

# Decoupling Reasoning and Knowledge Injection for In-Context Knowledge Editing

Changyue Wang, Weihang Su, Qingyao Ai, Yujia Zhou, Yiqun Liu

Department of Computer Science and Technology, Tsinghua University

## Abstract

Knowledge editing enables efficient updates to Large Language Models (LLMs) by modifying specific knowledge without full-model retraining. Among knowledge editing approaches, in-context editing (ICE) stands out for its ability to inject knowledge without modifying the model’s parameters. However, existing ICE approaches directly edit model context without isolating target knowledge from the reasoning path of model inference, resulting in unreliable and low-quality outputs, particularly in multi-hop tasks. To investigate this issue, we analyze the interaction between reasoning path planning and knowledge injection, showing that the reasoning ability of a LLM is usually coupled with its original knowledge, and directly replacing old knowledge with new one could simultaneously hurt the LLM’s performance in task reasoning. Based on these findings, we propose DecKER, a novel ICE framework that separates model reasoning from knowledge editing. Extensive experiments show that DecKER significantly improves multi-hop reasoning performance by mitigating knowledge conflicts and preserving reasoning integrity.<sup>1</sup>

## 1 Introduction

Large language models (LLMs) have achieved remarkable performance in knowledge-intensive tasks. Yet, as real-world knowledge evolves rapidly, ensuring these models consistently maintain accurate and up-to-date knowledge remains a significant challenge. The most straightforward knowledge update approach is continued pre-training. However, the computational cost and memory requirements for retraining the entire model are prohibitively high. In response, knowledge editing has emerged as an effective alternative to full-model retraining. The goal of knowledge editing is to conduct rapid,

targeted modifications to specific embedded knowledge, enabling LLMs to stay updated without the need for retraining the entire model.

Existing knowledge editing approaches can be broadly categorized into two categories: parameterized methods that modify model weights (e.g., ROME (Meng et al., 2022)) and non-parameterized in-context editing (ICE) that injects knowledge through model context (Wang et al., 2024b). While parameterized editing directly modifies knowledge stored in LLMs, it often hurt the performance of LLMs on general tasks (Gu et al., 2024b; Li et al., 2024) and struggles with complex reasoning (Zhong et al., 2023). In contrast, ICE methods address the limitations of parameterized editing by preserving the original parameters of LLMs and directly edit the input context (e.g., prompts) to guide LLMs to behave based on the updated knowledge. They have received considerable attention due to their superior performance in managing knowledge ripple effects and multi-hop reasoning (Cohen et al., 2024; Zhong et al., 2023).

Nonetheless, despite of their relative advantages comparing to parameterized methods, we still observe a significant averaged performance drops on LLMs before and after applying existing methods. As discussed in section 3, we test several state-of-the-art ICE methods, and find that the performance of answering the same questions before and after using these methods drops more than 80% on multi-hop question answering tasks. Ideally, knowledge editing techniques should edit the knowledge of LLMs without hurting their ability to solve the task, and ICE should be able to achieve this as they haven’t modified the model parameters. Obviously, existing ICE methods fail to achieve this goal.

Our analytical experiments reveal that an important reason behind this phenomenon is that existing ICE methods often hurt the reasoning ability of LLMs. When answering questions with their internal knowledge, LLMs have exhibited superior

<sup>1</sup>Our code is available at: <https://github.com/bebr2/DecKER>

reasoning ability to solve multi-hop questions step by step, which is also referred to as the chain-of-thought (COT) (Wei et al., 2022) ability. However, after replacing its context with updated knowledge, LLMs often "forget" how to do reasoning in the same task due to the conflicts between its context and internal knowledge. In other words, the reasoning ability of LLMs is often coupled with its internal knowledge, and ICE without separating knowledge with reasoning would inevitably hurt the performance of LLMs. This phenomenon could limit the future applications of ICE methods, particularly when inference scaling, which heavily relies on the reasoning ability of LLMs, has recently become the dominant direction for super-intelligence development (DeepSeek-AI et al., 2025).

In light of these findings, we propose DecKER, a novel in-context editing method that Decouples Knowledge Editing and model Reasoning. Taking multi-hop QA as an example, our method first extract a masked reasoning path from LLMs, where entities subject to potential edits are replaced with placeholders along with corresponding type hints. Then, for each placeholder, we propose a hybrid mechanism combining retrieval-based conflict detection and model judgment to determine if the entity is related to edited knowledge and fill it accordingly. Furthermore, by sampling multiple reasoning paths and evaluating them based on a couple of intuitive criteria on reasoning and knowledge injection, our method selects the best candidate accordingly so that it can preserve the reasoning framework originally generated by LLMs while answering the question with updated knowledge.

In summary, this paper makes the following key contributions:

1. We explore the impact of the conflict between injected contextual knowledge and LLMs' parametric knowledge, revealing how entangled reasoning and editing processes lead to significant performance degradation in multihop tasks.
2. We introduce DecKER, a novel ICE framework that employs global planning to decouple reasoning from knowledge injection via masked reasoning path generation.
3. We conduct comprehensive experiments to demonstrate that decoupling reasoning from knowledge injection significantly enhances multi-hop reasoning performance, addressing the limitations of prior ICE approaches.

## 2 Related Work

Knowledge Editing (KE) aims to efficiently update knowledge in LLMs, divided into parameterized and non-parameterized methods (Wang et al., 2024b). ROME (Meng et al., 2022), a typical parameterized method, uses causal intervention to locate and edit related neurons, while MEMIT (Meng et al., 2023) extends this capability to handle larger edit batch. Non-parameterized methods leverage the LLM's in-context learning (Brown et al., 2020) abilities. Mello (Zhong et al., 2023) employs in-context editing (ICE) to tackle complex problems by breaking them into subtasks and performing fine-grained edits. Built on this, PokeMQA (Gu et al., 2024a) enhances robustness and DeepEdit (Wang et al., 2024c) focuses on the reasoning process. Additionally, Shi et al. (2024) address multi-hop tasks in knowledge editing by emphasizing the retrieval process with their RAE method, which employs knowledge graph editing and retrieval to boost multi-hop reasoning performance.

To evaluate LLMs' reasoning abilities post-editing, Cohen et al. (2024) introduce the concept of Ripple Effects, where altering one piece of knowledge can impact related facts. Zhong et al. (2023) develop a multi-hop QA dataset to assess if edited models can utilize new knowledge for complex reasoning. Their findings indicate that ICE methods outperform parameterized methods in managing Ripple Effects and complex reasoning.

## 3 Preliminary Study

This section begins by introducing the multi-hop question-answering (MQA) task under knowledge editing. Then, we discuss the phenomenon of reasoning degradation in existing ICE methods and design an analytical experiment to explain it.

### 3.1 MQA under Knowledge Editing

Knowledge editing in the MQA task involves modifying the object component of a knowledge triple. Given an initial fact  $e = (s, r, o)$ , comprising a subject ( $s$ ), a relation ( $r$ ), and an object ( $o$ ), this fact is edited to  $e^* = (s, r, o^*)$ . For a multi-hop question  $Q$  and a set of edited facts  $\mathcal{E}$  associated with  $Q$ , the reasoning path  $P_{Q,\mathcal{E}}$  is represented as  $\langle e_1, \dots, e_n \rangle = \langle (s_1, r_1, o_1), \dots, (s_n, r_n, o_n) \rangle$ , where  $n$  is the number of reasoning hops,  $s_{i+1} = o_i$ , and  $o_n$  is the final answer. The reasoning path before editing is denoted as  $P_{Q,\emptyset}$ . If  $\mathcal{E}$  is non-empty, indicating modifications to one or more knowledge

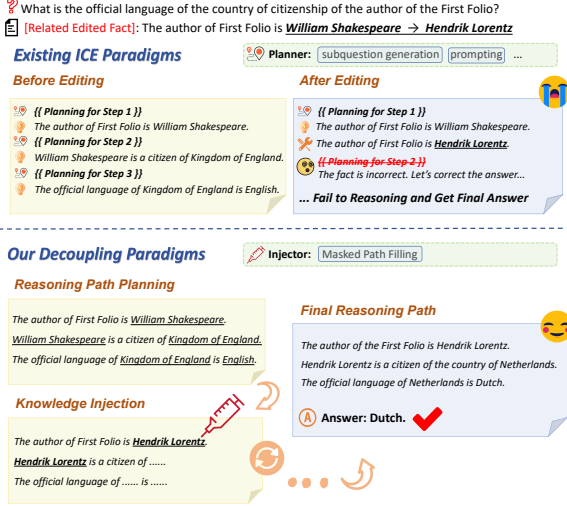


Figure 1: Comparisons of two types of ICE paradigms.

triples in  $Q$  (e.g.,  $e_k = (s_k, r_k, o_k)$  is edited into  $(s_k, r_k, o_k^*)$ ), the reasoning path  $P_{Q,\mathcal{E}}$  will reflect changes to  $e_k$  and all subsequent triples, thus altering the final answer. Notably, successfully addressing MQA depends on two key components: accurate reasoning path planning and precise knowledge injection. Reasoning path planning involves determining the reasoning framework, i.e., the list of relations  $R_Q = [r_1, \dots, r_n]$  in the reasoning path  $P_{Q,\mathcal{E}}$ , while knowledge injection ensures the model accurately provides  $o_k$  given  $(s_k, r_k)$ . In real tasks, the edit batch size (the size of the union set of all  $\mathcal{E}$ ) is often greater than one, and one reasoning path  $P_{Q,\mathcal{E}}$  may involve multiple edits, adding significant complexity to knowledge editing methods.

