal results are marked in blue and red, respectively. ♣ and ⋄ denote results from (Chen et al., 2024b) and (Gu et al., 2024). Other baselines are reproduced in the same experimental environment.

Method	$\mathcal{M}$	MQuAKE-CF-3k						MQuAKE-T					
		edit=1		edit=100		edit=3000		edit=1		edit=100		edit=1868	
		Acc↑	Hop-Acc↑	Acc↑	Hop-Acc↑	Acc↑	Hop-Acc↑	Acc↑	Hop-Acc↑	Acc↑	Hop-Acc↑	Acc↑	Hop-Acc↑
MEMIT♣	GPT-J-6B	12.30	–	9.80	–	1.80	–	4.80	–	1.00	–	0.00	–
MEND♣	GPT-J-6B	11.50	–	9.10	–	3.50	–	38.20	–	17.40	–	4.60	–
AlphaEdit	GPT-J-6B	3.77	–	3.60	–	1.80	–	1.34	–	1.34	–	0.96	–
MeLLo♣	GPT-J-6B	20.30	–	12.50	–	9.80	–	85.90	–	45.70	–	30.70	–
GMello♣	GPT-J-6B	76.30	–	53.40	–	49.00	–	86.90	–	82.10	–	81.50	–
G-HiRel (Ours)	GPT-J-6B	51.87	40.97	48.90	39.53	42.57	35.27	88.87	81.85	89.67	81.85	89.61	81.96
MeLLo ⋄	Vicuna-7B	20.70	7.03	12.83	6.77	10.90	6.70	84.40	–	56.30	–	51.30	–
GMello ♣	Vicuna-7B	71.30	–	46.50	–	41.90	–	97.10	–	86.30	–	85.10	–
PokeMQA ⋄	Vicuna-7B	45.83	34.80	38.77	31.23	31.63	25.30	74.57	55.19	–	–	73.07	55.09
RAE	Vicuna-7B	61.57	–	52.53	–	41.93	–	75.27	–	76.61	–	75.96	–
FastToG	Vicuna-7B	10.03	–	9.53	–	9.50	–	17.40	–	16.40	–	17.08	–
G-HiRel (Ours)	Vicuna-7B	63.33	49.93	57.07	46.80	48.73	41.00	95.13	89.56	94.70	88.81	94.38	88.60
Mello	DeepSeek-7B	56.70	13.70	32.50	12.20	25.33	11.27	92.08	2.78	63.22	6.26	53.14	7.17
PokeMQA	DeepSeek-7B	53.30	35.20	45.47	35.17	38.03	28.63	77.94	65.04	75.86	64.45	75.80	64.40
G-HiRel (Ours)	DeepSeek-7B	64.30	48.87	57.50	45.57	49.23	40.10	92.56	81.80	92.72	81.75	91.22	81.00
CoT	GPT-4o-mini	3.93	–	3.93	–	3.93	–	65.95	–	65.95	–	65.95	
ToG	GPT-4o-mini	17.17	–	14.50	–	12.00	–	68.63	–	68.20	–	69.27	–
KEDKG	GPT-4o-mini	48.13	–	44.70	–	38.03	–	88.38	–	86.78	–	86.03	–
G-HiRel (Ours)	GPT-4o-mini	72.37	57.07	63.97	52.30	53.03	44.27	96.20	90.10	95.82	89.51	95.56	89.03

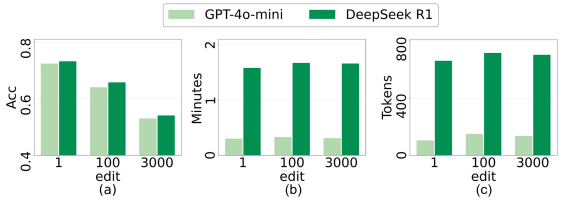

Figure 3: Reasoning performance on the MQuAKE-CF-3k. Consumption of Time (min) and Tokens per question are calculated.

lect MEMIT (Meng et al., 2023), MEND (Mitchell et al., 2022), and AlphaEdit (Fang et al., 2025) integrating edits by fine-tuning the language models. Other methods like Mello (Zhong et al., 2023), PokeMQA (Gu et al., 2024), GMello (Chen et al., 2024b), RAE (Shi et al., 2024b) and KEDKG (Lu et al., 2025) utilize externally stored knowledge to support multi-hop reasoning during knowledge editing. The comparison also contains CoT (Wei et al., 2022), ToG (Sun et al., 2024) and the most recent LLM reasoning method FastToG (Liang and Gu, 2025) in the knowledge editing scenario. More Details of baselines are in Appendix C.1.

Metrics. During the experiments, we calculate the **Accuracy (Acc)** of the language models in answering multi-hop questions (Zhong et al., 2023) after knowledge editing. We additionally utilize **Hop-wise Answering Accuracy (Hop-Acc)** (Gu et al., 2024) to evaluate the interpretability to see if the questions have been solved by the LLM only when the derived reasoning path exactly matches the provided one.

Implementation Details. Following MQuAKE (Zhong et al., 2023), the editions are processed in group of a times, with $a \in \{1, 100, 3000\}$ for MQuAKE-CF-3k and $a \in \{1, 100, 1868\}$ for MQuAKE-T. For $\mathcal{P}$ and $\mathcal{S}_E$, we choose Flan-T5-Large (Chung et al., 2024) and MiniLM-L6-v3 (Reimers and Gurevych, 2019) respectively. In expansion, we use $\mathcal{T} = 3$ in implementation. As for the reasoning LLM $\mathcal{M}$, we evaluate our methods on smaller LLMs, such as GPT-J-6B (Wang and Komatsuzaki, 2021), Vicuna-7B (Chiang et al., 2023), and DeepSeek-7B (Bi et al., 2024) for fair comparison. In order to explore SOTA reasoning results, we utilize black-box GPT-4o-mini[1] and Deepseek-R1 (Guo et al., 2025) as $\mathcal{M}$.

5.2 Main Results

5.2.1 Comparison with baselines (RQ1)

In this section, we compare the LLM reasoning results under knowledge editing with recent baselines, which are shown in Table 1. Compared to methods without external memory (gray block in Table 1), G-HiRel achieves the best reasoning performance. Specifically, G-HiRel achieves a significant improvement in Acc, with gains of no less than 39.10%. The results indicate that $\mathcal{M}$ struggles to perform effective knowledge editing without external tools.

[1]https://platform.openai.com/docs/models/gpt-4o-mini

Table 2: Ablation accuracy results (%) of G-HiRel on MQuAKE-CF-3k and MQuAKE-T.

