Retrieval-enhanced Knowledge Editing in Language Models for Multi-Hop Question Answering

Yucheng Shi
University of Georgia
Athens, Georgia, USA
yucheng.shi@uga.edu

Qiaoyu Tan
New York University
New York, USA
qiaoyu.tan@nyu.edu

Xuansheng Wu
University of Georgia
Athens, Georgia, USA
xuansheng.wu@uga.edu

Shaochen Zhong
Rice University
Houston, Texas, USA
shaochen.zhong@rice.edu

Kaixiong Zhou
North Carolina State University
Raleigh, North Carolina, USA
kzhou22@ncsu.edu

Ninghao Liu
University of Georgia
Athens, Georgia, USA
ninghao.liu@uga.edu

Abstract

Large Language Models (LLMs) have shown proficiency in question-answering tasks but often struggle to integrate real-time knowledge, leading to potentially outdated or inaccurate responses. This problem becomes even more challenging when dealing with multi-hop questions, since they require LLMs to update and integrate multiple knowledge pieces relevant to the questions. To tackle the problem, we propose the Retrieval-Augmented model Editing (RAE) framework for multi-hop question answering. RAE first retrieves edited facts and then refines the language model through in-context learning. Specifically, our retrieval approach, based on mutual information maximization, leverages the reasoning abilities of LLMs to identify chain facts that traditional similarity-based searches might miss. In addition, our framework includes a pruning strategy to eliminate redundant information from the retrieved facts, which enhances the editing accuracy and mitigates the hallucination problem. Our framework is supported by theoretical justification for its fact retrieval efficacy. Finally, comprehensive evaluation across various LLMs validates RAE's ability in providing accurate answers with updated knowledge. Our code is available at: https://github.com/sycny/RAE.

CCS Concepts

• **Information systems** → **Question answering**; • **Computing methodologies** → **Knowledge representation and reasoning**.

Keywords

Model editing; question answering; retrieval-augmented generation

ACM Reference Format:
Yucheng Shi, Qiaoyu Tan, Xuansheng Wu, Shaochen Zhong, Kaixiong Zhou, and Ninghao Liu. 2024. 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 '24), October 21–25, 2024, Boise, ID, USA.* ACM, New York, NY, USA, 11 pages. https://doi.org/10.1145/3627673.3679722

CIKM '24, October 21–25, 2024, Boise, ID, USA

ACM ISBN 979-8-4007-0436-9/24/10
https://doi.org/10.1145/3627673.3679722

1 Introduction

Large language models (LLMs) excel in answering factual questions due to their pre-training on extensive corpora [27, 32, 34]. However, their dependence on pre-trained knowledge can lead to outdated answers, which requires updates through model editing techniques [19, 43, 49, 51]. Among them, editing for **multi-hop questions** is particularly important, as real-world questions often require combining multiple knowledge pieces. For example, to answer "*Who is married to the British Prime Minister?*", one must connect several related facts, creating a **fact chain**, like "*(United Kingdom, head of government, Theresa May)*, "*(Theresa May, spouse, Philip May)*". If we update "*Theresa May*" to "*Rishi Sunak*" to reflect real-world change [1], it requires adjusting the related facts accordingly, resulting in a new fact chain: "*(United Kingdom, head of government, Rishi Sunak), (Rishi Sunak, spouse, Akshata Murty)*". Recent research shows that retrieval-augmented generation (RAG) is effective in LLM editing for multi-hop question answering [36, 52], surpassing other methods such as fine-tuning [53] and locate-and-edit [19, 20] approaches. The RAG-based methods first retrieve relevant facts related to the question and then feed these facts into LLMs through in-context learning. Studies have shown that LLMs are highly receptive to external knowledge, even when it contradicts their internal pre-trained knowledge [16, 39]. Therefore, RAG can effectively update knowledge within LLMs but also avoid problems such as catastrophic forgetting [12, 13] and hallucinations [10]. However, simply using multi-hop questions as queries in RAG often fails to retrieve pertinent facts due to the complexity of the questions involved, as illustrated in Figure 1.

To address this, some methods [36, 52] propose breaking down complex multi-hop questions into simpler, single-hop queries to improve the effectiveness of the retrieval process. Despite enhancements, critical issues persist: 1) Commonly used models with fewer parameters (e.g., Llama2-7B [31]) exhibit significantly worse editing performance (more than 30% gap) compared to more powerful models like GPT-3.5 or GPT-4 [52]; 2) The performance of existing editing methods degrades as the number of editing cases increases, making them impractical for large-scale editing tasks [36, 52]. The possible reason behind their performance degradation is that their editing effectiveness heavily relies on the quality of sub-questions

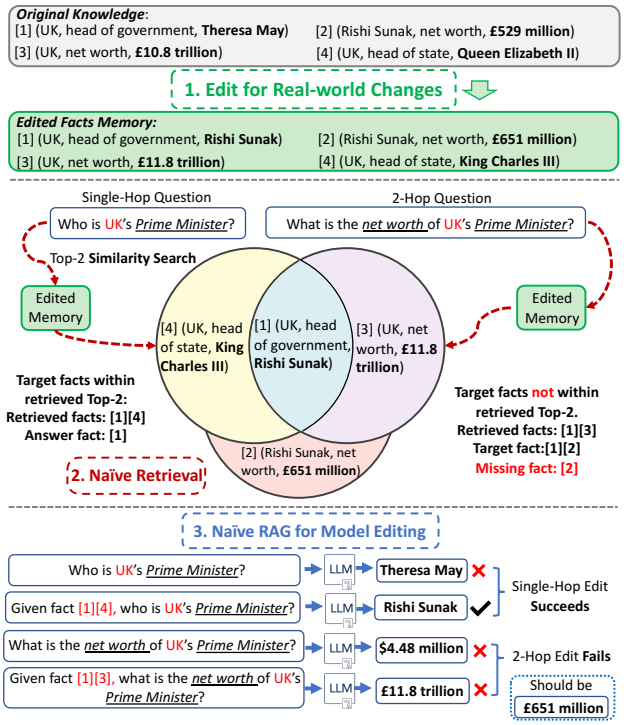


Figure 1: An example of the traditional similarity-based search that fails to retrieve the correct facts for LLM editing.

generated for querying. Generating accurate sub-questions is challenging for models with less strong reasoning and planning capabilities. Inaccurate sub-questions will result in irrelevant fact retrieval, which may mislead the LLMs and reduce editing effectiveness [45].

Since successful multi-hop editing depends on **accurate retrieval of the question-specific facts**, instead of applying the existing "generate then retrieve" approach, we propose to directly fetch the required facts from the database by leveraging the next-token prediction capabilities of LLMs. Below are the details of our editing method.

1. The connection between edited facts helps retrieval: Each edited fact can be represented as a triplet of *"(head entity, relation, tail entity)"*. The nearby facts in a fact chain are connected through entities: the tail entity of one fact becomes the head entity of the next. This observation inspires us to adopt a knowledge graph (KG) for storing these triplets. In a KG, each entity has only a limited number of neighboring entities, which narrows down the retrieval choices to a feasible number, rather than having to search through a vast database. Our method differs from existing approaches that store edited facts as embeddings in a vector database, whose editing performance decreases as the number of edits grows [36, 52].

2. Next-token prediction helps next-fact retrieval: LLMs are inherently skilled at predicting the next token in a sequence. In our study, we extend this capability to predict the next fact in a fact chain. Our approach first feeds LLMs a sequence that includes the question, any preceding fact, and the relevant entity. Then, the model predicts this entity's next possible logical relation within the context of its input question. Finally, we retrieve the tail entity from the KG, based on the head entity and relation, to complete the fact chain. However, directly predicting the most probable next fact can lead to biases toward frequently occurring words [30, 37]. Therefore, we predict facts that share the most relevant information with the question and preceding facts, using mutual information (MI) as our retrieval metric. We develop a technique to break down the MI into conditional probabilities that LLMs can effectively approximate [27], thus improving the accuracy of our predictions.

3. LLM internal state helps reduce redundancy: Irrelevant facts to the editing will mislead LLMs. Thus, we propose to prune redundant facts extracted by the retrieval step using the LLM's prediction uncertainty. The uncertainty is minimized when the LLM is prompted with a correct fact chain, and increases when provided with incomplete or excessive facts. We quantify this uncertainty using the LLM's output entropy. Unlike traditional methods that either limit the number of retrieval attempts or simply ask the model to identify redundant facts [36, 52], our approach provides a more effective way to ensure accuracy in the editing process.