### 3.2 Reasoning Degradation in ICE

The upper part of Figure 1 illustrates a typical example of Reasoning Degradation in current ICE methods. These methods can be abstracted as interactions between a *Planner*, which determines the next reasoning step through sub-question generation (e.g., Mello and PokeMQA) or prompting (e.g., DeepEdit and EditCoT), and an *Editor*, which incorporates edited knowledge by replacing intermediate answers or prompting the model to revise them. Before the relevant knowledge is edited, the LLM uses its internal knowledge for reasoning, allowing the interaction between the Planner and Editor to proceed normally and produce the correct reasoning path. However, after the knowledge is edited and the Editor replaces the contextual knowledge with new one, the Planner does not continue reasoning as expected. Instead, it shifts direction,

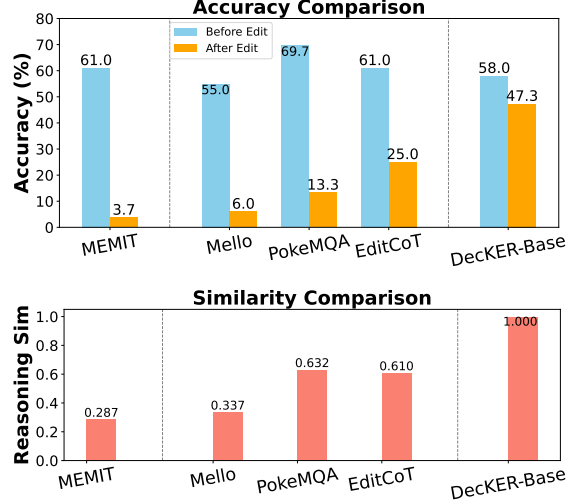


Figure 2: Performance and reasoning similarity of different KE Methods before and after editing. The second image indicates the similarity between the post-editing and pre-editing reasoning framework.

such as attempting to resolve conflicting knowledge. Moreover, as demonstrated in Appendix E, we present additional examples where the Planner influenced by edited knowledge exhibits behaviors like causing reasoning interruptions or redirecting to different reasoning paths. Clearly, current ICE methods fail to separate knowledge from reasoning, thereby impairing the reasoning abilities of LLMs. We design an experiment to quantitatively assess the Reasoning Degradation phenomenon.

**Experimental Setup** We randomly select 300 questions from MQuAKE-CF-3k-v2 (Zhong et al., 2023), along with the associated edited facts, for our experiment. We apply a knowledge editing method to Llama-3.1-8B-Instruct (Dubey et al., 2024) and compare its performance before and after editing. We evaluate several methods, including MEMIT (a parameterized method), Mello, PokeMQA, and EditCoT. The first two are classic methods of parameterized knowledge editing and in-context editing, while the latter two represent the state-of-the-art ICE methods. The experiment consists of two main parts:

**Performance Comparison:** For multi-hop questions involving edited knowledge, the answers change before and after editing. We use the ground truths before and after editing as references, with accuracy as the metric, to evaluate performance before and after editing, respectively. For MEMIT, whether before or after editing, we prompt the LLM to generate a CoT during inference. For EditCoT,

we also get pre-editing answers through CoT reasoning. For Mello and PokeMQA, pre-editing performance is obtained by replacing the edited fact in the retrieval corpus with the original one.

**Reasoning Framework Comparison:** The reasoning framework  $R_Q$  for a question is independent of the edit set  $\mathcal{E}$ , meaning the relation lists in the paths should remain unchanged after editing. A good ICE method should maintain this similarity to preserve the model’s reasoning ability. We use regular expressions to identify the reasoning paths generated by each knowledge editing method. For CoT-based methods, the CoT itself serves as the reasoning path, while for sub-question decomposition methods, we combine the answers to all sub-questions into the reasoning path. We then use GPT-4o-mini-0718 (OpenAI et al., 2023) to identify the relations in the knowledge triples involved at each reasoning step, forming the reasoning framework  $R$ . Prompts are detailed in Appendix G.

Then, we compare the similarity of reasoning framework before and after editing for each method. Given two reasoning frameworks  $R_1 = [r_{11}, \dots, r_{1n}]$  and  $R_2 = [r_{21}, \dots, r_{2m}]$ , we compute the similarity using the following formula:

$$\text{Similarity}(R_1, R_2) = \frac{\sum_{i=1}^{\min(n,m)} \text{Sim}(r_{1i}, r_{2i})}{\max(n, m)}. \quad (1)$$

It calculates the similarity between corresponding elements of two ordered lists, padding the shorter list with zeros if they differ in length. In Equation 1, the "Sim" function employs jina-embeddings-v3 (Sturua et al., 2024) to compute the embeddings of the two elements individually and calculates their cosine similarity.

**Results and Analysis** Figure 2 presents the results. Both MEMIT and current ICE methods show significant performance degradation. Ideally, a knowledge editing method should maintain a reasoning framework similarity close to 1.00, but the tested methods do not exceed 0.65, demonstrating substantial reasoning degradation. This phenomenon is most severe for the parameterized method MEMIT, as noted in previous research (Li et al., 2024). ICE, which does not modify LLM parameters, should theoretically maintain the model’s reasoning capability, but the low reasoning framework similarity for the three ICE methods suggests that they fail to achieve this goal. Additionally, we observe a positive correlation between editing performance and reasoning framework similarity,

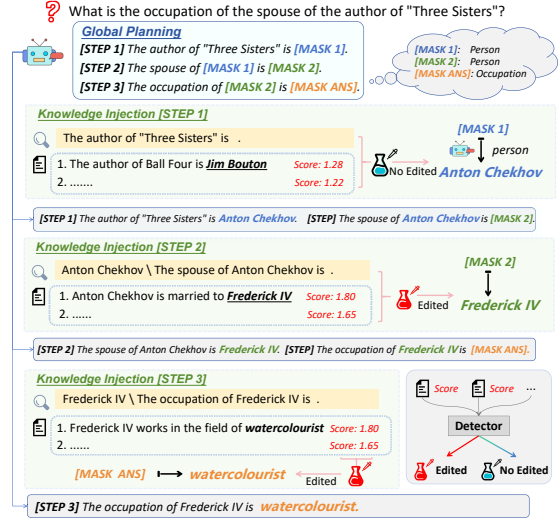


Figure 3: The workflow of DecKER, only showing the processing of a single path, omitting the final answer selection stage for multiple sampled reasoning paths.

further emphasizing the importance of maintaining the reasoning framework. In reasoning tasks, the reasoning planning and knowledge injection of current ICE methods are coupled, and the injection of conflicting knowledge affects reasoning planning. Therefore, this highlights the importance of decoupling reasoning from editing.

## 4 Methodology

We propose DecKER, a method decoupling reasoning and knowledge injection to maintain reasoning framework while editing. The workflow is shown in Figure 3, with its pseudocode in Appendix A.

### 4.1 Masked Reasoning Path Generation

We separate reasoning from editing by initially having the LLM conduct global planning. Specifically, we first prompt the LLM to generate a masked reasoning path for the provided multi-hop question  $Q$ . "Masked" means that during generation, the LLM replaces all positions that require entity generation (except those already present in the question) with [MASK \*] symbols, where \* can be an integer or string. We provide the LLM with a 5-shot prompt, leveraging the instruction-following and in-context learning capabilities of the LLM to accomplish this. This approach aims to preserve high-quality reasoning paths from the original LLM before editing. In practice, we instruct the LLM to start each reasoning step with a [STEP] symbol. Entities are numbered sequentially by the LLM to maintain consistency, as illustrated in Figure 3, where the



author of "Three Sisters" is represented as [MASK 1] in the first two steps. The final answer is represented as [MASK ANS].