Method	Scenario				MQuAKE-CF-3k			MQuAKE-T		
	HI	ML	MS	RPS	edit=1	edit=100	edit=3000	edit=1	edit=100	edit=1868
G-HiRel	✓	✓	✓	✓	**72.37**	**63.97**	**53.03**	**96.20**	**95.82**	**95.56**
G-HiRel$_{\text{w/o HI}}$	×	✓	✓	✓	61.73$_{\downarrow 11.64}$	55.40$_{\downarrow 8.57}$	46.87$_{\downarrow 6.16}$	77.09$_{\downarrow 19.11}$	76.98$_{\downarrow 18.84}$	76.87$_{\downarrow 18.69}$
G-HiRel$_{\text{w/o RPS}}$	✓	✓	✓	×	24.47$_{\downarrow 47.90}$	25.03$_{\downarrow 38.94}$	23.33$_{\downarrow 29.70}$	55.46$_{\downarrow 40.74}$	56.32$_{\downarrow 39.50}$	57.76$_{\downarrow 37.80}$
G-HiRel$_{\text{w/o ML}}$	✓	×	✓	✓	64.63$_{\downarrow 7.74}$	57.40$_{\downarrow 6.57}$	49.40$_{\downarrow 3.63}$	93.47$_{\downarrow 2.73}$	93.42$_{\downarrow 2.40}$	92.99$_{\downarrow 2.57}$
G-HiRel$_{\text{w/o MS}}$	✓	✓	×	✓	35.37$_{\downarrow 37.00}$	34.77$_{\downarrow 29.20}$	33.03$_{\downarrow 20.00}$	81.96$_{\downarrow 14.24}$	81.96$_{\downarrow 13.86}$	81.10$_{\downarrow 14.46}$

Table 3: Total time cost (hours) of G-HiRel with different LLMs on MQuAKE-CF-3k and MQuAKE-T.

Dataset	GPT-J-6B	Vicuna-7B	DeepSeek-7B	GPT-4o-mini
MQuAKE-CF-3k	113.98 h	42.61 h	71.87 h	47.38 h
MQuAKE-T	135.46 h	49.10 h	43.84 h	61.39 h

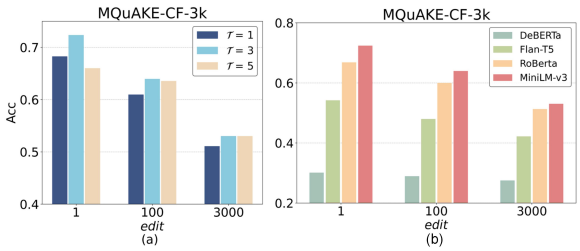

Figure 4: (a) Effectiveness of $\mathcal{T}$; (b) Effectiveness of $\mathcal{S}_E$. $\mathcal{M}$ is GPT-4o-mini.

Compared to methods that leverage external knowledge, G-HiRel achieves either the optimal or suboptimal reasoning performance across all settings. For the Hop-Acc metric, G-HiRel surpasses all baseline methods under 6 conditions, indicating that it is more capable of capturing the correct reasoning path for interpretability in knowledge editing scenarios. Compared to the most recent baseline approaches (e.g. Mello and PokeMQA), G-HiRel achieves average gains of at least 14.55% in Acc with various hops (Fig. 7). In large-scale editing settings (edit = 100, 1868, 3000), G-HiRel also maintains a clear advantage over baselines. As shown in Table 1, G-HiRel outperforms the most recent LLM reasoning methods as well, using GPT-4o-mini as $\mathcal{M}$ for a fair comparison. These results indicate the superiority of G-HiRel in both performance and interpretability. Additional results on WikiUpdate (Wu et al., 2024) are provided in Appendix D.4.

5.2.2 Performance on Different LLMs (RQ2)

We also compare the performance of G-HiRel on different reasoning LLMs. As shown in Table 1, G-HiRel achieves better reasoning performance when using Vicuna-7B and DeepSeek-7B as $\mathcal{M}$ than with GPT-J-6B. The reason might be that GPT-J-6B is less effective than other LLMs in selecting the correct relational path of G-HiRel. As for the black-box LLMs, the results of GPT-4o-mini and DeepSeek-R1 as $\mathcal{M}$ are shown in Table 1 and Fig. 3. Both models achieve impressive Acc and Hop-Acc results. However, DeepSeek-R1 requires significantly more time and tokens for reasoning, indicating G-HiRel is not sensitive to $\mathcal{M}$. As shown in Table 3, GPT-J-6B requires substantially more time and obtain worse and more unstable results. Therefore, compared to other baselines, the reasoning performance of G-HiRel increases with the improvement of $\mathcal{M}$. The chosen on $\mathcal{M}$ mainly depends on the time and cost by different LLMs.

5.2.3 Ablation Study (RQ3)

In this section, we remove key components of G-HiRel to illustrate their effectiveness, whose results are in Table 2. **G-HiRel$_{\text{w/o HI}}$** replaces the hierarchical relational retrieval with sub-questions. **G-HiRel$_{\text{w/o RPS}}$** replaces semantics-based relational path selection with a simple random selection. **G-HiRel$_{\text{w/o ML}}$** simplifies the multi-layer expansion into answering original question. **G-HiRel$_{\text{w/o MS}}$** removes multi-step expansions at each layer.

From Table 2, **(1) The performance is superior when all components work together.** On MQuAKE-CF-3k and MQuAKE-T, G-HiRel achieves the best performance on the multi-hop reasoning benchmarks under knowledge editing. **(2) The hierarchical instruction of G-HiRel is essential for LLM reasoning under knowledge editing.** The drop in performance of G-HiRel$_{\text{w/o HI}}$ indicates the impact of hierarchical instruction in integrating the structural edited KG into the LLM, which reduces the semantic gap between KG and natural language, facilitating the expansion of solutions based on the edited subgraph. In addition, G-HiRel$_{\text{w/o HI}}$ performs worse than G-HiRel$_{\text{w/o ML}}$, indicating the effectiveness of relational paths in editing. **(3) Multi-layer and multi-step expansion is crucial for building the edited subgraph.** The performance drops of G-HiRel$_{\text{w/o ML}}$ and G-

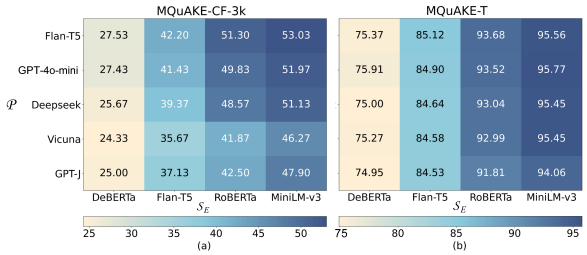

Figure 5: Acc (%) of different discriminator $\mathcal{P}$ and encoder $\mathcal{S}_E$, using GPT-4o-mini as $\mathcal{M}$.