Overall, we name our approach as Retrieval-Augmented model Editing (RAE), where we introduce a novel fact retrieval method for multi-hop questions in model editing. We also propose a knowledge-pruning strategy to reduce noise after the initial retrieval, mitigating the hallucination problem. Additionally, we provide theoretical analysis to justify our design for the retrieval objective.

2 Preliminary: Model Editing

2.1 Model Editing for Single-hop Questions

In LLMs, *a single model edit* refers to updating a specific piece of factual knowledge [20, 21, 50]. Each *knowledge* is defined as a triplet $\delta := (h, r, t)$, where h, r, and t denote the head entity, the relation, and the tail entity, respectively, such as (Misery, author, Stephen King). An edit is defined as changing the tail entity t to a new entity t', i.e., $\delta \rightarrow \delta' := (h, r, t) \rightarrow (h, r, t')$, where δ' is the edited knowledge. Let q denote the language model's input. The goal of model editing is to modify a target model f_θ, so that the new model f'_θ produces an output $f'_\theta(q)$ that follows the new fact δ', where $f'_\theta(q) \neq f_\theta(q)$. Specifically, given $q = [h; r]$, $h, r \in \delta'$, the model is expected to output $t' = f'_\theta([h; r])$, where $[;]$ is the concatenation operator. However, if the input question is not relevant to the edit, i.e., $h \notin \delta'$ or $r \notin \delta'$, the model should output $t = f'_\theta([h; r])$ that reflects the original knowledge of LLMs.

2.2 Model Editing for Multi-hop Questions

Answering multi-hop questions presents a greater challenge. A multi-hop question seeks to identify a specific tail entity t_k based on a sequence of linked facts: $\{(h_1, r_1, t_1), (h_2, r_2, t_2), ..., (h_k, r_k, t_k)\}$, where each tail entity is the head entity of the next fact: $t_i = h_{i+1}$. Answering each input question q requires a **fact chain** G_q. A k-hop question can be formulated using only the initial head entity h_1, and a series of relationships $\{r_1, r_2, ..., r_k\}$. An example of model editing for a 3-hop question is shown in Table 1. Here, we use counterfactual edits to simulate real-world updates. Different fact chains are used to answer the question before and after editing. One key observation is that fact chains represent connected knowledge graphs, where a single entity is involved in two consecutive facts. Additionally, we notice a "**ripple effect**" in these chains: An edit in the first fact δ_1 will lead to changes in the subsequent facts, forming a new chain

Table 1: Answering a 3-hop question q with counterfactual model editing. The pre-edited and edited answer are t_5 and t_3^*, respectively. t_5 and t_3^* are the tail entity of δ_5 and δ_3'. G_q and G_q^* denote the pre-edited and edited fact chain.

Edited Fact Bank Δ and Unedited Facts
$(\delta_1 \rightarrow \delta_1')$ Misery, author, Stephen King → Richard Dawkins #edit
(δ_2) Richard Dawkins, citizen of, United Kingdom
$(\delta_3 \rightarrow \delta_3')$ United Kingdom, capital, London → Birmingham #edit
(δ_4) Stephen King, citizen of, United States
(δ_5) United States, capital, Washington, D.C.
A 3-hop Question q
(q) Which city is the capital of the country where the author of Misery held citizenship?
Pre-edited Answer t_5 and Edited Answer t_3'
(t_5) Washington, D.C.
(t_3') Birmingham
Pre-edited Fact Chain G_q
(δ_1) (Misery , author, Stephen King)
(δ_4) (Stephen King , citizen of, United States)
(δ_5) (United States , capital, Washington, D.C.)
Edited Fact Chain G_q^*
(δ_1') (Misery , author, Richard Dawkins) #edited fact
(δ_2) (Richard Dawkins , citizen of, United Kingdom) #unedited fact
(δ_3') (United Kingdom , capital, Birmingham) #edited fact

G_q^*. In practice, knowledge editing is usually conducted in batches, involving multiple fact changes simultaneously, resulting in an edited fact bank $\Delta = \{\delta_1', \delta_2', ..., \delta_N'\}$, where N is a large number [20]. Locating the relevant edited facts for a question is non-trivial due to the "ripple effect". To correctly answer multi-hop questions after model editing, it is crucial to address the retrieval problem formally defined as follows:

Problem 1 (Retrieval-Augmented Editing). *Given an edited fact bank $\Delta = \{\delta_1', \delta_2', ..., \delta_N'\}$ with N instances and a multi-hop question q whose answer requires model editing, we want to retrieve its corresponding edited facts $\Delta_q = \{\delta_i', \delta_j', ..., \delta_k'\}$. The goal is to ensure that all the edited facts necessary for answering q are retrieved, i.e., $\Delta_q \subseteq G_q^*$ and $\Delta \backslash \Delta_q \not\subset G_q^*$. Then, these facts are used to refine the target model f_θ for editing.*

3 Methodology

Our Retrieval-Augmented Editing (RAE) framework, as shown in Figure 2, contains two key steps: (1) retrieving edited facts relevant to the question, and (2) editing the language model using these retrieved facts via in-context learning. We will first discuss step (2) with the motivation of our design in the following section. The details of step (1) are in Sections 3.2 and 3.3.

3.1 Retrieval-Augmented Editing

A naïve edit approach uses similarity-based search to retrieve edited facts similar to target question q [14, 35, 52]. These facts are then integrated into a prompt template for editing via in-context learning: $f_\theta'(q) = f_\theta(T_e(q, \{\delta_1', \delta_2', ...\delta_K'\}))$, where T_e is the editing template. For example, $T_e(\cdot)$ can be made as *"Given fact: {δ'}, {q} ?"*. The Top-K nearest edited facts to question q in the embedding space are denoted as $\{\delta_1', \delta_2', ...\delta_K'\} = \text{Top-}K_{\delta \in \Delta}\, \text{sim}(g_z(\delta), g_z(q))$, where $\text{sim}(\cdot)$ denotes the similarity function and g_z is an embedding model. However, the edited facts Δ_q needed to answer q are difficult to retrieve by this approach since they usually contain entities different from q, which will result in a low similarity score in a large bank Δ (e.g., in Table 1, *United Kingdom* in δ_3', but not in q).

To address this problem, we propose **edited fact chain extraction** to obtain G_q^*. Inherently, each G_q^* forms a connected knowledge graph (KG) [52]. Such KGs can be retrieved by iteratively traversing links from one entity to another. Take $G_q^* = \{\delta_1', \delta_2, \delta_3'\}$ in Table 1 as an example. It is composed of two edited facts δ_1', δ_3' and one unedited fact δ_2. We can observe that the question entity: (*Misery*) is the head entity h_1 in $\delta_1' = \{h_1, t_1, t_1'\}$, and the edited tail entity t_1': (*Richard Dawkins*) is also the head entity h_2 in the next fact $\delta_2 = \{h_2, r_2, t_2\}$. Moreover, for each subsequent fact in the chain, its head entity is always the tail entity of the previous fact. By effectively retrieving the KG that represents the fact chain G_q^*, we are able to capture all the edited factual triplets $\Delta_q = \{\delta_i', \delta_j', ...\delta_k'\}$. In light of this, we define our retrieval-augmented editing as:

$$f_\theta'(q) = f_\theta(T_e(q, G_q^*)), \tag{1}$$

where we give an example of such editing in Figure 2. In the next section, we will introduce the detailed strategy of retrieving G_q^*, where we first propose a mutual information-based retrieval strategy to extract facts needed to answer the target question (Section 3.2). Then, we propose a pruning method to delete irrelevant facts from the initial retrieval result (Section 3.3).

3.2 Edited Facts Retrieval via Maximizing MI

We first construct a knowledge graph that connects different facts. Then, we introduce our proposed retrieval objective of extracting relevant subgraphs given input questions.

3.2.1 External Knowledge Graph for Subgraph Retrieval. According to our previous discussion, we aim to retrieve the fact chain G_q^* for model editing. Additionally, it is worth noting that G_q^* consists of both edited and unedited facts. However, the unedited facts are not included in our edited fact bank Δ by default. To effectively incorporate both types of facts into our retrieval process, we propose integrating all edited facts into an external knowledge graph $\mathcal{G}$. By selecting a comprehensive KG such as WikiData [33], the new graph $\mathcal{G}^*$ will encompass both unedited and edited facts. It complements our edited fact bank Δ and connects different entities. Besides, the external knowledge graph provides extra factual knowledge that can enhance language to output correct answers.