We also have the LLM provide entity type hints, such as "country", "company", etc. This is crucial because each filling step is completed independently. Without type hints, the model might generate completions that fit the instructions but do not align with the reasoning framework. All of the prompt templates are detailed in [Appendix G](#).

## 4.2 Stepwise Knowledge Injection

Then we fill in each masked entity step by step.

**1. Retrieval:** For a masked step currently being filled, we remove the [MASK \*] tag in it and use it as a query to search within the edited memory (i.e., the union set of all  $\mathcal{E}$  in the dataset). If previous entities have been filled, we include the most recent entity in the query.

**2. Conflict Detection:** Conflict detection is performed by comparing the retrieval results with the query, using a hybrid method that combines retrieval scores and LLM judgments. When a query involves edits, the retrieval scores exhibit two key features: a notably high score for the top result and a substantial gap between the highest and second-highest scores, mathematically represented as:

$$S(d_1) > \alpha, \quad (2a)$$

$$S(d_1) - S(d_2) > \beta, \quad (2b)$$

where  $S$  is the retrieval score. In our experiments, it denotes the dot-product similarity between the embeddings of documents and queries, with  $d_i$  representing the result ranked  $i$ .  $\alpha$  and  $\beta$  are hyperparameters. If both criteria yield consistent results, we accept them without further LLM evaluation. Otherwise, LLM analysis is conducted. Each edited fact maps to an original one, allowing the LLM to determine if the original supports the masked step's filling. Support indicates a conflict between the edited fact and the masked step.

**3. Filling:** When a retrieval result conflicts with the masked step, we directly replace the masked part with the object from the knowledge triple corresponding to the retrieval result, and also update subsequent identical masked parts. For example, in [Figure 3](#), all instances of [MASK 2] are replaced with "Frederick IV". If no conflict is detected, the model treats the masked part as a fill-in-the-blank task, guided by the previously assigned entity type. This process continues until [MASK ANS] is filled, providing the final answer to the question.

## 4.3 Final Answer Selection

In the workflow, we assume that each masked step involves only one entity to be filled, which is reasonable for the MQA task. However, due to insufficient few-shot learning ability, the model might generate incorrect or incomplete masked reasoning paths, and the filled entities might not align with the intended types. To ensure the quality of the final answers, we filter multiple candidate answers generated through the above process with two evaluation methods — reasoning path planning and knowledge injection — to select the final answer.

**1. Reasoning Path Planning Evaluation (RPP Eval):** Inspired by the work on hallucinations ([Sut et al., 2024](#)) in LLMs, we can evaluate the reasoning path by calculating the model's uncertainty, as LLMs are prone to hallucinations or incorrect outputs when uncertain. For each generated token, Predictive Entropy (PE) is defined as follows:

$$PE = - \sum_{\tilde{w} \in W} p_i(\tilde{w}) \log p_i(\tilde{w}), \quad (3)$$

where  $W$  is the vocabulary of the LLM, and  $p_i(\tilde{w})$  is the generation probability of token  $\tilde{w}$  at this step. For each masked path, we use the negative average PE over all generated tokens as the evaluation score, with higher scores indicating greater certainty.

**2. Knowledge Injection Evaluation (KI Eval):** After filling the reasoning path, we obtain a list of filled entities. For each filled entity  $e_i$ , we instruct the model to determine whether it matches the pre-assigned type  $t_i$ . The score is calculated as follows:

$$KI(e_i, t_i) = \mathbb{1}[P_M(\text{"yes"} | e_i, t_i) > P_M(\text{"no"} | e_i, t_i)], \quad (4)$$

where  $M$  is the LLM and  $\mathbb{1}$  is an indicator function. The prompt template is detailed in [Appendix G](#). We compute the KI Eval score by averaging the scores of all filled entities. In practice, RPP Eval is used to retain the top 50% of sampled masked reasoning paths for filling. The final answer is selected from the filled path with the highest KI Eval score.

## 4.4 Discussion

We conduct the experiments in [subsection 3.2](#) on DecKER. For the pre-editing settings, we replace the edited memory with the corresponding original facts. The results indicate DecKER exhibits only a slight accuracy drop post-editing and significantly outperforms other baselines. Additionally, DecKER maintains complete consistency in reasoning frameworks between pre- and post-editing, as they share the same masked reasoning path, highlighting the reason behind its strong performance.

Table 1: The overall results. Ripple-Pop and Ripple-Rand denote the popular and random subsets of RippleEdits. The metric is Multi-Hop Accuracy (%). We bold the top performing methods and underline the second-best ones.

Models	Methods	MQuAKE-CF-3k-v2	MQuAKE-T	Ripple-Pop	Ripple-Rand
Llama-3.1-8B-Instruct	MEMIT	6.8	44.1	8.4	9.1
	AlphaEdit	7.0	42.0	8.6	9.6
	Mello	11.0	57.7	25.8	32.7
	PokeMQA	16.8	77.8	30.4	33.1
	DeepEdit	8.9	57.5	3.4	5.0
	RAE	54.3	50.9	20.6	17.0
	EditCoT	36.0	74.6	31.6	22.3
	DeckKER-Base	<u>57.8</u>	<u>80.9</u>	<u>46.6</u>	<b>46.1</b>
	DeckKER-BoN ( $N = 6$ )	<b>59.0</b>	<b>81.3</b>	<b>47.4</b>	<u>44.9</u>
Qwen2.5-7B-Instruct	MEMIT	6.6	21.5	9.8	17.2
	AlphaEdit	7.1	21.1	11.0	17.0
	Mello	1.7	50.3	3.6	1.4
	PokeMQA	7.1	45.3	7.0	7.6
	DeepEdit	12.3	25.3	6.0	12.0
	RAE	38.3	28.7	24.0	18.3
	EditCoT	26.2	<b>74.5</b>	29.2	30.7
	DeckKER-Base	<u>42.1</u>	<u>67.7</u>	<u>35.8</u>	<u>32.1</u>
	DeckKER-BoN ( $N = 6$ )	<b>49.8</b>	<u>70.8</u>	<b>39.8</b>	<b>38.3</b>
Qwen2.5-14B-Instruct	Mello	1.2	26.4	4.6	2.3
	PokeMQA	2.3	32.4	5.9	3.9
	DeepEdit	7.3	51.7	9.4	14.4
	EditCoT	37.0	<u>82.7</u>	35.0	33.5
	DeckKER-Base	<u>50.8</u>	81.4	<b>45.0</b>	<u>45.6</u>
	DeckKER-BoN ( $N = 6$ )	<b>56.6</b>	<b>83.6</b>	<u>44.0</u>	<b>46.1</b>

## 5 Experimental Settings

### 5.1 Evaluation Details

We evaluate our methods using four datasets: MQuAKE-CF-3k-v2, MQuAKE-T (Zhong et al., 2023), and the Popular and Random subsets of RippleEdits (Cohen et al., 2024). MQuAKE-CF-3k-v2 comprises 3,000 questions with 2,764 edits, with 2-, 3-, and 4-hop questions. MQuAKE-T contains 1,868 questions and 96 edits, with 2- and 3-hop questions. RippleEdits-Popular has 500 2-hop questions and 266 edits, while RippleEdits-Random has 1,137 2-hop questions and 626 edits.

As in the original paper of Mello, we employ the Multi-Hop Accuracy metric, where a question is correct if at least one rewritten version is answered correctly. For MQuAKE datasets, each test includes 3 rewritten questions, while RippleEdits datasets contain only 1 question per test.

Three LLMs are evaluated: Meta-Llama-3.1-8B-Instruct (Dubey et al., 2024), Qwen2.5-7B-Instruct, and Qwen2.5-14B-Instruct (Team, 2024), using their official chat templates across all baselines.

### 5.2 Baselines

We compare DeckKER with other ICE methods and two parameterized approaches: MEMIT (Meng et al., 2023) and AlphaEdit (Fang et al., 2024). These two approaches focus on editing neurons

related to the targeted knowledge, with AlphaEdit mitigating unintended disruptions through matrix projection. ICE methods include: (1) Mello (Zhong et al., 2023) and PokeMQA (Gu et al., 2024a), subproblem-based methods; (2) DeepEdit (Wang et al., 2024c), depth-first search-based methods; and (3) EditCoT (Wang et al., 2024a), CoT editing methods. We also evaluate RAE (Shi et al., 2024), a method that achieves knowledge editing by modifying knowledge graphs. Due to the resource-intensive nature of the parameterized methods and RAE, we test them only on the two smaller LLMs. Detailed implementation is in Appendix B.