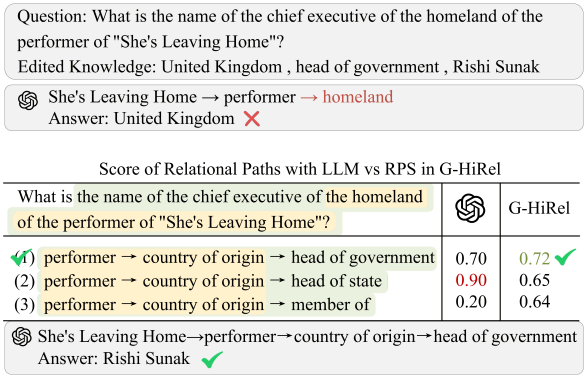

Figure 6: A case illustrating the effectiveness of proposed RPS in G-HiRel.

HiRel$_{\text{w/o MS}}$ indicate that multi-layer and multi-step expansion improves the relational retrieval performance during reasoning. **(4) RPS not only reduces reasoning cost but also is critical for selecting relational paths under knowledge editing.** The results of G-HiRel$_{\text{w/o RPS}}$ show that the relational semantics-based retrieval makes efficient selections than retrieval by the LLM under knowledge editing.

5.3 Weight Analysis (RQ5)

Analysis of $\mathcal{S}_E$. The effectiveness of RPS determines both the relevance of collected relational paths to the question and the quality of the edited subgraph. We test several language models as encoder $\mathcal{S}_E$, including DeBERTa-v3-Large (He et al., 2023), the Flan-T5-Large encoder, RoBERTa-Large (Liu et al., 2019), and MiniLM-L6-v3 (Reimers and Gurevych, 2019). As shown in Fig. 4 (b), G-HiRel achieves the best accuracy in the reasoning task when using MiniLM-L6-v3, indicating that the performance of $\mathcal{S}_E$ is critical for enabling the edited subgraph.

Analysis of $\mathcal{P}$. In the expansion, $\mathcal{P}$ is utilized for judging if moving to next layer. As shown in Fig. 5, when $\mathcal{S}_E$ is decided, various language models, such as Flan-T5-Large, GPT-4o-mini, DeepSeek-7B, Vicuna-7B, and GPT-J-6B, obtain different performance. As decided before, if $\mathcal{S}_E$ is MiniLM-L6-v3, G-HiRel obtains optimal results when $\mathcal{P}$ is Flan-T5-Large. This indicates that in the hierarchical relational retrieval, pretrained language model may obtain better reasoning results than LLMs, because of their stability in semantics-based retrieval.

Analysis of top-$\mathcal{T}$. During the expansion of the edited subgraph, the top-$\mathcal{T}$ relational paths with the highest relevance are retained as candidates. As shown in Fig. 4 (a), G-HiRel achieves the best performance on the benchmark of knowledge editing when $\mathcal{T} = 3$. This indicates that if $\mathcal{T}$ is too small, the edited subgraph may miss key information and become less effective. Moreover, if $\mathcal{T}$ is too large, irrelevant and redundant information may be introduced. Therefore, having too few or too many relational paths in the edited subgraph does not lead to positive outcomes.

5.4 Case Study (RQ3, 4)

In this section, we use a case in dataset to illustrate the effectiveness of G-HiRel. As shown in Fig. 6, the LLM tends to trust its internal knowledge and rejects the edited knowledge, resulting in an incorrect relational paths. Specifically, the GPT-4o-mini gets *United Kingdom*, which is an incorrect answer even with the edited knowledge *"The head of government of United Kingdom is Rishi Sunak"*. G-HiRel utilizes hierarchical instructions (shown as green and yellow blocks) to construct the edited subgraph based on relational paths and obtain entity independence, thereby alleviating knowledge inconsistency between the LLM and the KG after knowledge editing. As a result, in the case shown in Fig. 6, G-HiRel provides the correct reasoning process and answer. It also indicates the effectiveness of the RPS strategy. Our proposed semantics-based RPS can obtain more reliable score when extracting relational paths.

6 Conclusion

We propose G-HiRel, a framework to enhance the adaption of LLM reasoning under knowledge editing. G-HiRel firstly extracts KG containing edited knowledge. Secondly, the hierarchical instruction bridges the semantic gap between the model and KGs. Then it utilizes multi-layer and multi-step expansions for relational paths, supporting accurate reasoning and achieving the goal of knowledge editing. Experiments on datasets of multi-hop reasoning under knowledge editing verify the effectiveness of our approach.

7 Limitations

To the best of our knowledge, this work is a pioneering study on LLM reasoning under knowledge editing. G-HiRel still have some limitations to be improved in the future. It lacks an effective reasoning trace-back mechanism, which means that once the reasoning path deviates from the correct direction, the system cannot quickly correct the errors. This also explains the phenomenon, when adopting the question decomposition strategy, the performance of G-HiRel is worse than directly using the complete question as a guide.

References

Josh Achiam, Steven Adler, Sandhini Agarwal, Lama Ahmad, Ilge Akkaya, Florencia Leoni Aleman, Diogo Almeida, Janko Altenschmidt, Sam Altman, Shyamal Anadkat, and 1 others. 2023. Gpt-4 technical report. *arXiv preprint arXiv:2303.08774*.

Xiao Bi, Deli Chen, Guanting Chen, Shanhuang Chen, Damai Dai, Chengqi Deng, Honghui Ding, Kai Dong, Qiushi Du, Zhe Fu, Huazuo Gao, Kaige Gao, Wenjun Gao, Ruiqi Ge, Kang Guan, Daya Guo, Jianzhong Guo, Guangbo Hao, Zhewen Hao, and 67 others. 2024. Deepseek LLM: scaling open-source language models with longtermism. *CoRR*, abs/2401.02954.

Liyi Chen, Panrong Tong, Zhongming Jin, Ying Sun, Jieping Ye, and Hui Xiong. 2024a. Plan-on-graph: Self-correcting adaptive planning of large language model on knowledge graphs. *Advances in Neural Information Processing Systems*, 37:37665–37691.

Ruirui Chen, Weifeng Jiang, Chengwei Qin, Ishaan Singh Rawal, Cheston Tan, Dongkyu Choi, Bo Xiong, and Bo Ai. 2024b. Llm-based multi-hop question answering with knowledge graph integration in evolving environments. In *Findings of the Association for Computational Linguistics: EMNLP*, pages 14438–14451. Association for Computational Linguistics.

Siyuan Cheng, Ningyu Zhang, Bozhong Tian, Xi Chen, Qingbin Liu, and Huajun Chen. 2024a. Editing language model-based knowledge graph embeddings. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 38, pages 17835–17843.

Xin Cheng, Di Luo, Xiuying Chen, Lemao Liu, Dongyan Zhao, and Rui Yan. 2024b. Lift yourself up: Retrieval-augmented text generation with self-memory. *Advances in Neural Information Processing Systems*, 36.

Wei-Lin Chiang, Zhuohan Li, Zi Lin, Ying Sheng, Zhanghao Wu, Hao Zhang, Lianmin Zheng, Siyuan Zhuang, Yonghao Zhuang, Joseph E. Gonzalez, Ion Stoica, and Eric P. Xing. 2023. Vicuna: An open-source chatbot impressing gpt-4 with 90%* chatgpt quality.