Specifically, given the edits $\Delta = \{\delta_1', ..., \delta_n'\}$ and an external $\mathcal{G}$, we consider two types of operations to combine them. **(1) Modifying existing facts:** If the original fact appears in $\mathcal{G}$, i.e., $(h, r, t) \in \mathcal{G}$, we will modify the KG according to the edits, so $\mathcal{G}^* = (h, r, t') \cup$

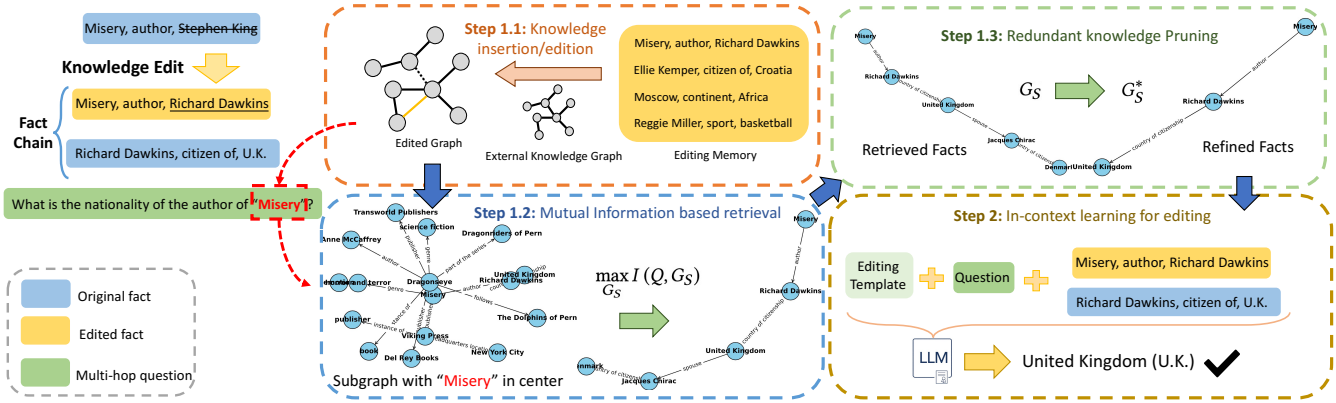


Figure 2: The overall framework of our retrieval-augmented in-context model editing method.

$\mathcal{G} \setminus (h, r, t)$. **(2) Adding new facts:** If the original fact does not appear in $\mathcal{G}$, i.e., $(h, r, t) \notin \mathcal{G}$, then we append the modified fact to the KG, so $\mathcal{G}^* = (h, r, t') \cup \mathcal{G}$. Next, given a question q, we retrieve a subgraph G_S from $\mathcal{G}^*$, so that $G_S \subset \mathcal{G}^*$. Our goal is to ensure that G_S contains fact chains of q, i.e., $G_q^* \subseteq G_S$.

3.2.2 Mutual Information based Retrieval Objective. For effective editing, the retrieved subgraph G_S must share relevant information with the question. Therefore, we define the objective of subgraph retrieval as maximizing the mutual information (MI) between the subgraph and a set of questions Q whose answers require editing. The objective is formalized as below, where the theoretical justification is provided in Section 3.4:

$$\max_{G_S} I(Q; G_S) = H(Q) - H(Q \mid G = G_S). \tag{2}$$

Given a fixed question set Q, its Shannon entropy $H(Q)$ is constant. Therefore, maximizing the mutual information $I(Q; G_S)$ is equivalent to minimizing the conditional entropy $H(Q \mid G = G_S)$. Thus, we optimize the following objective:

$$\max_{G_S} I(Q; G_S) = \min_{G_S} H(Q \mid G = G_S) \tag{3}$$

$$= \max_{G_S} \sum_{q \in Q} p(q|G = G_S) \log_2 p(q|G = G_S). \tag{4}$$

In practice, quantifying $p(q|G = G_S)$ is challenging due to its computational complexity. This is because there are numerous subgraph candidates G_S within the entire knowledge graph, making it prohibitively expensive to exhaustively search for the optimal one. To circumvent this issue, we first replace the intractable term $p(q|G = G_S)$ with $\frac{p(q, G=G_S)}{p(G=G_S)}$. Then, suppose we consider one question each time, where $Q = q$, the objective is reformulated as:

$$\max_{G_S} \frac{p(q, G = G_S)}{p(G = G_S)} \log_2 \frac{p(q, G = G_S)}{p(G = G_S)}. \tag{5}$$

In the following, we will discuss how to estimate probability $p(q, G = G_S)$ and $p(G = G_S)$ efficiently.

3.2.3 Probabilities Estimation. We propose to compute probabilities by leveraging the **next-word prediction capability of LLMs**. Given that the fact chain forms a tail-to-head connected knowledge graph, our extracted subgraph G_S can be represented as $G_S = (h_1, r_1, t_1, ..., h_n, r_n, t_n)$, where h_i and t_i are nodes, r_i is the edge, and n is the number of retrieved triplets. Thus, we can estimate $\frac{p(q, G=G_S)}{p(G=G_S)}$ as:

$$\frac{p(q, G = G_S)}{p(G = G_S)} = \frac{p(r_1, t_1, h_2, r_2, t_2, ..., h_n, r_n, t_n|q, h_1)}{p(r_1, t_1, h_2, r_2, t_2, ..., h_n, r_n, t_n|h_1)} \cdot \frac{p(q, h_1)}{p(h_1)}. \tag{6}$$

Specifically, for the term $p(r_1, t_1, h_2, r_2, t_2, ...|q, h_1)$, we can further decompose it into following form:

$$\begin{aligned} &p(r_1, t_1, h_2, r_2, t_2, ..., h_n, r_n, t_n|q, h_1) \\ &= p(t_1, h_2, r_2, t_2, ..., h_n, r_n, t_n|q, h_1, r_1) \cdot p(r_1|q, h_1). \end{aligned} \tag{7}$$

This decomposition allows us to initially focus on estimating the $p(r_1|q, h_1)$. Specifically, the head entity h_1 is determined if q is given, since we assume h_1 is mentioned in question q. Candidate relations for r_1 can also be selected from the edited KG. Practically, we can estimate the probability $p(r_1|q, h_1)$ for each candidate relation using an auto-regressive language model f_ϕ [27, 38]:

$$p(r_1|q, h_1) \approx \prod_{i=1}^{|r_1|} f_\phi(w_{r_1}^{(i)} | w_q^{(1)}, ..., w_q^{(|q|)}, w_{h_1}^{(1)}, ..., w_{h_1}^{(|h_1|)}, w_{r_1}^{(1)}, ..., w_{r_1}^{(i-1)}), \tag{8}$$

where f_ϕ is the predicted word probability, and w_q, w_{h_1}, w_{r_1} denote the words in question q, head entity h_1, and relation r_1, respectively. We can employ open-source LLMs like GPT-2 [27] for this estimation. Please note that, the model f_θ being edited does not need to be the same model used for probability estimation, making our method applicable even for editing proprietary LLMs. With a specific input context $\{q, h_1\}$, the language model will assign different probabilities to each relation based on its contextual understanding and reasoning ability.

Then, $p(t_1, h_2, r_2, t_2, ...|q, h_1, r_1)$ can be further decompose into $p(h_2, r_2, t_2, ...|q, h_1, r_1, t_1) \cdot p(t_1|q, h_1, r_1)$. In our case, we assume $p(t_1|q, h_1, r_1) = 1$, since one relation usually only corresponds to one tail entity. When there are multiple tail entities, we find the assumption still works well empirically. So, $p(h_2, r_2, t_2, ...|q, h_1, r_1, t_1)$ can be decomposed into $p(r_2, t_2, ...|q, h_1, r_1, t_1, h_2) \cdot p(h_2|q, h_1, r_1, t_1)$. Additionally, since the tail entity in one fact becomes the head entity in the subsequent fact, we can also have $p(h_2|q, h_1, r_1, t_1) = 1$. Thus, we can have $p(t_1, h_2, r_2, t_2, ...|q, h_1, r_1) = p(r_2, t_2, ...|q, h_1, r_1, t_1, h_2)$. This is a nice property that helps us iteratively decompose this

intractable probability term. By iteratively applying the aforementioned step for n times, we can compute the conditional probability of all subgraphs within an n-hop distance from the question entity. The final estimation can be expressed as:

$$\begin{aligned} & p(r_1, t_1, h_2, r_2, t_2, ..., h_n, r_n, t_n | q, h_1) \\ = & \, p(r_n | q, h_1, r_1, t_1, ..., h_{n-1}, r_{n-1}, t_{n-1}, h_n) \cdot \\ & \, p(r_{n-1} | q, h_1, r_1, t_1, ..., h_{n-2}, r_{n-2}, t_{n-2}, h_{n-1}) \cdot \\ & \, ... \cdot \\ & \, p(r_2 | q, h_1, r_1, t_1, h_2) \cdot p(r_1 | q, h_1). \end{aligned} \tag{9}$$