### 5.3 Implementation Details of DeckKER

We evaluate two settings:

**DeckKER-BoN:** Sample  $N$  masked reasoning paths, use RPP Eval (Equation 3) to select the best half for filling, and then employ KI Eval (Equation 4) to choose the best result. One greedy decoding is followed by  $N - 1$  top-p sampling (probability threshold 0.95, temperature 1.2). Other generation processes use greedy decoding. In our experiments,  $N$  is set to 6.

**DeckKER-Base:** This method does not utilize two round evaluation. Instead, it generates one masked reasoning path using greedy decoding. If the format is incorrect, e.g. no masked parts, a new

Table 2: Results on GPT-4o-mini. The dataset is MQuAKE-300. The best is in bold.

Mello	PokeMQA	EditCoT	RAE	DecKER-Base
11.0	51.3	46.7	67.0	<b>69.7</b>

path is sampled by nucleus sampling. Failure in both attempts is viewed as an error result.

In conflict detection,  $\alpha$  is set to 1.5 and  $\beta$  is 0.1.

## 5.4 Retrieval Settings

Except for RAE, which uses knowledge graph retrieval, all ICE methods pertaining to retrieval, following Mello with contriever-msmarco(Izacard et al., 2022) as the retriever, accessing a knowledge base of edited knowledge. Notably, RAE accesses Wikidata(Wang et al., 2021) during inference steps even involving unedited knowledge, creating an unfair comparison. To ensure consistency, we modify its graph retrieval process: if the subject and the retrieved relation involve unedited knowledge triples, the LLM generates the object of the triple instead of getting it from Wikidata. This adjustment prevents data leakage, ensuring a uniform retrieval scope across all methods.

## 6 Experimental Results

### 6.1 Main Results

**DecKER outperforms all baselines.** Table 1 presents the main results, showing that DecKER-Base and DecKER-BoN outperform previous methods across most models and datasets, particularly excelling in the Popular subset of RippleEdits. This highlights DecKER’s effectiveness in handling multi-hop reasoning and the Ripple Effect after editing. Moreover, DecKER demonstrates consistent performance, unlike other baselines, such as PokeMQA, which may excel in some datasets and models but perform poorly in others. This suggests that DecKER achieves a stable decoupling of reasoning and editing, maintaining the model’s original reasoning capabilities.

**Sampling multiple paths enhances DecKER’s performance.** DecKER-BoN outperforms the Base version by selecting reasoning paths through two evaluation rounds from options, prioritizing paths with higher LLM confidence and better entity type matching. However, the BoN version occasionally underperforms the Base version, suggesting the potential of improving evaluation design.

**Preserving the reasoning framework is crucial.** EditCoT and PokeMQA exhibit more com-

petitive performance than Mello and MEMIT. As analyzed in subsection 3.2, despite these two methods not being able to completely retain the pre-edit reasoning framework, they still surpass Mello and MEMIT in the similarity dimension, leading to higher answer accuracy. RAE, a graph retrieval method, constrains the search space for the construction of reasoning frameworks by LLMs, as the edges in the graph are fixed, resulting in commendable performance. These results confirm that maintaining the reasoning framework is essential.

### 6.2 Comparative Analysis

**Performance on Proprietary LLMs** One advantage of in-context editing is its applicability to proprietary models. We test GPT-4o-mini-0718. Following Shi et al. (2024), we randomly select 300 questions from MQuAKE-CF-3k-v2 along with the corresponding edits, named MQuAKE-300. Since some processes in EditCoT and RAE, such as the editor or the complete output probabilities, cannot be executed on proprietary LLMs, we use Llama-3.1-8B-Instruct as a proxy model, following Wang et al. (2024a). The results are presented in Table 2. DecKER-Base outperforms all baselines, indicating its potential for editing proprietary LLMs.

**Efficiency** Figure 4 presents the average inference time per problem and performance on MQuAKE-3k-CF-v2 for various ICE methods. The LLM is Llama-3.1-8B-Instruct. DecKER-Base is positioned in the upper left corner, achieving superior editing results over other baselines in the shortest time. DecKER-BoN, due to the additional evaluation and more sampled sentences, takes slightly longer but achieves the best. Among all the baselines, the top-performing RAE requires over 24 seconds per problem, while the fastest EditCoT exhibits weaker performance compared to ours, highlighting the efficiency advantage of our method. The details are in Appendix C.

**Performance on Problems with Different Hop Numbers** We analyze the performance of several competitive methods on two MQuAKE datasets based on the number of hops, as shown in Figure 5. DecKER-BoN consistently ranks first or second in 2-4 hop problems, with particularly strong performance on 2-hop problems, reflecting the inherent reasoning abilities of LLMs. RAE leverages a knowledge graph, maintaining stable performance across 2-4 hops by constraining the exploration space during inference with fixed graph edges. Despite this, DecKER matches or exceeds RAE’s per-

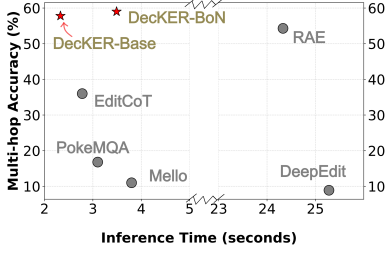


Figure 4: Avg. time vs. Performance. We truncate the x-axis due to its length.

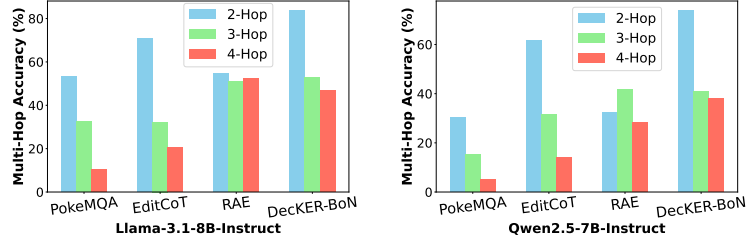


Figure 5: #Hop vs. Performance: performance on the MQuAKE-CF-3k-v2 and -T datasets based on the number of hops of each problem.

Table 3: An ablation study on the selection process on MQuAKE-CF-3k-v2. Using "L" for Llama-3.1-Instruct and "Q" for Qwen2.5-Instruct. "Random Choice" refers to randomly selecting one answer from 6 sampled paths.

	L-8B	Q-7B	Q-14B
<b>DeckKER-BoN</b>	59.9	<b>49.8</b>	<b>56.6</b>
<i>w/o RPP Eval</i>	<b>60.2</b>	49.4	56.2
<i>w/o KI Eval</i>	58.1	49.5	55
<b>DeckKER-Base</b>	57.8	42.1	50.8
<b>Random Choice</b>	55.2	47.1	51.6

formance on 3 and 4 hop problems.

### 6.3 Ablation Studies

In this section, We perform ablation studies on Llama-3.1-8B-Instruct.

**Components of Conflict Detection** In Figure 6, we compare the performance of DeckKER-Base with three experimental settings on MQuAKE-300: conflict detection 1) using only LLM; 2) using only the highest retrieval score as shown in Equation 2a (*Only detected by  $\alpha$* ); 3) using only the difference between retrieval scores of the top two documents as shown in Equation 2b (*Only detected by  $\beta$* ). The hyperparameter settings are consistent with the main experiments. Results indicate that single-method detection degrades performance, especially with LLM-only detection. The combination of all three methods achieves the best performance. Additionally, we analyze hyperparameter impact by fixing either  $\alpha$  or  $\beta$  while varying the other, as shown in Figure 6. The stability of performance within a certain range of these parameters demonstrates the robustness of conflict detection.

**Two-round Evaluation** In Table 3, we present the impact of RPP eval and KI eval in DeckKER-BoN on MQuAKE-CF-3k-v2. The Reasoning Path Planning Evaluation alone yields the poorest performance, while the Knowledge Injection Evaluation alone performs comparably to using both evalua-

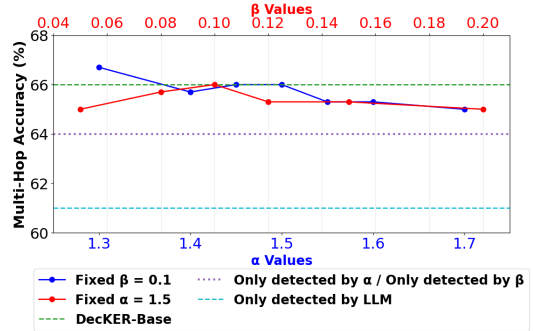


Figure 6: Ablation studies for conflict detection on DeckKER-Base. The dashed lines compare single versus composite detection, with identical hyperparameters to main experiments. The solid line shows the effect of varying one hyperparameter while fixing the other, with the color of x-axis labels matching the solid line.

tions. Notably, RPP Eval involves filtering multiple masked reasoning paths, thus the computational load during filling is less than that of performing only KI Eval, demonstrating its effective of reducing computational costs. Additionally, selecting one answer randomly from 6 sampled paths significantly underperforms compared to DeckKER-BoN. Additionally, we explore the impact of edit batch size and the number of sampled masked paths on our method. The results are shown in Appendix D.