Aakanksha Chowdhery, Sharan Narang, Jacob Devlin, Maarten Bosma, Gaurav Mishra, Adam Roberts, Paul Barham, Hyung Won Chung, Charles Sutton, Sebastian Gehrmann, and 1 others. 2023. Palm: Scaling language modeling with pathways. *Journal of Machine Learning Research*, 24(240):1–113.

Hyung Won Chung, Le Hou, Shayne Longpre, Barret Zoph, Yi Tay, William Fedus, Yunxuan Li, Xuezhi Wang, Mostafa Dehghani, Siddhartha Brahma, Albert Webson, Shixiang Shane Gu, Zhuyun Dai, Mirac Suzgun, Xinyun Chen, Aakanksha Chowdhery, Alex Castro-Ros, Marie Pellat, Kevin Robinson, and 16 others. 2024. Scaling instruction-finetuned language models. *J. Mach. Learn. Res.*, 25:70:1–70:53.

Junnan Dong, Qinggang Zhang, Chuang Zhou, Hao Chen, Daochen Zha, and Xiao Huang. 2024. Cost-efficient knowledge-based question answering with large language models. In *Advances in Neural Information Processing Systems 38: Annual Conference on Neural Information Processing Systems 2024, NeurIPS 2024, Vancouver, BC, Canada, December 10 - 15, 2024*.

Junfeng Fang, Houcheng Jiang, Kun Wang, Yunshan Ma, Shi Jie, Xiang Wang, Xiangnan He, and Tat-Seng Chua. 2024. Alphaedit: Null-space constrained knowledge editing for language models. *arXiv preprint arXiv:2410.02355*.

Junfeng Fang, Houcheng Jiang, Kun Wang, Yunshan Ma, Jie Shi, Xiang Wang, Xiangnan He, and Tat-Seng Chua. 2025. Alphaedit: Null-space constrained model editing for language models. In *The Thirteenth International Conference on Learning Representations*.

Jiazhan Feng, Chongyang Tao, Xiubo Geng, Tao Shen, Can Xu, Guodong Long, Dongyan Zhao, and Daxin Jiang. 2024. Synergistic interplay between search and large language models for information retrieval. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), ACL 2024, Bangkok, Thailand, August 11-16, 2024*, pages 9571–9583. Association for Computational Linguistics.

Michael R. Glass, Gaetano Rossiello, Md. Faisal Mahbub Chowdhury, Ankita Naik, Pengshan Cai, and Alfio Gliozzo. 2022. Re2g: Retrieve, rerank, generate. In *Proceedings of the 2022 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL 2022, Seattle, WA, United States, July 10-15, 2022*, pages 2701–2715. Association for Computational Linguistics.

Hengrui Gu, Kaixiong Zhou, Xiaotian Han, Ninghao Liu, Ruobing Wang, and Xin Wang. 2024. Pokemqa: Programmable knowledge editing for multi-hop question answering. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), ACL 2024, Bangkok, Thailand, August 11-16, 2024*, pages 8069–8083. Association for Computational Linguistics.

Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Ruoyu Zhang, Runxin Xu, Qihao Zhu, Shirong Ma, Peiyi Wang, Xiao Bi, and 1 others. 2025. Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning. *arXiv preprint arXiv:2501.12948*.

Pengcheng He, Jianfeng Gao, and Weizhu Chen. 2023. Debertav3: Improving deberta using electra-style pre-training with gradient-disentangled embedding sharing. In *The Eleventh International Conference on Learning Representations, ICLR 2023, Kigali, Rwanda, May 1-5, 2023*. OpenReview.net.

Xiaopeng Li, Shasha Li, Shezheng Song, Jing Yang, Jun Ma, and Jie Yu. 2024. Pmet: Precise model editing in a transformer. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 38, pages 18564–18572.

Xujian Liang and Zhaoquan Gu. 2025. Fast think-on-graph: Wider, deeper and faster reasoning of large language model on knowledge graph. In *AAAI-25, Sponsored by the Association for the Advancement of Artificial Intelligence*, pages 24558–24566. AAAI Press.

Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2019. Roberta: A robustly optimized BERT pretraining approach. *CoRR*, abs/1907.11692.

Yifan Lu, Yigeng Zhou, Jing Li, Yequan Wang, Xuebo Liu, Daojing He, Fangming Liu, and Min Zhang. 2025. Knowledge editing with dynamic knowledge graphs for multi-hop question answering. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 39, pages 24741–24749.

Linhao Luo, Yuan-Fang Li, Gholamreza Haffari, and Shirui Pan. 2024a. Reasoning on graphs: Faithful and interpretable large language model reasoning. In *The Twelfth International Conference on Learning Representations, ICLR*. OpenReview.net.

Linhao Luo, Yuan-Fang Li, Gholamreza Haffari, and Shirui Pan. 2024b. Reasoning on graphs: Faithful and interpretable large language model reasoning. In *The Twelfth International Conference on Learning Representations, ICLR*. OpenReview.net.

Yun Luo, Zhen Yang, Fandong Meng, Yafu Li, Jie Zhou, and Yue Zhang. 2023. An empirical study of catastrophic forgetting in large language models during continual fine-tuning. *arXiv preprint arXiv:2308.08747*.

Mingyu Derek Ma, Xiaoxuan Wang, Po-Nien Kung, P Jeffrey Brantingham, Nanyun Peng, and Wei Wang. 2023. Star: Improving low-resource information extraction by structure-to-text data generation with large language models. In *NeurIPS 2023 Workshop on Synthetic Data Generation with Generative AI*.

Aman Madaan, Niket Tandon, Peter Clark, and Yiming Yang. 2022. Memory-assisted prompt editing to improve GPT-3 after deployment. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing, EMNLP*, pages 2833–2861. Association for Computational Linguistics.

Costas Mavromatis and George Karypis. 2024. GNN-RAG: graph neural retrieval for large language model reasoning. *CoRR*, abs/2405.20139.

Kevin Meng, Arnab Sen Sharma, Alex J. Andonian, Yonatan Belinkov, and David Bau. 2023. Mass-editing memory in a transformer. In *The Eleventh International Conference on Learning Representations, ICLR 2023, Kigali, Rwanda, May 1-5, 2023*. OpenReview.net.

Eric Mitchell, Charles Lin, Antoine Bosselut, Chelsea Finn, and Christopher D. Manning. 2022. Fast model editing at scale. In *The Tenth International Conference on Learning Representations, ICLR 2022, Virtual Event, April 25-29, 2022*. OpenReview.net.

João Monteiro, Pierre-André Noël, Étienne Marcotte, Sai Rajeswar Mudumba, Valentina Zantedeschi, David Vázquez, Nicolas Chapados, Chris Pal, and Perouz Taslakian. 2024. Repliqa: A question-answering dataset for benchmarking llms on unseen reference content. In *Advances in Neural Information Processing Systems 38: Annual Conference on Neural Information Processing Systems 2024, NeurIPS*.