Till now, we have decomposed $p(r_1, t_1, h_2, ...|q, h_1)$ in Equation (6) into the product of conditional probabilities of predicting different relations within the n-hop subgraph. This nice property ensures the selection of the subgraph will only be determined by relation probability, which is free from the interference of any potential edited tail entity. Similarly, we can decompose the denominator term $p(r_1, t_1, h_2, ...|h_1)$ into:

$$\begin{aligned} & p(r_1, t_1, h_2, r_2, t_2, ..., h_n, r_n, t_n | h_1) \\ = & \, p(r_n | h_1, r_1, t_1, ..., h_{n-1}, r_{n-1}, t_{n-1}, h_n) \cdot \\ & \, p(r_{n-1} | h_1, r_1, t_1, ..., h_{n-2}, r_{n-2}, t_{n-2}, h_{n-1}) \cdot ... \cdot p(r_1 | h_1). \end{aligned} \tag{10}$$

Then, for the last term $p(q, h_1)/p(h_1)$ in Equation (6), based on Bayes' theorem, we can transform it into $p(q, h_1)/p(h_1) = p(q|h_1)$, which is a constant value given a specific question q. We can also apply model f_ϕ to estimate this conditional probability. Now, since we are able to estimate every term in Equation (5) and (6), we can effectively identify the subgraph that yields the maximum Mutual Information. Additionally, we utilize beam search [29] to expedite the computational process, eliminating the necessity for exhaustively traversing all connected nodes. In this work, we treat n as a hyperparameter since the number of hops required to answer a question is unknown in advance. To ensure thorough exploration, n is assigned a large value. However, this approach will also introduce irrelevant information in the retrieved subgraph G_S, which can potentially mislead the language model to hallucinate and generate undesired answers [17, 18]. In the next section, we will discuss how to mitigate this problem.

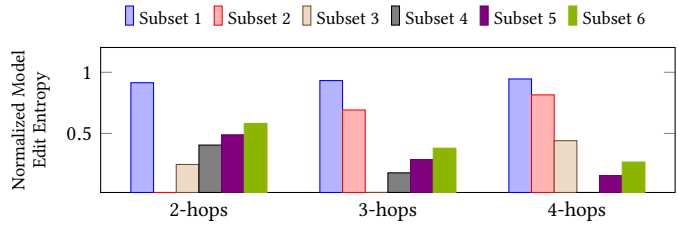
3.3 Uncertainty-based Redundant Fact Pruning

This section introduces a pruning method, which utilizes model output uncertainty, to eliminate redundant facts from G_S.

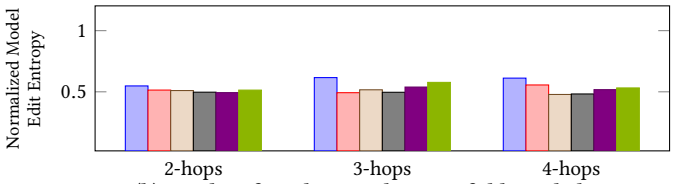
3.3.1 Editing Uncertainty. We define editing uncertainty as the uncertainty of the output generated by large language models. Formally, the output uncertainty is quantified by Shannon entropy:

$$H(Y|X = x) = -\sum_y p(y|x) \log_2 p(y|x), \tag{11}$$

where y represents each possible answer generated by the language model, and $x = \{q, G_S\}$ is the model input composed of the question q and facts G_S. A higher entropy value $H(Y|X = x)$ means less confidence in the answer, reflecting greater uncertainty. In contrast, a lower entropy value indicates higher confidence and less uncertainty. Ideally, if input facts G_S are exactly the edited question fact chain G_q^*, i.e., $G_S = G_q^*$, then the model output should exhibit maximum confidence with minimal entropy, since G_q^* contains the precise knowledge to answer question q. In the next part, we conduct empirical experiments to verify this assumption.



(a) Fact chain G_q^* with redundant knowledge.



(b) Random fact chain without useful knowledge.

Figure 3: Distribution of normalized model editing entropy with different fact subsets as input. A lower normalized entropy indicates that the model is more confident in answering the question with the given facts. "Subset 1" includes the first fact $\{\delta_1\}$, "Subset 2" includes the first two facts $\{\delta_1, \delta_2\}$, and so on. Figure 3a shows that the entropy is significantly lower if the subset contains exactly the entire fact chain of the question (e.g., Subset 2 has low entropy for 2-hop questions).

In our experiments, we choose GPT-J (6B) [34] as the base language model. We select 1000 instances for each of the 2, 3 and 4-hop questions from the MQUAKE-CF dataset [52] for testing. The MQUAKE-CF dataset comprises multi-hop questions that are based on real-world facts, where the edited facts are counterfactual, meaning they do not exist in actual real-world scenarios. An example of such a question with an edit is provided in Table 1.

Our experiment seeks to identify the fact set G'_S that, when used as model input, yields the lowest output entropy (i.e., minimal editing uncertainty). Our first step is to construct different fact set candidates. We begin with the first fact δ_1 in the fact chain G_q^* as our initial fact set G'_S. Then, we add each subsequent fact from the chain until the G'_S encompasses the entire fact chain G_q^*. After that, we insert unrelated facts $\hat{\delta}$ into the set. In this experiment, the process is repeated until G'_S contains six elements, where we build a prefix set $\bar{G}_q$ with all the six subsets G'_S. For example, for a 4-hop question, we have $\bar{G}_q = \{\{\delta_1\}, \{\delta_1, \delta_2\}, \{\delta_1, \delta_2, \delta_3\}, ..., \{\delta_1, \delta_2, \delta_3, \delta_4, \hat{\delta}_5, \hat{\delta}_6\}\}$, where $\delta_1, \delta_2, \delta_3, \delta_4 \in G_q^*$ and $\hat{\delta}_5, \hat{\delta}_6$ are two irrelevant facts. Finally, we conduct in-context editing using each subset G'_S from $\bar{G}_q$ with editing template T_e:"*Given fact:* $\{G'_S\}$, $\{q\}$*?*". In our experiment, we consider model output y to be each of the next predicted word. We report the entropy over all the words in the vocabulary as the editing uncertainty.

The editing uncertainty with different subsets is listed in Figure 3a. For comparison, we also report the editing uncertainty with random facts selected from Wikidata, as shown in Figure 3b. Our observations reveal a phenomenon: the language model produces

answers with much lower entropy when G'_S is equal to the ground-truth fact chain G^*_q. If G'_S presents redundant facts or insufficient facts, the entropy will increase. Meanwhile, if the LLM is fed with random facts, the entropy level also remains consistently high. **This observation justifies the use of entropy as the indicator of whether G'_S contains the correct facts for LLM inference.**

3.3.2 Knowledge Pruning with Editing Uncertainty. Since incorporating the most relevant facts will result in the lowest entropy, we propose to utilize this finding for knowledge pruning. Specifically, we first follow Section 3.2 to retrieve a knowledge graph G_S containing n triplets for question q: $G_S = \{\delta_1, \delta_2, ..., \delta_n\}$, where n is a sufficiently large number, and G_S could contain redundant knowledge. Then, to remove redundant knowledge, we first build the prefix sets $\bar{G}_q$ for target question q based on the retrieved graph G_S. Then, we can obtain the pruned fact set G^*_S using the objective:

$$G^*_S = \underset{G'_S \in \bar{G}_q}{\operatorname{argmin}} - \sum_y p(y|T_e(q, G'_S)) \log_2(p(y|T_e(q, G'_S))). \quad (12)$$

Finally, we can apply G^*_S as our retrieved fact chain for the in-context learning introduced in Section 3.1 to conduct editing.

3.4 Theoretical Justification

In this subsection, we theoretically justify that the facts collected by our retrieval objective Eq. (2) are effective in performing model editing with in-context learning. To begin with, we discuss what kinds of input can effectively activate in-context learning. Then, we explore how to build such effective input for model editing.

Our proposed editing method relies on the in-context learning ability of LLMs. In the following, we provide an analysis of how in-context learning can be effectively triggered. Theoretically, the text generation process of a language model can be understood as a Hidden Markov Model [3, 40]. The model initially selects a concept $\theta_c \in \Theta$ from a set of underlying concepts denoted as Θ, and then samples a sequence of words based on the chosen concept. Based on that, the in-context learning can be written as $p(y|S, x) = \int_{\theta_c \in \Theta} p(y|S, x, \theta_c) p(\theta_c|S, x) d\theta_c$, where S denotes in-context prompt and x denotes query. Existing research has theoretically proven that the condition to activate in-context learning is when there is a shared latent concept θ_c between prompt text S and the input query x. More discussions can be found in [40].