## 7 Conclusion

In this paper, we explore Multi-hop QA to highlight the importance of decoupling reasoning and knowledge injection when applying in-context editing. Consequently, we propose a novel in-context editing method, DeckKER, which achieves this decoupling by first planning the reasoning path and then filling knowledge entities. Our method demonstrates strong performance across 3 LLMs and 4 datasets, offering a new perspective for optimizing in-context editing methods.



## 8 Limitations

In this work, we conduct evaluations using three widely-used open-source LLMs with parameter sizes ranging from 7B to 14B, as well as a proprietary LLM. Due to resource constraints, we do not perform experiments on larger parameter-scale open-source models. Additionally, experiments with the three resource-intensive baselines are conducted only on the 7B and 8B parameter LLMs.

Besides, we focus on the enhancement of the knowledge editing process, with DecKER’s performance relying solely on the LLM’s own reasoning capabilities. The integration of knowledge augmentation techniques, such as knowledge graphs, to further improve DecKER’s performance remains an open area of research.

## References

- Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel M. Ziegler, Jeffrey Wu, Clemens Winter, Christopher Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. 2020. [Language models are few-shot learners](#). In *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual*.
- Roi Cohen, Eden Biran, Ori Yoran, Amir Globerson, and Mor Geva. 2024. [Evaluating the ripple effects of knowledge editing in language models](#). *Transactions of the Association for Computational Linguistics*, 12:283–298.
- DeepSeek-AI, Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Ruoyu Zhang, Runxin Xu, Qihao Zhu, Shirong Ma, Peiyi Wang, Xiao Bi, Xiaokang Zhang, Xingkai Yu, Yu Wu, Z. F. Wu, Zhibin Gou, Zhihong Shao, Zhuoshu Li, Ziyi Gao, Aixin Liu, Bing Xue, Bingxuan Wang, Bochao Wu, Bei Feng, Chengda Lu, Chenggang Zhao, Chengqi Deng, Chenyu Zhang, Chong Ruan, Damai Dai, Deli Chen, Dongjie Ji, Erhang Li, Fangyun Lin, Fucong Dai, Fuli Luo, Guangbo Hao, Guanting Chen, Guowei Li, H. Zhang, Han Bao, Hanwei Xu, Haocheng Wang, Honghui Ding, Huajian Xin, Huazuo Gao, Hui Qu, Hui Li, Jianzhong Guo, Jiashi Li, Jiawei Wang, Jingchang Chen, Jingyang Yuan, Junjie Qiu, Junlong Li, J. L. Cai, Jiaqi Ni, Jian Liang, Jin Chen, Kai Dong, Kai Hu, Kaige Gao, Kang Guan, Kexin Huang, Kuai Yu, Lean Wang, Lecong Zhang, Liang Zhao, Litong Wang, Liyue Zhang, Lei Xu, Leyi Xia, Mingchuan Zhang, Minghua Zhang, Minghui Tang, Meng Li, Miaoqun Wang, Mingming Li, Ning Tian, Panpan Huang, Peng Zhang, Qiancheng Wang, Qinyu Chen, Qiusi Du, Ruiqi Ge, Ruisong Zhang, Ruizhe Pan, Runji Wang, R. J. Chen, R. L. Jin, Ruyi Chen, Shanghao Lu, Shangyan Zhou, Shanhuang Chen, Shengfeng Ye, Shiyu Wang, Shuiping Yu, Shunfeng Zhou, Shuting Pan, S. S. Li, Shuang Zhou, Shaoqing Wu, Shengfeng Ye, Tao Yun, Tian Pei, Tianyu Sun, T. Wang, Wangding Zeng, Wanbiao Zhao, Wen Liu, Wenfeng Liang, Wenjun Gao, Wenqin Yu, Wentao Zhang, W. L. Xiao, Wei An, Xiaodong Liu, Xiaohan Wang, Xiaokang Chen, Xiaotao Nie, Xin Cheng, Xin Liu, Xin Xie, Xingchao Liu, Xinyu Yang, Xinyuan Li, Xuecheng Su, Xuheng Lin, X. Q. Li, Xiangyue Jin, Xiaojin Shen, Xiaosha Chen, Xiaowen Sun, Xiaoxiang Wang, Xinnan Song, Xinyi Zhou, Xianzu Wang, Xinxia Shan, Y. K. Li, Y. Q. Wang, Y. X. Wei, Yang Zhang, Yanhong Xu, Yao Li, Yao Zhao, Yaofeng Sun, Yaohui Wang, Yi Yu, Yichao Zhang, Yifan Shi, Yiliang Xiong, Ying He, Yishi Piao, Yisong Wang, Yixuan Tan, Yiyang Ma, Yiyuan Liu, Yongqiang Guo, Yuan Ou, Yuduan Wang, Yue Gong, Yuheng Zou, Yujia He, Yunfan Xiong, Yuxiang Luo, Yuxiang You, Yuxuan Liu, Yuyang Zhou, Y. X. Zhu, Yanhong Xu, Yanping Huang, Yaohui Li, Yi Zheng, Yuchen Zhu, Yunxian Ma, Ying Tang, Yukun Zha, Yuting Yan, Z. Z. Ren, Zehui Ren, Zhangli Sha, Zhe Fu, Zhean Xu, Zhenda Xie, Zhengyan Zhang, Zhewen Hao, Zhicheng Ma, Zhigang Yan, Zhiyu Wu, Zihui Gu, Zijia Zhu, Zijun Liu, Zilin Li, Ziwei Xie, Ziyang Song, Zizheng Pan, Zhen Huang, Zhipeng Xu, Zhongyu Zhang, and Zhen Zhang. 2025. [Deepseek-r1: Incantivizing reasoning capability in llms via reinforcement learning](#). *Preprint*, arXiv:2501.12948.
- Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Amy Yang, Angela Fan, Anirudh Goyal, Anthony Hartshorn, Aobo Yang, Archi Mitra, Archie Sravankumar, Artem Korenev, Arthur Hinsvark, Arun Rao, Aston Zhang, Aurelien Rodriguez, Austen Gregerson, Ava Spataru, Baptiste Roziere, Bethany Biron, Binh Tang, Bobbie Chern, Charlotte Caucheteux, Chaya Nayak, Chloe Bi, Chris Marra, Chris McConnell, Christian Keller, Christophe Touret, Chunyang Wu, Corinne Wong, Cristian Canton Ferrer, Cyrus Nikolaidis, Damien Al-lonsius, Daniel Song, Danielle Pintz, Danny Livshits, David Esiobu, Dhruv Choudhary, Dhruv Mahajan, Diego Garcia-Olano, Diego Perino, Dieuwke Hupkes, Egor Lakomkin, Ehab AlBadawy, Elina Lobanova, Emily Dinan, Eric Michael Smith, Filip Radenovic, Frank Zhang, Gabriel Synnaeve, Gabrielle Lee, Georgia Lewis Anderson, Graeme Nail, Gregoire Mialon, Guan Pang, Guillem Cucurell, Hailey Nguyen, Hannah Korevaar, Hu Xu, Hugo Touvron, Iliyan Zarov, Imanol Arrieta Ibarra, Isabel Kloumann, Ishan Misra, Ivan Evtimov, Jade Copet, Jaewon Lee, Jan Geffert, Jana Vranes, Jason Park, Jay Mahadeokar, Jeet Shah, Jelmer van der Linde, Jennifer Billock, Jenny Hong, Jenya Lee, Jeremy Fu, Jianfeng Chi, Jianyu Huang, Jiawen Liu, Jie Wang, Jiecao Yu, Joanna Bitton, Joe Spisak, Jongsoo Park, Joseph