Shiwen Ni, Dingwei Chen, Chengming Li, Xiping Hu, Ruifeng Xu, and Min Yang. 2024. Forgetting before learning: Utilizing parametric arithmetic for knowledge updating in large language models. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), ACL*, pages 5716–5731. Association for Computational Linguistics.

Yunjia Qi, Hao Peng, Xiaozhi Wang, Bin Xu, Lei Hou, and Juanzi Li. 2024. ADELIE: aligning large language models on information extraction. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing, EMNLP 2024, Miami, FL, USA, November 12-16, 2024*, pages 7371–7387. Association for Computational Linguistics.

Nils Reimers and Iryna Gurevych. 2019. Sentence-bert: Sentence embeddings using siamese bert-networks. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing*.

Thomas Scialom, Tuhin Chakrabarty, and Smaranda Muresan. 2022. Fine-tuned language models are continual learners. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing, EMNLP*, pages 6107–6122. Association for Computational Linguistics.

Zhihong Shao, Yeyun Gong, Yelong Shen, Minlie Huang, Nan Duan, and Weizhu Chen. 2023. Enhancing retrieval-augmented large language models

with iterative retrieval-generation synergy. In *Findings of the Association for Computational Linguistics: EMNLP 2023, Singapore, December 6-10, 2023*, pages 9248–9274. Association for Computational Linguistics.

Yucheng Shi, Qiaoyu Tan, Xuansheng Wu, Shaochen Zhong, Kaixiong Zhou, and Ninghao Liu. 2024a. Retrieval-enhanced knowledge editing in language models for multi-hop question answering. In *Proceedings of the 33rd ACM International Conference on Information and Knowledge Management*, pages 2056–2066.

Yucheng Shi, Qiaoyu Tan, Xuansheng Wu, Shaochen Zhong, Kaixiong Zhou, and Ninghao Liu. 2024b. Retrieval-enhanced knowledge editing in language models for multi-hop question answering. In *Proceedings of the 33rd ACM International Conference on Information and Knowledge Management, CIKM*, pages 2056–2066. ACM.

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

Qiaoyu Tang, Jiawei Chen, Zhuoqun Li, Bowen Yu, Yaojie Lu, Cheng Fu, Haiyang Yu, Hongyu Lin, Fei Huang, Ben He, Xianpei Han, Le Sun, and Yongbin Li. 2024. Self-retrieval: End-to-end information retrieval with one large language model. In *Advances in Neural Information Processing Systems 38: Annual Conference on Neural Information Processing Systems 2024, NeurIPS 2024, Vancouver, BC, Canada, December 10 - 15, 2024*.

Komal K. Teru, Etienne G. Denis, and William L. Hamilton. 2020. Inductive relation prediction by subgraph reasoning. In *Proceedings of the 37th International Conference on Machine Learning, ICML*, volume 119 of *Proceedings of Machine Learning Research*, pages 9448–9457. PMLR.

Romal Thoppilan, Daniel De Freitas, Jamie Hall, Noam Shazeer, Apoorv Kulshreshtha, Heng-Tze Cheng, Alicia Jin, Taylor Bos, Leslie Baker, Yu Du, and 1 others. 2022. Lamda: Language models for dialog applications. *arXiv preprint arXiv:2201.08239*.

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

Ben Wang and Aran Komatsuzaki. 2021. GPT-J-6B: A 6 Billion Parameter Autoregressive Language Model. https://github.com/kingoflolz/mesh-transformer-jax.

Peng Wang, Zexi Li, Ningyu Zhang, Ziwen Xu, Yunzhi Yao, Yong Jiang, Pengjun Xie, Fei Huang, and Huajun Chen. 2024a. Wise: Rethinking the knowledge memory for lifelong model editing of large language models. *Advances in Neural Information Processing Systems*, 37:53764–53797.

Song Wang, Yaochen Zhu, Haochen Liu, Zaiyi Zheng, Chen Chen, and Jundong Li. 2024b. Knowledge editing for large language models: A survey. *ACM Computing Surveys*, 57(3):1–37.

Xiaohan Wang, Shengyu Mao, Shumin Deng, Yunzhi Yao, Yue Shen, Lei Liang, Jinjie Gu, Huajun Chen, and Ningyu Zhang. 2024c. Editing conceptual knowledge for large language models. In *Findings of the Association for Computational Linguistics: EMNLP 2024, Miami, Florida, USA, November 12-16, 2024*, pages 706–724. Association for Computational Linguistics.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Fei Xia, Ed Chi, Quoc V Le, Denny Zhou, and 1 others. 2022. Chain-of-thought prompting elicits reasoning in large language models. *Advances in neural information processing systems*, 35:24824–24837.

Xiaobao Wu, Liangming Pan, William Yang Wang, and Anh Tuan Luu. 2024. AKEW: Assessing knowledge editing in the wild. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*.

Rongwu Xu, Zehan Qi, Zhijiang Guo, Cunxiang Wang, Hongru Wang, Yue Zhang, and Wei Xu. 2024. Knowledge conflicts for llms: A survey. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing, EMNLP*, pages 8541–8565. Association for Computational Linguistics.

Lang Yu, Qin Chen, Jie Zhou, and Liang He. 2024. Melo: Enhancing model editing with neuron-indexed dynamic lora. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 38, pages 19449–19457.

Ce Zheng, Lei Li, Qingxiu Dong, Yuxuan Fan, Zhiyong Wu, Jingjing Xu, and Baobao Chang. 2023. Can we edit factual knowledge by in-context learning? In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing, EMNLP*, pages 4862–4876. Association for Computational Linguistics.

Zexuan Zhong, Zhengxuan Wu, Christopher D. Manning, Christopher Potts, and Danqi Chen. 2023. Mquake: Assessing knowledge editing in language models via multi-hop questions. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing, EMNLP 2023, Singapore, December 6-10, 2023*, pages 15686–15702. Association for Computational Linguistics.

Xiangrong Zhu, Yuexiang Xie, Yi Liu, Yaliang Li, and Wei Hu. 2025. Knowledge graph-guided retrieval augmented generation. In *Proceedings of the 2025 Conference of the Nations of the Americas Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, pages 8912–8924.