Motivated by the above analysis, as in-context prompt S is the edited knowledge in our design, we seek to include the edited knowledge that shares the same latent concept θ_c as question q. Ideally, this will activate in-context learning for effective model editing. Formally, we can define such knowledge graph as

$$G_S = \underset{G \in \mathcal{G}}{\operatorname{argmax}} I(G; \theta_c), \quad (13)$$

where θ_c is the latent concept used to generate question q, and we use mutual information $I(G; \theta_c)$ to quantify the share information. However, this is a non-trivial task since concept θ_c is an intractable hidden variable. To address this issue, we propose obtaining the target knowledge graph that maximizes the lower bound of such an objective. Specifically, we can have the following theorem:

Theorem 1. *Given retrieved graph $G_S \in \mathcal{G}$, the latent concept θ_c, and the question q sampled conditioned on concept θ_c, there exists a mutual information inequality:*

$$I(G_S; \theta_c) \geq I(G_S; q). \quad (14)$$

Theorem 1 shows that we can maximize the mutual information between the selected knowledge graph G_S and question concepts θ_c by maximizing the mutual information between the selected graph G_S and the question q itself. In this way, the in-context learning ability of LLMs would be effectively triggered. When we apply such knowledge as the prompt, we can effectively conduct the in-context editing. The proof of Theorem 1 is in Appendix A.

4 Experiments

We conduct experiments to answer the following questions. **Q1**: How effective is RAE in editing LLM output? **Q2**: How does our retrieval strategy perform compared to other retrieval methods? **Q3**: Does our proposed pruning technique remove redundant facts from the retrieved facts? **Q4**: Does RAE work for propriety LLMs?

4.1 Experiment Settings

4.1.1 Language Models. We evaluate RAE across various kinds of language models in different sizes and families, including GPT-2 (1.5B) [27], GPT-J (6B) [34], Falcon (7B) [2], Vicuna (7B) [5], and Llama2-chat (7B) [31]. Among them, GPT-2, GPT-J, and Falcon are pre-trained language models without instruction tuning [6, 26], while Vicuna is an instruction-tuned variation of Llama1 [32] and Llama2-chat is the instruction-tuned version of Llama2. Instruction-tuned models (Vicuna and Llama2-chat) are expected to better follow the instructions in the prompt compared to native pre-trained models (GPT-2, GPT-J, and Falcon). We include both kinds of models to verify the effectiveness of the proposed methods.

4.1.2 Editing Baselines. For comparison, we consider three kinds of model editing methods: (1) Model weight updating methods: Fine-tuning [53] edits the model weights by language modeling the edited knowledge. ROME [19] and MEMIT [20] focus on identifying and updating particular neurons associated with the knowledge that needs editing. (2) Auxiliary models methods: SEARC [22] trains an extra language model to store updated knowledge, and it switches to the auxiliary model when answering questions relevant to the edited facts. (3) RAG-based methods: Mello [52] and DeepEdit [36] represent cutting-edge editing methods for multi-hop questions, employing multi-round conversations to edit model outputs. Additionally, the Subgraph Retriever (SR) [47] introduces an advanced knowledge retrieval approach for multi-hop question-answering tasks. We adapt their retrieval method as a baseline.

4.1.3 Implementation Details. We evaluate our editing method on the MQUAKE-CF and MQUAKE-T datasets from [52] and Popular datasets from [7]. The MQUAKE-CF (M-CF) comprises counterfactual editing instances in 2, 3, and 4-hop questions, totaling 3000 edits. The MQUAKE-T dataset (M-T) features temporal editing examples in 2 and 3-hop questions, with a total of 1868 edits. Additionally, the Popular dataset contains counterfactual editing in 2-hop questions, comprising 274 edits. Following previous work [50, 52], we leverage relevant cases from the MQUAKE-CF-9k dataset to craft prompt templates for both baselines and our method. We evaluated our editing method using the multi-hop edited accuracy metrics

Table 2: Edited accuracy (%) on multi-hop question editing datasets.

		Editing Methods							
Language Models	Datasets	Fine Tune	ROME	MEMIT	SEARC	Mello	DeepEdit	Subgraph Retriever	RAE(ours)
GPT-2 (1.5B)	M-CF	3.8	1.7	2.3	4.0	0.0	0.0	21.9	**62.8**
	M-T	5.8	6.4	1.6	2.7	0.0	0.0	20.3	**61.8**
	Popular	6.2	4.3	2.9	1.1	0.0	0.0	26.7	**47.1**
GPT-J (6B)	M-CF	7.7	7.6	8.1	6.8	15.3	9.3	36.2	**69.3**
	M-T	3.1	4.1	10.6	2.8	36.7	19.6	51.2	**63.9**
	Popular	6.8	7.5	4.4	1.3	12.8	6.6	45.8	**49.6**
Falcon (7B)	M-CF	5.6	1.7	2.3	7.9	10.7	10.8	40.1	**66.8**
	M-T	17.2	7.3	1.6	4.5	51.5	31.7	56.1	**61.6**
	Popular	2.1	4.0	1.1	3.0	8.1	9.5	43.0	**50.0**
Vicuna (7B)	M-CF	4.8	8.4	7.6	7.9	10.2	11.4	39.4	**67.2**
	M-T	23.1	5.0	1.7	4.5	51.7	40.4	58.6	**63.2**
	Popular	4.0	3.8	2.4	3.0	7.7	8.2	29.5	**36.1**
Llama2 (chat) (7B)	M-CF	5.4	6.3	3.8	7.9	20.7	11.2	45.7	**69.1**
	M-T	17.1	8.7	1.7	4.5	49.4	37.9	63.1	**66.2**
	Popular	5.2	13.8	4.9	3.0	13.5	11.1	41.9	**51.4**

Table 3: Multi-hop facts retrieval precision (%) comparison.

MQUAKE-CF							
Question Type		2-hops		3-hops		4-hops	
Category	Retrieval	P@1	P@2	P@1	P@3	P@1	P@4
Embedding	KG Link	52.7	28.7	18.2	3.7	14.0	0.0
	QR	62.3	7.7	14.7	0.0	12.3	0.0
	Mello(Llama2)	84.3	80.0	80.7	42.3	83.3	25.7
Probability	SR(GPT-2)	77.7	50.3	67.3	25.3	65.0	20.0
	SR(Llama2)	78.3	55.7	79.7	37.0	69.3	28.7
Mutual Information	RAE(GPT-2)	83.0	66.3	77.3	41.0	80.3	43.7
	RAE(GPT-J)	83.0	69.7	81.3	53.7	82.7	54.0
	RAE(Falcon)	82.3	70.7	72.3	44.3	81.7	47.3
	RAE(Vicuna)	81.0	66.7	79.3	50.3	85.0	50.0
	RAE(Llama2)	82.7	69.3	84.0	49.3	82.0	47.0

from [36, 52]. The results, which reflect the accuracy when all edits are applied in one batch, are reported in Table 2.

4.2 Editing Performance Evaluation

To answer **Q1**, we assess our model editing method across various language models, compared against different baseline methods. Our key observations from Table 2 are: **(1)** Our RAE outperforms all others in three datasets across five language models when conducting thousands of edits at the same time. This superior performance primarily stems from our novel MI-based retrieval objective and an effective pruning strategy. Our design can also seamlessly integrate an external knowledge graph, which effectively links all edited facts, thereby facilitating the multi-hop editing process. **(2)** RAG-based methods generally show better performance than other methods. Specifically, Mello and DeepEdit demonstrate good performance on the M-T dataset with models larger than 6B. However, they underperform on the M-CF and popular datasets and fail with GPT-2, likely due to GPT-2's inability to follow their complex prompts, resulting in no output. Additionally, the subgraph retriever ranks second in performance. However, its probability-driven retrieval method occasionally fails to fetch relevant facts, leading to reduced effectiveness. **(3)** Other methods generally show lower performance across all language models, which aligns with the findings from [52].

4.3 Retrieval Performance Evaluation

To answer **Q2**, we assess the effectiveness of our mutual information-based retrieval method for multi-hop question-answering tasks.