Rocca, Joshua Johnstun, Joshua Saxe, Junteng Jia, Kalyan Vasuden Alwala, Kartikeya Upasani, Kate Plawiak, Ke Li, Kenneth Heafield, Kevin Stone, Khalid El-Arini, Krithika Iyer, Kshitiz Malik, Kuenley Chiu, Kunal Bhalla, Lauren Rantala-Yearly, Laurens van der Maaten, Lawrence Chen, Liang Tan, Liz Jenkins, Louis Martin, Lovish Madaan, Lubo Malo, Lukas Blecher, Lukas Landzaat, Luke de Oliveira, Madeline Muzzi, Mahesh Pasupuleti, Mannat Singh, Manohar Paluri, Marcin Kardas, Mathew Oldham, Mathieu Rita, Maya Pavlova, Melanie Kambadur, Mike Lewis, Min Si, Mitesh Kumar Singh, Mona Hassan, Naman Goyal, Narjes Torabi, Nikolay Bashlykov, Nikolay Bogoychev, Niladri Chatterji, Olivier Duchenne, Onur Çelebi, Patrick Alrassy, Pengchuan Zhang, Pengwei Li, Petar Vasic, Peter Weng, Prajjwal Bhargava, Pratik Dubal, Praveen Krishnan, Punit Singh Koura, Puxin Xu, Qing He, Qingxiao Dong, Ragavan Srinivasan, Raj Ganapathy, Ramon Calderer, Ricardo Silveira Cabral, Robert Stojnic, Roberta Raileanu, Rohit Girdhar, Rohit Patel, Romain Sauvestre, Ronnie Polidoro, Roshan Sumbaly, Ross Taylor, Ruan Silva, Rui Hou, Rui Wang, Saghar Hosseini, Sahana Chennabasappa, Sanjay Singh, Sean Bell, Seohyun Sonia Kim, Sergey Edunov, Shaoqiang Nie, Sharan Narang, Sharath Raparthy, Sheng Shen, Shengye Wan, Shruti Bhosale, Shun Zhang, Simon Vandenhende, Soumya Batra, Spencer Whitman, Sten Sootla, Stephane Collot, Suchin Gururangan, Sydney Borodinsky, Tamar Herman, Tara Fowler, Tarek Sheasha, Thomas Georgiou, Thomas Scialom, Tobias Speckbacher, Todor Mihaylov, Tong Xiao, Ujjwal Karn, Vedanuj Goswami, Vibhor Gupta, Vignesh Ramanathan, Viktor Kerkez, Vincent Gonguet, Virginie Do, Vish Vogeti, Vladan Petrovic, Weiwei Chu, Wenhan Xiong, Wenxin Fu, Whitney Meers, Xavier Martinet, Xiaodong Wang, Xiaoqing Ellen Tan, Xinfeng Xie, Xuchao Jia, Xuewei Wang, Yaelle Goldschlag, Yashesh Gaur, Yasmine Babaei, Yi Wen, Yiwen Song, Yuchen Zhang, Yue Li, Yuning Mao, Zacharie Delpierre Coudert, Zheng Yan, Zhengxing Chen, Zoe Papakipos, Aaditya Singh, Aaron Grattafiori, Abha Jain, Adam Kelsey, Adam Shajnfeld, Adithya Gangidi, Adolfo Victoria, Ahuva Goldstand, Ajay Menon, Ajay Sharma, Alex Boesenberg, Alex Vaughan, Alexei Baevski, Allie Feinstein, Amanda Kallet, Amit Sangani, Anam Yunus, Andrei Lupu, Andres Alvarado, Andrew Caples, Andrew Gu, Andrew Ho, Andrew Poulton, Andrew Ryan, Ankit Ramchandani, Annie Franco, Aparajita Saraf, Arkabandhu Chowdhury, Ashley Gabriel, Ashwin Bharambe, Assaf Eisenman, Azadeh Yazdan, Beau James, Ben Maurer, Benjamin Leonhardi, Bernie Huang, Beth Loyd, Beto De Paola, Bhargavi Paranjape, Bing Liu, Bo Wu, Boyu Ni, Braden Hancock, Bram Wasti, Brandon Spence, Brani Stojkovic, Brian Gamido, Britt Montalvo, Carl Parker, Carly Burton, Catalina Mejia, Changan Wang, Changkyu Kim, Chao Zhou, Chester Hu, Ching-Hsiang Chu, Chris Cai, Chris Tindal, Christoph Feichtenhofer, Damon Civin, Dana Beaty, Daniel Kreymer, Daniel Li, Danny Wyatt, David Adkins, David Xu, Davide Testuggine, Delia David, Devi Parikh, Diana Liskovich, Didem Foss, Dingkan Wang, Duc Le, Dustin Hol-

land, Edward Dowling, Eissa Jamil, Elaine Montgomery, Eleonora Presani, Emily Hahn, Emily Wood, Erik Brinkman, Esteban Arcaute, Evan Dunbar, Evan Smothers, Fei Sun, Felix Kreuk, Feng Tian, Firat Ozgenel, Francesco Caggioni, Francisco Guzmán, Frank Kanayet, Frank Seide, Gabriela Medina Florez, Gabriella Schwarz, Gada Badeer, Georgia Swee, Gil Halpern, Govind Thattai, Grant Herman, Grigory Sizov, Guangyi, Zhang, Guna Lakshminarayanan, Hamid Shojanazeri, Han Zou, Hannah Wang, Hanwen Zha, Haroun Habeeb, Harrison Rudolph, Helen Suk, Henry Aspegren, Hunter Goldman, Ibrahim Damraj, Igor Molybog, Igor Tufanov, Irina-Elena Veliche, Itai Gat, Jake Weissman, James Geboski, James Kohli, Japhet Asher, Jean-Baptiste Gaya, Jeff Marcus, Jeff Tang, Jennifer Chan, Jenny Zhen, Jeremy Reizenstein, Jeremy Teboul, Jessica Zhong, Jian Jin, Jingyi Yang, Joe Cummings, Jon Carvill, Jon Shepard, Jonathan McPhie, Jonathan Torres, Josh Ginsburg, Junjie Wang, Kai Wu, Kam Hou U, Karan Saxena, Karthik Prasad, Kartikay Khan-delwal, Katayoun Zand, Kathy Matosich, Kaushik Veeraraghavan, Kelly Michelena, Keqian Li, Kun Huang, Kunal Chawla, Kushal Lakhotia, Kyle Huang, Lailin Chen, Lakshya Garg, Lavender A, Leandro Silva, Lee Bell, Lei Zhang, Liangpeng Guo, Licheng Yu, Liron Moshkovich, Luca Wehrstedt, Madian Khabza, Manav Avalani, Manish Bhatt, Maria Tsim-poukelli, Martynas Mankus, Matan Hasson, Matthew Lennie, Matthias Reso, Maxim Groshev, Maxim Naumov, Maya Lathi, Meghan Keneally, Michael L. Seltzer, Michal Valko, Michelle Restrepo, Mihir Patel, Mik Vyatskov, Mikayel Samvelyan, Mike Clark, Mike Macey, Mike Wang, Miquel Jubert Her-moso, Mo Metanat, Mohammad Rastegari, Mun-ish Bansal, Nandhini Santhanam, Natascha Parks, Natasha White, Navyata Bawa, Nayan Singhal, Nick Egebo, Nicolas Usunier, Nikolay Pavlovich Laptev, Ning Dong, Ning Zhang, Norman Cheng, Oleg Chernoguz, Olivia Hart, Omkar Salpekar, Ozlem Kalinli, Parkin Kent, Parth Parekh, Paul Saab, Pa-van Balaji, Pedro Rittner, Philip Bontrager, Pierre Roux, Piotr Dollar, Polina Zvyagina, Prashant Ratan-chandani, Pritish Yuvraj, Qian Liang, Rachad Alao, Rachel Rodriguez, Rafi Ayub, Raghotham Murthy, Raghu Nayani, Rahul Mitra, Raymond Li, Rebekkah Hogan, Robin Battey, Rocky Wang, Rohan Mah-eswari, Russ Howes, Ruty Rinott, Sai Jayesh Bondu, Samyak Datta, Sara Chugh, Sara Hunt, Sargun Dhillon, Sasha Sidorov, Satadru Pan, Saurabh Verma, Seiji Yamamoto, Sharadh Ramaswamy, Shaun Lind-say, Shaun Lindsay, Sheng Feng, Shenghao Lin, Shengxin Cindy Zha, Shiva Shankar, Shuqiang Zhang, Shuqiang Zhang, Sinong Wang, Sneha Agar-wal, Soji Sajuyigbe, Soumith Chintala, Stephanie Max, Stephen Chen, Steve Kehoe, Steve Satterfield, Sudarshan Govindaprasad, Sumit Gupta, Sungmin Cho, Sunny Virk, Suraj Subramanian, Sy Choudhury, Sydney Goldman, Tal Remez, Tamar Glaser, Tamara Best, Thilo Kohler, Thomas Robinson, Tianhe Li, Tianjun Zhang, Tim Matthews, Timothy Chou, Tzook Shaked, Varun Vontimitta, Victoria Ajayi, Victoria Montanez, Vijai Mohan, Vinay Satish Kumar, Vishal Mangla, Vitor Albiero, Vlad Ionescu, Vlad Poenaru,