Algorithm 1 The algorithm of G-HiRel

Require: Question q, LLM $\mathcal{M}$, Discriminator $\mathcal{P}$, Encoder $\mathcal{S}_E$, KG $\mathcal{G}$
Ensure: Final Answer p_{ans}

Extract starting entity e_s from q
Inject edited triples into KG $\mathcal{G}$
Initialize $G_0^{(0)} \leftarrow \{e_s\}$
Reformulate q via $\mathcal{M}$ into $S_q = \{q^1, q^2, \ldots, q^L\}$
for $l = 1$ to L **do**
  for $k = 0$ to $K - 1$ **do**
    if $\mathcal{P}$ determines $G_k^{(l)}$ is sufficient to answer q^l **then**
      continue
    else
      Collect candidate relational paths $\overline{P}_{k+1}^{(l)}$ from $\mathcal{G}$ for each $p_{k,i}^{(l)} \in G_k^{(l)}$
      Initialize empty score list $S \leftarrow [\,]$
      for all $p_{k+1}^{(l)} \in \overline{P}_{k+1}^{(l)}$ **do**
        Compute score $s \leftarrow \mathcal{S}_E(p_{k+1}^{(l)}, q^l)$
        Append s to S
      end for
      $G_{k+1}^{(l)} \leftarrow$ top-$\mathcal{T}$ paths in $\overline{P}_{k+1}^{(l)}$ ranked by S
    end if
  end for
end for
Select final relational path p_r from $G_K^{(L)}$ using $\mathcal{M}$
Instantiate P_{cand} from p_r
Generate answer $p_{\text{ans}} \leftarrow \mathcal{M}(P_{\text{cand}})$
return p_{ans}

Table 4: Notations and descriptions used in G-HiRel.

Notation	Description
$\mathcal{E}, \mathcal{R}$	The set of entities and relations.
$\mathcal{Q}$	The set of questions.
q	The text of an original question, $q \in \mathcal{Q}$.
e_s	The start entity of a question, $e_s \in \mathcal{E}$.
$q^{(l)}$	The hierarchical instruction at the l-th layer, $l \in \{1, 2, \cdots, L\}$.
S_q	The set of hierarchical instructions.
$\mathcal{G}$	The KG extracted as the repository for editing from the large-scale knowledge base based on $\mathcal{Q}$.
$G_k^{(l)}$	The edited subgraph at the l-th layer and k-th step of expansion.
$p_{k,i}^{(l)}$	The i-th relational path after k steps of expansion in the l-th layer.
$\overline{P}_k^{(l)}$	Set of candidate relational paths at the l-th layer and (k)-th step.
$P_{k,\mathcal{T}}^{(l)}$	Set of top-$\mathcal{T}$ relational paths after k steps of exploration in the l-th layer.
$E_{k,i}^{(l)}$	The set of tail entities of $p_{k,i}^{(l)}$.
$\mathcal{N}(\cdot)$	The neighbor relations of an entity.
$\mathcal{S}_E$	The encoder for relational path selection.
$\boldsymbol{q}^{(l)}$	Feature vector $\boldsymbol{q}^{(l)} \in \mathbb{R}^m$ of $q^{(l)}$ via $\mathcal{S}_E$.
$\boldsymbol{p}_{k,i}^{(l)}$	Feature vector $\boldsymbol{p}_{k,i}^{(l)} \in \mathbb{R}^m$ of $p_{k,i}^{(l)}$ via $\mathcal{S}_E$.
p_r	The final relational path from $P_{k,\mathcal{T}}^{(L)}$ identified by the LLM using few-shot prompts.
P_{cand}	Candidate reasoning paths instantiated on $G_k^{(l)}$.
p_{ans}	Final reasoning path used to answer the complex question.
$\mathcal{M}$	The LLM for reasoning.
$\mathcal{P}$	The discriminator for determining whether further exploration is necessary.

A Supplementary Details of G-HiRel

A.1 Pseudocode and Code of G-HiRel

The pseudocode of G-HiRel is shown in Algorithm 1, and the code of G-HiRel is available at the anonymous hyperlink: https://anonymous.4open.science/r/GraphReasoningKE-24AB.

B Notation table

All the notations are in Table 4.

C Supplementary Experimental Settings

C.1 Baseline Methods

In this section, the descriptions about the baseline methods are as follows:

- **MEMIT** (Meng et al., 2023) updates the feed-forward networks to modify the knowledge embedded in the model.
- **MEND** (Mitchell et al., 2022) employs gradient decomposition for low-rank parameter updates and auxiliary editing networks for rapid local tuning.
- **AlphaEdit** (Fang et al., 2025) projects the perturbation onto the null space of the preserved knowledge before applying it to the parameters, thereby ensuring that the preserved knowledge remains unaffected after editing.
- **Mello** (Zhong et al., 2023) stores updated facts externally. It decomposes complex questions step by step during reasoning, correcting the reasoning process with updated facts.
- **PokeMQA** (Gu et al., 2024) employs a memory-based strategy, which urges LLMs to decompose knowledge-enhanced multi-hop queries. A conflict detector is used to identify conflicts in external memory and uses external signals to adjust reasoning of LLMs.
- **GMello** (Chen et al., 2024b) leverages external memory to store edited knowledge and incorporates KBQA to generate answers by integrating the reasoning results of LLMs and the query results from KBQA.
- **CoT** (Wei et al., 2022) introduces a multi-hop chain reasoning process to guide LLMs in

Table 5: Acc of discriminator $\mathcal{P}$ selection on MQuAKE-CF-3k and MQuAKE-T. The symbols ↓ and ↑ denote the performance variations when different $\mathcal{P}$ are employed with the same $\mathcal{M}$.

$\mathcal{M}$	$\mathcal{P}$	MQuAKE-CF-3k			MQuAKE-T		
		edit=1	**edit=100**	**edit=3000**	**edit=1**	**edit=100**	**edit=1868**
GPT-4o-mini	Flan-T5-Large	72.37	63.97	53.03	96.20	95.82	95.56
	GPT-4o-mini	$69.40_{\downarrow 2.97}$	$62.03_{\downarrow 1.94}$	$51.97_{\downarrow 1.06}$	$96.41_{\uparrow 0.21}$	$96.20_{\uparrow 0.38}$	$95.77_{\uparrow 0.21}$
	DeepSeek-7B	$67.83_{\downarrow 4.45}$	$61.23_{\downarrow 2.74}$	$51.13_{\downarrow 1.90}$	$96.41_{\uparrow 0.21}$	$96.15_{\uparrow 0.33}$	$95.45_{\downarrow 0.11}$
	Vicuna-7B	$60.53_{\downarrow 11.84}$	$54.03_{\downarrow 9.94}$	$46.27_{\downarrow 6.76}$	$96.20_{\downarrow 0.00}$	$96.09_{\uparrow 0.27}$	$95.45_{\downarrow 0.11}$
	GPT-J-6B	$64.23_{\downarrow 8.14}$	$57.93_{\downarrow 6.04}$	$47.90_{\downarrow 5.13}$	$95.07_{\downarrow 1.13}$	$94.81_{\downarrow 1.01}$	$94.06_{\downarrow 1.50}$

solving complex tasks.

- **ToG** (Sun et al., 2024) uses a KG to guide the reasoning of LLMs, iteratively dragging the reasoning path through relationship pruning and entity pruning until the information is collected to answer the question.

- **KEDKG** (Lu et al., 2025) builds a dynamic KG to store and update edited knowledge, resolves conflicts, and uses question decomposition with retrieval of relevant entities and relations to supply the needed facts to the LLM, thus enabling accurate multi-hop reasoning.