4.3.1 Retrieval Baselines. We consider three embedding-based and one probability-based methods as baselines.

(1) Embedding-based methods mostly utilize dense retrieval to fetch relevant corpus from vector databases. Here, we adapt them to fit our multi-hop knowledge graph retrieval task. We employ a cutting-edge retriever, the Contriever [15], to encode all edited facts into embeddings, and then cache these embeddings for similarity search. In addition to Mello, we include two representative multi-hop retrieval methods as follows:

- **KG Link** [9, 41] is a straightforward strategy, which identifies the query entity and its linked entities as candidates to find the one that is the most similar to the original question. This process is repeated K times to retrieve the entire fact chain.
- **Question Reform (QR)** [23, 42] appends the retrieved entity to the previous query for the next hop retrieval. This design is motivated by simulating a reasoning path where the retrieved fact at each hop is viewed as an intermediate reasoning result.

(2) Probability-based method (i.e., subgraph retriever [47]) retrieves the subgraph $G_S \in \mathcal{G}^*$ that maximizes the conditional probability $p(G_S|q)$, where q is the input multi-hop question.

4.3.2 Implementation Details. Following [28], we select 300 cases for each of the 2, 3, and 4-hop questions from the M-CF dataset. For each type, we report the retrieval precision scores in Table 3. Specifically, we use the metric $Precision@K$, which calculates the proportion of relevant facts within the top K results [28]. Here, $Precision@K = |\{\text{relevant facts}\}|/K \times 100\%$, abbreviated as $P@K$. For this study, a k-hop question has k facts as its fact chain. A fact is considered relevant if it is part of the fact chain used to answer a multi-hop question.

4.3.3 Results. We have the following observations: **(1)** Our proposed mutual information-based retrieval method demonstrates

Table 4: Edited accuracy (%) with (w) or without (w/o) pruning.

Dataset		MQUAKE-CF				
Type	Strategy	GPT-2	GPT-J	Falcon	Vicuna	Llama2 (chat)
2-hops	w/o Pruning	63.0	63.7	65.2	63.8	70.1
	w/ Pruning	73.3	75.5	74.5	73.5	75.8
	Gain	16.3%↑	18.5%↑	14.3%↑	15.2%↑	8.1%↑
3-hops	w/o Pruning	43.1	53.8	55.6	55.0	60.3
	w/ Pruning	53.2	65.4	62.1	62.7	65.8
	Gain	23.4%↑	21.6%↑	11.7%↑	14.0%↑	9.1%↑
4-hops	w/o Pruning	49.9	58.8	55.2	61.5	61.6
	w/ Pruning	61.9	66.9	62.9	65.5	65.8
	Gain	24.0%↑	13.8%↑	13.9%↑	6.5%↑	6.8%↑

excellent performance across various LLMs in multi-hop fact extraction. We also find its success with relatively small language models, such as GPT-2, showing strong generalization ability. **(2)** In contrast, traditional embedding-based methods (KG Link and Question Reform) underperform in this multi-hop fact retrieval challenge. Their limitation mainly lies in failing to comprehend the complex interplay between multiple relations in a question. Thus, it hinders the extraction of target facts from the extensive knowledge base. **(3)** Mello demonstrates the efficacy of decomposing multi-hop questions into single-hop questions for this multi-hop facts retrieval task. Notably, its performance drops significantly with an increasing number of hops, probably because it becomes much more challenging for language models to perform question decomposing. **(4)** Probability maximization-based methods outperform traditional embedding methods, while there is still a gap compared to ours, probably because the predicted most probable fact may not be the most necessary one for answering a specific question.

4.4 Ablation Studies on Pruning Strategy

To answer **Q3**, we verify that our proposed pruning strategies are beneficial to multi-hop editing tasks. To simulate the situation where the retrieved facts contain redundant information, we conduct our experiment by always retrieving 2 additional facts over the facts needed by the original question. That is, given a k-hop question, we set the total number of retrieved facts as $n = k + 2$.

Table 4 reports the edited accuracy of RAE working with or without the pruning strategy, and we draw the following conclusions: **(1)** The proposed pruning technique significantly enhances the performance of model editing, demonstrated by achieving an average accuracy improvement of 14.5% across various language models. **(2)** We observe a more profound improvement for smaller language models on complex questions, while this benefit to larger language models is relatively less significant. In particular, the performance improvement of GPT-2 on the 4-hop questions reaches 24.0%, while for Llama-2 it is just 6.8%. This observation suggests that larger models are more robust to redundant information.

4.5 Editing Performance on Proprietary LLMs

To answer **Q4**, we apply the proposed RAE to edit proprietary language models, where we can only access the model via APIs, such as ChatGPT [24]. In such cases, RAE utilizes a different lightweight language model to perform relevant facts retrieval and pruning as introduced in Section 3.2 and Section 3.3. Then, the obtained facts

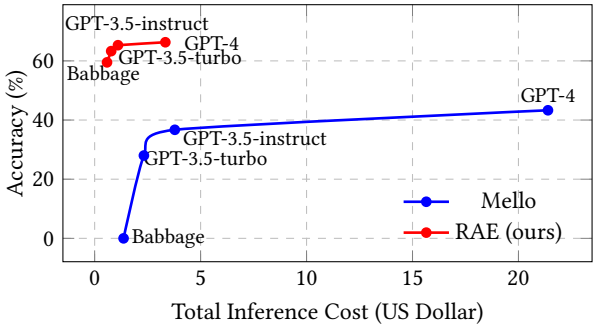


Figure 4: Editing performance and inference cost over different proprietary models.

will be fed into the proprietary models to perform in-context editing. We evaluate our proposed editing method on *GPT-babbage-002*, *GPT-3.5-turbo-0613*, *GPT-3.5-instruct*, and *GPT-4-0613* models [25]. We use GPT-2 (1.5B) as our retrieval model. We report the edited accuracy and total editing cost of our method for 300 randomly selected cases (MQUAKE-CF) in Figure 4. The editing cost includes the total fees of calling APIs and the cost of running GPT-2 for knowledge retrieval on rented GPUs. We also report the results of Mello [52] for comparison, where only API fees are counted.

In Figure 4, we first observe that Babbage has 0.0% edited accuracy by using Mello, showing its ineffectiveness in editing the un-instruction tuned proprietary language model. This is expected since Mello relies on the conversational ability of language models to decompose multi-hop questions. In contrast, RAE is effective in editing all these proprietary models with a remarkably lower cost than Mello. In particular, RAE improves Mello for editing GPT-4 with almost 20% edited accuracy by only costing around its 15% budget. This highlights the benefit of utilizing the inherent language modeling ability instead of the instruction following ability to perform knowledge retrieval for multi-hop question answering.

4.6 Performance with Different Batch Sizes

In this section, we evaluate editing performance with different editing batch sizes. Specifically, we set the editing instances in sizes of 1, 10, 100, and 1000 cases for 2-hop questions. We choose Mello [52] as our baseline. The result is presented in Figure 5. We can observe that, in both the Vicuna and Llama2 models, RAE's accuracy remains stable across different editing instances, whereas Mello's accuracy significantly declines with increasing instances.

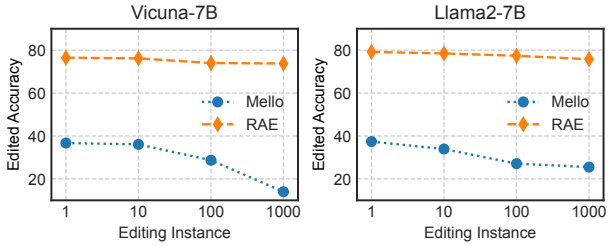
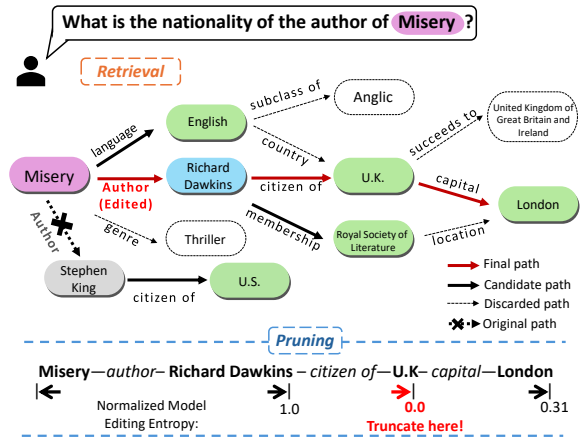


Figure 5: Edited accuracy with different edit batch sizes.