- Vlad Tiberiu Mihailescu, Vladimir Ivanov, Wei Li, Wenchen Wang, Wenwen Jiang, Wes Bouaziz, Will Constable, Xiaocheng Tang, Xiaofang Wang, Xiaojian Wu, Xiaolan Wang, Xide Xia, Xilun Wu, Xinbo Gao, Yanjun Chen, Ye Hu, Ye Jia, Ye Qi, Yenda Li, Yilin Zhang, Ying Zhang, Yossi Adi, Youngjin Nam, Yu, Wang, Yuchen Hao, Yundi Qian, Yuzi He, Zach Rait, Zachary DeVito, Zef Rosnbrick, Zhaoduo Wen, Zhenyu Yang, and Zhiwei Zhao. 2024. [The llama 3 herd of models](#).
- Junfeng Fang, Houcheng Jiang, Kun Wang, Yunshan Ma, Xiang Wang, Xiangnan He, and Tat seng Chua. 2024. [Alphaedit: Null-space constrained knowledge editing for language models](#). *Preprint*, arXiv:2410.02355.
- Hengrui Gu, Kaixiong Zhou, Xiaotian Han, Ninghao Liu, Ruobing Wang, and Xin Wang. 2024a. [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)*, pages 8069–8083, Bangkok, Thailand. Association for Computational Linguistics.
- Jia-Chen Gu, Hao-Xiang Xu, Jun-Yu Ma, Pan Lu, Zhen-Hua Ling, Kai-Wei Chang, and Nanyun Peng. 2024b. [Model editing harms general abilities of large language models: Regularization to the rescue](#). In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing, EMNLP 2024, Miami, FL, USA, November 12-16, 2024*, pages 16801–16819. Association for Computational Linguistics.
- Gautier Izacard, Mathilde Caron, Lucas Hosseini, Sebastian Riedel, Piotr Bojanowski, Armand Joulin, and Edouard Grave. 2022. [Unsupervised dense information retrieval with contrastive learning](#). *Trans. Mach. Learn. Res.*, 2022.
- Qi Li, Xiang Liu, Zhenheng Tang, Peijie Dong, Zeyu Li, Xinglin Pan, and Xiaowen Chu. 2024. [Should we really edit language models? on the evaluation of edited language models](#). *Preprint*, arXiv:2410.18785.
- Kevin Meng, David Bau, Alex Andonian, and Yonatan Belinkov. 2022. [Locating and editing factual associations in GPT](#). In *Advances in Neural Information Processing Systems 35: Annual Conference on Neural Information Processing Systems 2022, NeurIPS 2022, New Orleans, LA, USA, November 28 - December 9, 2022*.
- 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.
- OpenAI, Josh Achiam, Steven Adler, Sandhini Agarwal, Lama Ahmad, Ilge Akkaya, Florencia Leoni Aleman, Diogo Almeida, Janko Altmenschmidt, Sam Altman, Shyamal Anadkat, Red Avila, Igor Babuschkin, Suchir Balaji, Valerie Balcom, Paul Baltescu, Haiming Bao, Mohammad Bavarian, Jeff Belgum, Irwan Bello, Jake Berdine, Gabriel Bernadett-Shapiro, Christopher Berner, Lenny Bogdonoff, Oleg Boiko, Madelaine Boyd, Anna-Luisa Brakman, Greg Brockman, Tim Brooks, Miles Brundage, Kevin Button, Trevor Cai, Rosie Campbell, Andrew Cann, Brittany Carey, Chelsea Carlson, Rory Carmichael, Brooke Chan, Che Chang, Fotis Chantzis, Derek Chen, Sully Chen, Ruby Chen, Jason Chen, Mark Chen, Ben Chess, Chester Cho, Casey Chu, Hyung Won Chung, Dave Cummings, Jeremiah Currier, Yunxing Dai, Cory Decareaux, Thomas Degry, Noah Deutsch, Damien Deville, Arka Dhar, David Dohan, Steve Dowling, Sheila Dunning, Adrien Ecoffet, Atty Eleti, Tyna Eloundou, David Farhi, Liam Fedus, Niko Felix, Simón Posada Fishman, Juston Forte, Isabella Fulford, Leo Gao, Elie Georges, Christian Gibson, Vik Goel, Tarun Gogineni, Gabriel Goh, Rapha Gontijo-Lopes, Jonathan Gordon, Morgan Grafstein, Scott Gray, Ryan Greene, Joshua Gross, Shixiang Shane Gu, Yufei Guo, Chris Hallacy, Jesse Han, Jeff Harris, Yuchen He, Mike Heaton, Johannes Heidecke, Chris Hesse, Alan Hickey, Wade Hickey, Peter Hoeschele, Brandon Houghton, Kenny Hsu, Shengli Hu, Xin Hu, Joost Huizinga, Shantanu Jain, Shawn Jain, Joanne Jang, Angela Jiang, Roger Jiang, Haozhun Jin, Denny Jin, Shino Jomoto, Billie Jonn, Heewoo Jun, Tomer Kaftan, Łukasz Kaiser, Ali Kamali, Ingmar Kanitscheider, Nitish Shirish Keskar, Tabarak Khan, Logan Kilpatrick, Jong Wook Kim, Christina Kim, Yongjik Kim, Jan Hendrik Kirchner, Jamie Kiros, Matt Knight, Daniel Kokotajlo, Łukasz Kondraciuk, Andrew Kondrich, Aris Konstantinidis, Kyle Kosic, Gretchen Krueger, Vishal Kuo, Michael Lampe, Ikai Lan, Teddy Lee, Jan Leike, Jade Leung, Daniel Levy, Chak Ming Li, Rachel Lim, Molly Lin, Stephanie Lin, Mateusz Litwin, Theresa Lopez, Ryan Lowe, Patricia Lue, Anna Makanju, Kim Malfacini, Sam Manning, Todor Markov, Yaniv Markovski, Bianca Martin, Katie Mayer, Andrew Mayne, Bob McGrew, Scott Mayer McKinney, Christine McLeavey, Paul McMillan, Jake McNeil, David Medina, Aalok Mehta, Jacob Menick, Luke Metz, Andrey Mishchenko, Pamela Mishkin, Vinnie Monaco, Evan Morikawa, Daniel Mossing, Tong Mu, Mira Murati, Oleg Murk, David Mély, Ashvin Nair, Reiichiro Nakano, Rajeev Nayak, Arvind Neelakantan, Richard Ngo, Hyeonwoo Noh, Long Ouyang, Cullen O’Keefe, Jakub Pachocki, Alex Paino, Joe Palermo, Ashley Pantuliano, Giambattista Parascandolo, Joel Parish, Emy Parparita, Alex Passos, Mikhail Pavlov, Andrew Peng, Adam Perialman, Filipe de Avila Belbute Peres, Michael Petrov, Henrique Ponde de Oliveira Pinto, Michael, Pokorny, Michelle Pokrass, Vitchyr H. Pong, Tolly Powell, Alethea Power, Boris Power, Elizabeth Proehl, Raul Puri, Alec Radford, Jack Rae, Aditya Ramesh, Cameron Raymond, Francis Real, Kendra Rimbach, Carl Ross, Bob Rotsted, Henri Roussez, Nick Ryder, Mario Saltarelli, Ted Sanders, Shibani Santurkar, Girish Sastry, Heather Schmidt, David Schnurr, John Schulman, Daniel Selsam, Kyla Sheppard, Toki Sherbakov, Jessica Shieh, Sarah Shoker, Pranav