- **RAE** (Shi et al., 2024b) injects edited knowledge by merging updated facts into KG, and performs multi-hop reasoning by retrieving question-relevant fact chains via mutual information score and entropy-based pruning to support in-context editing at inference time.

- **FastToG** (Liang and Gu, 2025) constructs multiple reasoning chains over a KG with communities by iteratively performing local community search on hop-bounded subgraphs and selecting query-relevant communities via modularity-based coarse pruning plus LLM-based fine pruning. Then converting the selected communities into text using Triple2Text or Graph2Text for LLM reasoning and answer generation.

C.2 Statistics of MQuAKE and WikiUpdate

This section presents statistics of the MQuAKE-CF-3k and MQuAKE-T datasets, as shown in Table 6. As observed, MQuAKE-CF-3k exhibits a relatively balanced distribution across different hop counts $(2, 3, 4)$ and edit numbers, whereas all instances in MQuAKE-T require only a single edit and are mostly concentrated on 2-hop reasoning problems.

Table 6: Statistics of datasets used in experiments.

Dataset	**Edits**	**1-hop**	**2-hop**	**3-hop**	**4-hop**	**Total**
MQuAKE-CF-3k	1	–	513	356	224	1,093
	2	–	487	334	246	1,067
	3	–	–	310	262	572
	4	–	–	–	268	268
MQuAKE-T	1 (All)	–	1,421	445	2	1,868
WikiUpdate	1 (All)	1,056	–	–	–	1,056

The WikiUpdate dataset contains instances that require only one edit and are limited to single-hop reasoning.

D Supplementary Experimental Results

D.1 Weight Analysis of discriminator $\mathcal{P}$

As described in Section 4.2.2, the discriminator $\mathcal{P}$ is essential in hierarchical relational retrieval because its performance directly determines whether expansion at each layer stops at the right moment, which in turn affects the quality of the final relational path p_r. To further validate the effectiveness of $\mathcal{P}$, we further design two comparative experiments evaluating a PLM against several LLMs. The PLM is Flan-T5-Large, and the LLMs include GPT-4o-mini, DeepSeek-7B, Vicuna-7B, and GPT-J-6B. Flan-T5-Large is prompted in a zero-shot setting (see Appendix E.2), while the LLMs use few-shot prompts to extract the target relation from each hierarchical instruction for the discrimination task. Firstly, we fixed the encoder $\mathcal{S}_E$ to MiniLM-L6-v3 and evaluated performance across different edit scenarios on the MQuAKE benchmark. As Table 5 shows, on MQuAKE-CF-3k, Flan-T5-Large consistently and significantly outperforms all the LLMs. On MQuAKE-T, although Flan-T5-Large performs slightly worse in most cases, the reductions remain $< 1.00\%$.

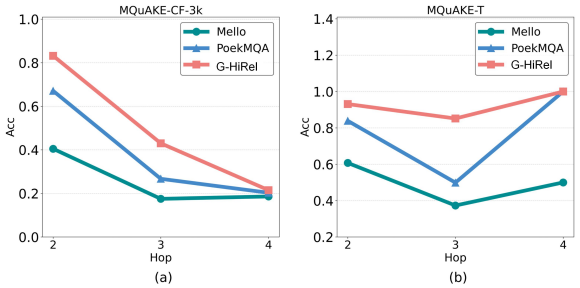

Figure 7: Performance of different hops of G-HiRel with DeepSeek-7B as $\mathcal{M}$.

D.2 Analysis of Encoder $\mathcal{S}_E$

We implement a study to further investigate the impact of different encoders $\mathcal{S}_E$ on the semantics-based RPS as introduced in Section 4.2.2. In this experiment, $\mathcal{P}$ is fixed as Flan-T5-Large, and the reasoning LLM $\mathcal{M}$ is GPT-4o-mini. $\mathcal{S}_E$ to be evaluated includes DeBERTa-v3-Large (He et al., 2023), Flan-T5-Large encoder, RoBERTa-Large-v1, and MiniLM-L6-v3. Among them, DeBERTa-v3-Large and Flan-T5-Large encoder employ mean pooling over token representations to generate sentence-level embeddings. As shown in the Fig. 8, MiniLM-L6-v3 achieves the best performance across all metrics in every edit scenario. Based on them, we select MiniLM-L6-v3 as the encoder $\mathcal{S}_E$ for the semantics-based RPS.

D.3 Analysis of top-$\mathcal{T}$

In this section, we conduct research to further explore how different values of $\mathcal{T}$ affect the performance of G-HiRel during the construction of the edited subgraph. $\mathcal{T}$ was introduced in Section 4.2.2. In this experiment, we choose $\mathcal{M}$ as GPT-4o-mini and then perform evaluations on MQuAKE-CF-3k and MQuAKE-T. The values of $\mathcal{T}$ are set to $1, 3$, and 5. As shown in Fig. 9, when $\mathcal{T}$ increases from 1 to 3, the performance of G-HiRel keeps rising. However, when it further increases from 3 to 5, the performance drops. This phenomenon shows that appropriately increasing the number of relational paths helps improve the quality of retrieval. However, when there are too many relational paths in the edited subgraph, it will introduce redundant or even wrong information, which creates noise and influences the reasoning results. Based on these observations, we set the final value of $\mathcal{T}$ as 3.

Table 7: Acc (%) on WikiUpdate. **Time** (hour) measures total runtime for each method.

Method	$\mathcal{M}$	edit=100		edit=1056	
		Acc↑	Time↓	Acc↑	Time↓
Mello	GPT-4o-mini	31.06	**2.50**	29.26	**2.45**
G-HiRel	GPT-4o-mini	**68.37**	**2.50**	**68.47**	2.76

Table 8: Time (hour) of our G-HiRel and ToG. $\mathcal{M}$ is GPT-4o-mini.

Method	MQuAKE-CF-3k			MQuAKE-T		
	edit=1	edit=100	edit=ALL	edit=1	edit=100	edit=ALL
G-HiRel	**45.73**	**49.64**	**46.76**	**61.18**	**60.31**	**62.28**
ToG	294.66	287.33	309.15	188.45	191.62	194.22

D.4 Results on WikiUpdate

In this section, we further supplement the experimental results on an additional dataset. We introduce WikiUpdate (Wu et al., 2024), a dataset designed for real-world knowledge editing tasks, which comprises 1,056 instances involving single-hop reasoning. We evaluate the performance of G-HiRel on the WikiUpdate dataset, with the results shown in Table 7. Compared to Mello, G-HiRel obtains significantly better performance while maintaining similar time. This shows that under a knowledge editing scenario, G-HiRel can achieve higher-quality and efficient performance in one-hop LLM reasoning as well.