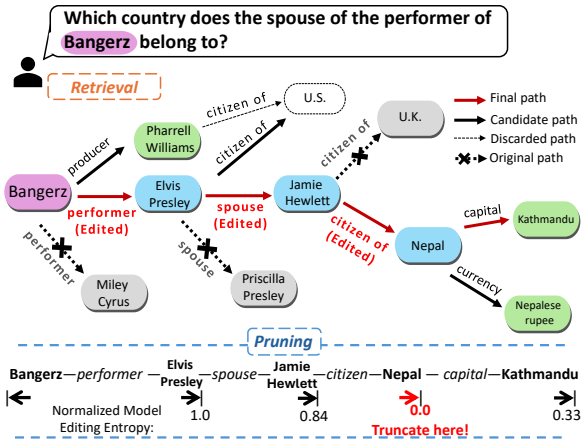
4.7 Case Study

In Figure 6, we present two cases from the M-CF dataset to demonstrate the retrieval process on a knowledge graph and the pruning

process of the retrieved facts. For visualizations, the red, black, and dotted lines represent the final, candidate, and discarded paths in the knowledge graph with beam search, reflecting the decision-making process of our retrieval design. In Figure 6a, a potential path *Misery–language–>English–country–>U.K.* is discarded even though it can lead to the correct answer (U.K. in this case). This is because it does not share the information that is needed to answer the target question, resulting in a low MI score. In Figure 6b, even though this question is associated with three edited facts, our method can successfully locate all of them, demonstrating robustness over multi-hop questions. In both cases, our pruning strategy successfully truncates the retrieved fact chain to only contain necessary facts relying on the normalized model editing entropy.



(a) A two-hop question example (CaseID 2).



(b) A three-hop question (CaseID 1143).

Figure 6: Case studies for edited facts retrieval and pruning. The retrieval process involves the beam search, starting from the query entity and navigating through the knowledge graph with two beams. At each entity hop, the two primary candidate edges are highlighted in bold, while others are discarded and marked with dashed lines. The beam search result with the highest MI score is emphasized in red.

5 Related Work: Model Editing

Existing editing methods can be categorized into the following kinds: First, methods that alter model parameters, including fine-tuning [53], locate-then-edit [19, 20], and meta-learning [21, 48], are prone to catastrophic forgetting issue [7, 8, 46], where the previously encoded knowledge could be lost after editing. They also struggle with using updated language for knowledge reasoning, which can lead to severe hallucinations. [4, 10, 52]. While these methods perform well in single-hop editing, they are less effective in multi-hop scenarios. [52]. Second, methods that depend on training auxiliary models also fall short in these scenarios. The auxiliary models are usually smaller language models, which lack the necessary reasoning capability to infer correct answers [22]. In contrast, a third category of methods, based on Retrieval-Augmented Generation (RAG), modifies model outputs in a more effective manner [11, 36, 44, 52]. These methods integrate updated knowledge directly into the model input and edit LLMs through an editing prompt. [7, 50]. RAG-based approaches offer notable benefits as they are resistant to catastrophic forgetting, and allow for on-the-fly editing. [14, 35]. For multi-hop editing, existing RAG-based methods perform well on powerful proprietary LLMs like GPT-3.5, but show significant degradation on less robust LLMs such as Llama2-7b, particularly as the number of editing instances increases. In contrast, our RAE can maintain good editing performance even with relatively small LLMs and large editing batch sizes.

6 Conclusion

We propose a novel LLM editing framework for multi-hop QA that employs mutual information maximization for fact retrieval and a self-optimizing technique to prune redundant data. This effectively tackles the integration of real-time knowledge updates in LLMs. Our extensive evaluations confirm the framework's ability to enhance the accuracy of LLM responses, marking significant progress in model editing and dynamic knowledge integration research.

Acknowledgments

The work is, in part, supported by NSF (#IIS-2223768). The views and conclusions in this paper are those of the authors and should not be interpreted as representing any funding agencies.

A Theoretical Justification

Theorem 1. *Given retrieved graph $G_S \in \mathcal{G}$, the latent concept θ_c, and the question q sampled conditioned on concept θ_c, there exists a mutual information inequality: $I(G_S;\theta_c) \geq I(G_S;q)$.*

Proof. Consider latent concepts θ_c is inferred from a knowledge graph G_S, and a question q is then generated based on this concept. The following Markov chain: $G_S \rightarrow \theta_c \rightarrow q$ exists, indicating that q is conditionally independent to G_s. According to the chain rule for mutual information, we can have:

$$I(G_S;\theta_c,q) = I(G_S;q)+I(G_S;\theta_c|q) = I(G_S;\theta_c)+I(G_S;q|\theta_c). \quad (15)$$

Given that the question q is conditionally independent of the graph G_S, the term $I(G_S;q|\theta_c)$ equals zero. Therefore, we are left with: $I(G_S;\theta_c) \geq I(G_S;q)$. Equality holds precisely when $I(G_S;q|\theta_c)$ is zero, meaning that the graph G_S provides no additional information about q once θ_c is known. □

References

[1] [n. d.]. Rishi Sunak. https://en.wikipedia.org/wiki/Rishi_Sunak Accessed: 2024-05-16.

[2] Ebtesam Almazrouei, Hamza Alobeidli, Abdulaziz Alshamsi, Alessandro Cappelli, Ruxandra Cojocaru, Mérouane Debbah, Étienne Goffinet, Daniel Hesslow, Julien Launay, Quentin Malartic, et al. 2023. The falcon series of open language models. *arXiv preprint arXiv:2311.16867* (2023).

[3] Leonard E Baum and Ted Petrie. 1966. Statistical inference for probabilistic functions of finite state Markov chains. *The annals of mathematical statistics* 37, 6 (1966), 1554–1563.

[4] Canyu Chen, Baixiang Huang, Zekun Li, Zhaorun Chen, Shiyang Lai, Xiongxiao Xu, Jia-Chen Gu, Jindong Gu, Huaxiu Yao, Chaowei Xiao, Xifeng Yan, William Yang Wang, Philip Torr, Dawn Song, and Kai Shu. 2024. Can Editing LLMs Inject Harm? arXiv:2407.20224 [cs.CL] https://arxiv.org/abs/2407.20224

[5] 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. https://lmsys.org/blog/2023-03-30-vicuna/

[6] Hyung Won Chung, Le Hou, Shayne Longpre, Barret Zoph, Yi Tay, William Fedus, Eric Li, Xuezhi Wang, Mostafa Dehghani, Siddhartha Brahma, et al. 2022. Scaling instruction-finetuned language models. *arXiv preprint arXiv:2210.11416* (2022).

[7] Roi Cohen, Eden Biran, Ori Yoran, Amir Globerson, and Mor Geva. 2023. Evaluating the ripple effects of knowledge editing in language models. *arXiv preprint arXiv:2307.12976* (2023).

[8] Damai Dai, Li Dong, Yaru Hao, Zhifang Sui, Baobao Chang, and Furu Wei. 2021. Knowledge neurons in pretrained transformers. *arXiv preprint arXiv:2104.08696* (2021).

[9] Rajarshi Das, Ameya Godbole, Dilip Kavarthapu, Zhiyu Gong, Abhishek Singhal, Mo Yu, Xiaoxiao Guo, Tian Gao, Hamed Zamani, Manzil Zaheer, et al. 2019. Multi-step entity-centric information retrieval for multi-hop question answering. In *Proceedings of the 2nd Workshop on Machine Reading for Question Answering*. 113–118.

[10] Zorik Gekhman, Gal Yona, Roee Aharoni, Matan Eyal, Amir Feder, Roi Reichart, and Jonathan Herzig. 2024. Does Fine-Tuning LLMs on New Knowledge Encourage Hallucinations? *arXiv preprint arXiv:2405.05904* (2024).

[11] Hengrui Gu, Kaixiong Zhou, Xiaotian Han, Ninghao Liu, Ruobing Wang, and Xin Wang. 2024. PokeMQA: Programmable knowledge editing for Multi-hop Question Answering. arXiv:2312.15194 [cs.CL] https://arxiv.org/abs/2312.15194

[12] Jia-Chen Gu, Hao-Xiang Xu, Jun-Yu Ma, Pan Lu, Zhen-Hua Ling, Kai-Wei Chang, and Nanyun Peng. 2024. Model editing can hurt general abilities of large language models. *arXiv preprint arXiv:2401.04700* (2024).

[13] Akshat Gupta, Anurag Rao, and Gopala Anumanchipalli. 2024. Model Editing at Scale leads to Gradual and Catastrophic Forgetting. *arXiv preprint arXiv:2401.07453* (2024).

[14] Xiaoqi Han, Ru Li, Hongye Tan, Wang Yuanlong, Qinghua Chai, and Jeff Pan. 2023. Improving Sequential Model Editing with Fact Retrieval. In *Findings of the Association for Computational Linguistics: EMNLP 2023*. 11209–11224.