D.5 Consumption of Time and Computing Resources

Although previous experiments demonstrate that the performance of G-HiRel varies with the reasoning LLM $\mathcal{M}$, the efficiency of reasoning is also a critical factor. This section presents the reasoning time (in hours) of G-HiRel across different models on two datasets. As shown in Table 3, GPT-J-6B requires substantially more time, whereas Vicuna-7B achieves the highest efficiency. Furthermore, using GPT-4o-mini as $\mathcal{M}$ on the MQuAKE benchmark, we thoroughly compare the efficiency of G-HiRel against ToG. As shown in Table 8, the time consumption of G-HiRel is significantly lower than that of ToG.

We conducted our experiments primarily using two NVIDIA RTX 4090 GPUs. For models which are not reproduced locally, including DeepSeek-R1 and GPT-4o-mini, we accessed them via their respective APIs.

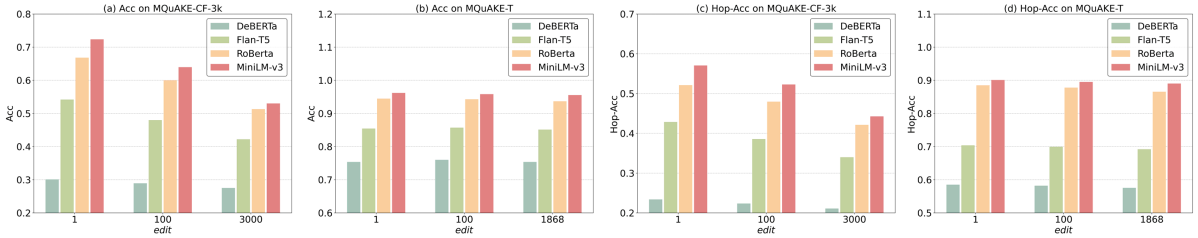

Figure 8: Results of different $\mathcal{S}_E$ on MQuAKE-CF-3k and MQuAKE-T.

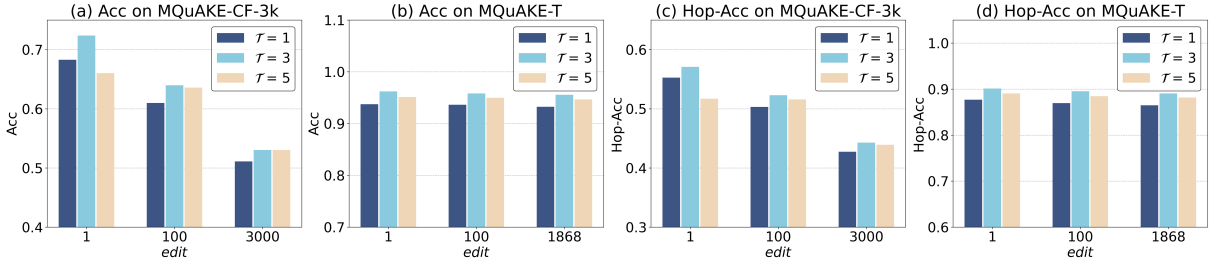

Figure 9: Results of different $\mathcal{T}$ on MQuAKE-CF-3k and MQuAKE-T.

E Prompts

E.1 Prompt to Extract $\mathcal{G}$

Task: Extract relationships from a given sentence based on relationships defined in **Wikidata**. You can learn how to do from the <Examples>.

Question and answer template:
Question [id] sentence: <sentence that require extraction of the relationships>
Question [id]: Based on <Relationships>, what is the target relationship of <Question [id] sentence>?
Answer [id]: [relationships of the <Question id sentence> which is in the <Relationships>]

Examples: *In Context Few shot*

Now is start, please answer the question as the examples and do not output any extra information.

Question 0 sentence: {Question}
Question 0: Based on the relationships defined in **Wikidata**, what is the target relationship of <Question 0 sentence>?
Answer 0: {Answer}

E.2 Prompt for Retrieval Termination

Is the reasoning Path: "{relational path}" sufficient to describe the sentence: "{question}"? Answer only yes or no.

E.3 Prompt to Obtain the Final Hop Relation of Each Question

TASK: Extract target relationships from a given sentence based on <Relationships>. You can learn how to do from the <Examples>.

Relationships: {Relationships} <Relationships> is a collection of relationships, separated by commas.

Question and answer template:
Question [id] sentence: <sentence that require extraction of the target relationship>
Question [id]: Based on <Relationships>, what is the target relationship of <Question [id] sentence>?
Answer [id]: [target relation of the <Question id sentence> which is in the <Relationships>]

Description of symbols:
Pointed brackets ("<" and ">") indicate references to existing information; square brackets ("[" and "]") indicate information that may be subject to change.

Examples:
In Context Few Shot

Now is start, please answer the question as the examples and do not output any extra information.

Question 0 sentence: {Sentence} Question 0: Based on <Relationships>, what is the target relationship of <Question 0 sentence>? Answer 0: {Answer}

E.4 Prompt to Obtain the Start Entity

Prompt Example: For the following questions, we extract the initial entities involved in the question through the entity set based on **Wikidata**.

In Context Few shot

Question_id. Question: {Question}

E.5 Prompt to Get Hierarchical Instructions

Task: The given text is a complex problem. It is necessary to reconstruct the text layer by layer based on the corresponding Start Entity after understanding the problem text, ensuring that each sub-text is semantically unique, concise, and non-repetitive. For specific practices, you can refer to the requirement or learn from my examples.

Requirement:
1. The number of sub-texts in Split is greater than or equal to 1
2. The text in Split must be completely derived from the corresponding Text
3. The sub-text of the next level must contain the complete text of the sub-text of the previous level
4. Start Entity cannot appear as a separate subtext

In Context Few Shot

Now is start, please answer the question as the examples and do not output any extra information.

Question_id. Text: {Question}
Start Entity: {Start Entity}
Question_id. Split: {Answer}

E.6 Prompt to Get Final Answer

 # The prompt for selecting relational paths from the edited subgraph

Task: Based on the provided information, from the corresponding information in the question, select a relational path which is most likely to answer the question.
You can learn how to do this from the examples I've provided.

Examples:
In Context Few Shot

Now is start, please answer the question as the examples and do not output any extra information.
Question_id: {Question}
Question_id information: (1) {Relational Path} ...
Question_id relation reasoning path: {Answer}

 # The prompt for selecting reasoning path after obtaining relational path
Task: Based on the provided information, select a reasoning path to answer the question based on edit information.
You can learn how to do this from the examples provided.

Important: - You must compare ALL candidate reasoning paths before selecting the final answer. - If any reasoning step matches exactly with the edit information (full triple: head → relation → tail), you must prioritize it. - If no step matches edit information, you must select the reasoning path with the latest time (the latest). - You must NOT stop at the first reasonable option; you must check all options completely.

Examples:

In Context Few Shot

Now is start, please answer the question as the examples and do not output any extra information.
Question_id: {Question}
Question_id edit information: (1) {Edit Triple} ...
Question_id candidate reasoning path: (1) {Reasoning Path} ...
Please select one reasoning path from Question_id candidate reasoning path to answer Question_id.
Question_id answer: {Answer}