[15] Gautier Izacard, Mathilde Caron, Lucas Hosseini, Sebastian Riedel, Piotr Bojanowski, Armand Joulin, and Edouard Grave. 2021. Unsupervised dense information retrieval with contrastive learning. *arXiv preprint arXiv:2112.09118* (2021).

[16] Evgenii Kortukov, Alexander Rubinstein, Elisa Nguyen, and Seong Joon Oh. 2024. Studying Large Language Model Behaviors Under Realistic Knowledge Conflicts. *arXiv preprint arXiv:2404.16032* (2024).

[17] Junyi Li, Jie Chen, Ruiyang Ren, Xiaoxue Cheng, Wayne Xin Zhao, Jian-Yun Nie, and Ji-Rong Wen. 2024. The Dawn After the Dark: An Empirical Study on Factuality Hallucination in Large Language Models. *arXiv preprint arXiv:2401.03205* (2024).

[18] Junyi Li, Xiaoxue Cheng, Wayne Xin Zhao, Jian-Yun Nie, and Ji-Rong Wen. 2023. HELMA: A Large-Scale Hallucination Evaluation Benchmark for Large Language Models. *arXiv preprint arXiv:2305.11747* (2023).

[19] Kevin Meng, David Bau, Alex Andonian, and Yonatan Belinkov. 2022. Locating and editing factual associations in GPT. *NeurIPS* (2022).

[20] Kevin Meng, Arnab Sen Sharma, Alex Andonian, Yonatan Belinkov, and David Bau. 2022. Mass-editing memory in a transformer. *arXiv preprint arXiv:2210.07229* (2022).

[21] Eric Mitchell, Charles Lin, Antoine Bosselut, Chelsea Finn, and Christopher D Manning. 2021. Fast model editing at scale. *arXiv preprint arXiv:2110.11309* (2021).

[22] Eric Mitchell, Charles Lin, Antoine Bosselut, Christopher D Manning, and Chelsea Finn. 2022. Memory-based model editing at scale. In *ICML*.

[23] Ping Nie, Yuyu Zhang, Arun Ramamurthy, and Le Song. 2020. Answering Any-hop Open-domain Questions with Iterative Document Reranking. *arXiv preprint arXiv:2009.07465* (2020).

[24] OpenAI. 2023. GPT-3.5. https://openai.com/blog/gpt-3-5/. Accessed on [Date].

[25] OpenAI. 2023. Models Referred to as GPT-3.5. https://platform.openai.com/docs/models/gpt-3-5. Accessed on [Date].

[26] Long Ouyang, Jeffrey Wu, Xu Jiang, Diogo Almeida, Carroll Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, et al. 2022. Training language models to follow instructions with human feedback. *NeurIPS* (2022).

[27] Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, Ilya Sutskever, et al. 2019. Language models are unsupervised multitask learners. *OpenAI blog* 1, 8 (2019), 9.

[28] Hinrich Schütze, Christopher D Manning, and Prabhakar Raghavan. 2008. *Introduction to information retrieval*. Vol. 39. Cambridge University Press Cambridge.

[29] CARNEGIE-MELLON UNIV PITTSBURGH PA DEPT OF COMPUTER SCIENCE. 1977. *Speech Understanding Systems. Summary of Results of the Five-Year Research Effort at Carnegie-Mellon University*.

[30] Tony Sun, Andrew Gaut, Shirlyn Tang, Yuxin Huang, Mai ElSherief, Jieyu Zhao, Diba Mirza, Elizabeth Belding, Kai-Wei Chang, and William Yang Wang. 2019. Mitigating gender bias in natural language processing: Literature review. *arXiv preprint arXiv:1906.08976* (2019).

[31] Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, et al. 2023. Llama: Open and efficient foundation language models. *arXiv preprint arXiv:2302.13971* (2023).

[32] Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, et al. 2023. Llama 2: Open foundation and fine-tuned chat models. *arXiv preprint arXiv:2307.09288* (2023).

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

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

[35] Weixuan Wang, Barry Haddow, and Alexandra Birch. 2023. Retrieval-augmented Multilingual Knowledge Editing. *arXiv preprint arXiv:2312.13040* (2023).

[36] Yiwei Wang, Muhao Chen, Nanyun Peng, and Kai-Wei Chang. 2024. DeepEdit: Knowledge Editing as Decoding with Constraints. *arXiv preprint arXiv:2401.10471* (2024).

[37] Johannes Welbl, Amelia Glaese, Jonathan Uesato, Sumanth Dathathri, John Mellor, Lisa Anne Hendricks, Kirsty Anderson, Pushmeet Kohli, Ben Coppin, and Po-Sen Huang. 2021. Challenges in detoxifying language models. *arXiv preprint arXiv:2109.07445* (2021).

[38] Xuansheng Wu, Huachi Zhou, Yucheng Shi, Wenlin Yao, Xiao Huang, and Ninghao Liu. 2024. Could Small Language Models Serve as Recommenders? Towards Data-centric Cold-start Recommendation. In *Proceedings of the ACM on Web Conference 2024*.

[39] Jian Xie, Kai Zhang, Jiangjie Chen, Renze Lou, and Yu Su. 2023. Adaptive chameleon or stubborn sloth: Revealing the behavior of large language models in knowledge conflicts. In *ICLR*.

[40] Sang Michael Xie, Aditi Raghunathan, Percy Liang, and Tengyu Ma. 2021. An explanation of in-context learning as implicit bayesian inference. *arXiv preprint arXiv:2111.02080* (2021).

[41] Wenhan Xiong, Mo Yu, Xiaoxiao Guo, Hong Wang, Shiyu Chang, Murray Campbell, and William Yang Wang. 2019. Simple yet effective bridge reasoning for open-domain multi-hop question answering. *arXiv preprint arXiv:1909.07597* (2019).

[42] Vikas Yadav, Steven Bethard, and Mihai Surdeanu. 2020. Unsupervised alignment-based iterative evidence retrieval for multi-hop question answering. *arXiv preprint arXiv:2005.01218* (2020).

[43] Yunzhi Yao, Peng Wang, Bozhong Tian, Siyuan Cheng, Zhoubo Li, Shumin Deng, Huajun Chen, and Ningyu Zhang. 2023. Editing large language models: Problems, methods, and opportunities. *arXiv preprint arXiv:2305.13172* (2023).

[44] Xunjian Yin, Jin Jiang, Liming Yang, and Xiaojun Wan. 2024. History matters: Temporal knowledge editing in large language model. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 38. 19413–19421.

[45] Ori Yoran, Tomer Wolfson, Ori Ram, and Jonathan Berant. 2023. Making retrieval-augmented language models robust to irrelevant context. *arXiv preprint arXiv:2310.01558* (2023).

[46] Yuexiang Zhai, Shengbang Tong, Xiao Li, Mu Cai, Qing Qu, Yong Jae Lee, and Yi Ma. 2023. Investigating the Catastrophic Forgetting in Multimodal Large Language Models. *arXiv preprint arXiv:2309.10313* (2023).

[47] Jing Zhang, Xiaokang Zhang, Jifan Yu, Jian Tang, Jie Tang, Cuiping Li, and Hong Chen. 2022. Subgraph Retrieval Enhanced Model for Multi-hop Knowledge Base Question Answering. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. 5773–5784.

[48] Mengqi Zhang, Xiaotian Ye, Qiang Liu, Pengjie Ren, Shu Wu, and Zhumin Chen. 2024. Knowledge Graph Enhanced Large Language Model Editing. *arXiv preprint arXiv:2402.13593* (2024).

[49] Ningyu Zhang, Yunzhi Yao, Bozhong Tian, Peng Wang, Shumin Deng, Mengru Wang, Zekun Xi, Shengyu Mao, Jintian Zhang, Yuansheng Ni, et al. 2024. A comprehensive study of knowledge editing for large language models. *arXiv preprint arXiv:2401.01286* (2024).

[50] 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?

arXiv preprint arXiv:2305.12740 (2023).

[51] Shaochen Zhong, Duy Le, Zirui Liu, Zhimeng Jiang, Andrew Ye, Jiamu Zhang, Jiayi Yuan, Kaixiong Zhou, Zhaozhuo Xu, Jing Ma, Shuai Xu, Vipin Chaudhary, and Xia Hu. 2024. GNNs Also Deserve Editing, and They Need It More Than Once. In *Forty-first International Conference on Machine Learning*. https://openreview.net/forum?id=rIc9adYbH2

[52] 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 *EMNLP*.

[53] Chen Zhu, Ankit Singh Rawat, Manzil Zaheer, Srinadh Bhojanapalli, Daliang Li, Felix Yu, and Sanjiv Kumar. 2020. Modifying memories in transformer models. *arXiv preprint arXiv:2012.00363* (2020).