- Shyam, Szymon Sidor, Eric Sigler, Maddie Simens, Jordan Sitkin, Katarina Slama, Ian Sohl, Benjamin Sokolowsky, Yang Song, Natalie Staudacher, Felipe Petroski Such, Natalie Summers, Ilya Sutskever, Jie Tang, Nikolas Tezak, Madeleine B. Thompson, Phil Tillet, Amin Tootoonchian, Elizabeth Tseng, Preston Tuggle, Nick Turley, Jerry Tworek, Juan Felipe Cerón Uribe, Andrea Vallone, Arun Vijayvergiya, Chelsea Voss, Carroll Wainwright, Justin Jay Wang, Alvin Wang, Ben Wang, Jonathan Ward, Jason Wei, CJ Weinmann, Akila Welihinda, Peter Welinder, Jiayi Weng, Lilian Weng, Matt Wiethoff, Dave Willner, Clemens Winter, Samuel Wolrich, Hannah Wong, Lauren Workman, Sherwin Wu, Jeff Wu, Michael Wu, Kai Xiao, Tao Xu, Sarah Yoo, Kevin Yu, Qiming Yuan, Wojciech Zaremba, Rowan Zellers, Chong Zhang, Marvin Zhang, Shengjia Zhao, Tianhao Zheng, Juntang Zhuang, William Zhuk, and Barret Zoph. 2023. [Gpt-4 technical report](#).
- 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 2024, Boise, ID, USA, October 21-25, 2024*, pages 2056–2066. ACM.
- Saba Sturua, Isabelle Mohr, Mohammad Kalim Akram, Michael Günther, Bo Wang, Markus Krimmel, Feng Wang, Georgios Mastrapas, Andreas Koukounas, Nan Wang, and Han Xiao. 2024. [jina-embeddings-v3: Multilingual embeddings with task lora](#). *Preprint*, arXiv:2409.10173.
- Weihang Su, Changyue Wang, Qingyao Ai, Yiran Hu, Zhijing Wu, Yujia Zhou, and Yiqun Liu. 2024. [Unsupervised real-time hallucination detection based on the internal states of large language models](#). In *Findings of the Association for Computational Linguistics ACL 2024*, pages 14379–14391, Bangkok, Thailand and virtual meeting. Association for Computational Linguistics.
- Qwen Team. 2024. [Qwen2.5: A party of foundation models](#).
- Changyue Wang, Weihang Su, Qingyao Ai, and Yiqun Liu. 2024a. [Knowledge editing through chain-of-thought](#). *Preprint*, arXiv:2412.17727.
- Changyue Wang, Weihang Su, Yiran Hu, Qingyao Ai, Yueyue Wu, Cheng Luo, Yiqun Liu, Min Zhang, and Shaoping Ma. 2024b. [Lekube: A knowledge update benchmark for legal domain](#). In *Proceedings of the 2024 Annual International ACM SIGIR Conference on Research and Development in Information Retrieval in the Asia Pacific Region*, pages 175–185.
- Xiaozhi Wang, Tianyu Gao, Zhaocheng Zhu, Zhengyan Zhang, Zhiyuan Liu, Juanzi Li, and Jian Tang. 2021. [Kepler: A unified model for knowledge embedding and pre-trained language representation](#). *Transactions of the Association for Computational Linguistics*, 9:176–194.
- Yiwei Wang, Muhao Chen, Nanyun Peng, and Kai-Wei Chang. 2024c. [Deepedit: Knowledge editing as decoding with constraints](#). *Preprint*, arXiv:2401.10471.
- Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Brian Ichter, Fei Xia, Ed H. Chi, Quoc V. Le, and Denny Zhou. 2022. [Chain-of-thought prompting elicits reasoning in large language models](#). In *Advances in Neural Information Processing Systems 35: Annual Conference on Neural Information Processing Systems 2022, NeurIPS 2022, New Orleans, LA, USA, November 28 - December 9, 2022*.
- Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Remi Louf, Morgan Funtowicz, Joe Davison, Sam Shleifer, Patrick von Platen, Clara Ma, Yacine Jernite, Julien Plu, Canwen Xu, Teven Le Scao, Sylvain Gugger, Mariama Drame, Quentin Lhoest, and Alexander Rush. 2020. [Transformers: State-of-the-art natural language processing](#). In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 38–45, Online. Association for Computational Linguistics.
- Zexuan Zhong, Zhengxuan Wu, Christopher 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*, pages 15686–15702, Singapore. Association for Computational Linguistics.

## A Pseudocode Description of DeckER-BoN

The pseudocode of DeckER-BoN is shown in Algorithm 1.

## B Details of Baselines

**MEMIT (Meng et al., 2023) and AlphaEdit (Fang et al., 2024):** We adopt the official implementation of AlphaEdit, setting the target editing layers to 4-8, with a learning rate of 0.1 and a weight decay of 0.5. The nullspace threshold for AlphaEdit is set to 0.02.

**Mello (Zhong et al., 2023):** We adhere to the original paper’s settings, decomposing up to 4 sub-problems.

**PokeMQA (Gu et al., 2024a):** We follow the original paper’s settings, decomposing up to 5 sub-problems. As the original repository does not provide entities for the new dataset, we allow the model to extract problem entities in a similar format.

**DeepEdit (Wang et al., 2024c):** In line with the official implementation, we set the search width limit to 2 and the depth limit to 9.



---

**Algorithm 1** DecKER

---

**Input:** Question  $Q$ , LLM  $M$ , Knowledge Edits  $\mathcal{E}$ , Hyperparameters  $\alpha, \beta$ , Sample Size  $N$

**Output:** Final Answer  $A$

```
1: Step 1: Generate Masked Reasoning Paths and Entity Types
2: for  $i \leftarrow 1$  to  $N$  do
3:    $R_Q^{(i)}, \{t_1, t_2, \dots\} \leftarrow \text{GENERATEMASKEDREASONINGPATH}(M, Q)$ 
4: end for
5: Step 2: Fill Masked Entities
6: for each masked reasoning path  $R_Q^{(i)}$  do
7:    $\text{PathLength} \leftarrow \text{length}(R_Q^{(i)})$ 
8:    $P_Q^{(i)} \leftarrow [R_Q^{(i)}[0]]$ 
9:   for  $j$  in  $[0, 1, \dots, \text{PathLength} - 1]$  do
10:     $s_k \leftarrow P_Q^{(i)}[j]$  ▷ The current masked step
11:     $\text{MaskTag} \leftarrow \text{GetMaskTag}(s_k)$ 
12:     $\text{Query} \leftarrow \text{PrevFilledEntity}() + \text{RemoveMaskTag}(s_k)$ 
13:     $\{(d_1, \text{score}_1), (d_2, \text{score}_2), \dots\} \leftarrow \text{RETRIEVE}(\mathcal{E}, \text{Query})$ 
14:     $\text{IsEdited} \leftarrow \text{False}$ 
15:    if  $(\text{score}_1 > \alpha) = (\text{score}_1 - \text{score}_2 > \beta)$  then
16:       $\text{IsEdited} \leftarrow (\text{score}_1 > \alpha)$ 
17:    else
18:       $\text{IsEdited} \leftarrow \text{CONFLICTCHECK}(M, s_k, \text{OriginalFact}(d_1))$ 
19:    end if
20:    if  $\text{IsEdited}$  then
21:       $e_k \leftarrow \text{GetObjectEntity}(d_1)$ 
22:    else
23:       $e_k \leftarrow M(\text{"Fill \{MaskTag\} with type } t_k\text{"})$  ▷ LLM-based completion
24:    end if
25:    Replace all  $\text{MaskTag}$  in  $R_Q^{(i)}$  with  $e_k$ 
26:    if  $\text{MaskTag} = \text{"[MASK ANS]"}$  then
27:       $A^{(i)} \leftarrow e_k$ 
28:      break
29:    end if
30:     $P_Q^{(i)} \leftarrow P_Q^{(i)} + [R_Q^{(i)}[j + 1]]$ 
31:  end for
32: end for
33: Step 3: Select the Best Sample
34:  $\text{RPP Scores} \leftarrow \text{COMPUTEAVGPE}(M, \{R_Q^{(i)}\})$ 
35:  $\text{KI Scores} \leftarrow []$ 
36:  $\text{Answers} \leftarrow []$ 
37:  $\text{TopPaths} \leftarrow \text{Top 50\% } P_Q \text{ by RPP Scores}$ 
38: for each filled reasoning path  $P_Q^{(i)}$  in  $\text{TopPaths}$  do
39:    $\text{KI Scores.append}(\frac{1}{\text{length}(P_Q^{(i)})} \sum \mathbb{1}[P_M(\text{"yes"}|\text{"Is } e_j \text{ type } t_j?\text{"}) > P_M(\text{"no"}|\text{"Is } e_j \text{ type } t_j?\text{"})])$ 
40:    $\text{Answers.append}(A^{(i)})$ 
41: end for
42:  $A \leftarrow \text{Answers}[\arg \max \text{KI Scores}]$ 
```

---

**RAE (Shi et al., 2024):** We use the code provided by the official implementation and adjust its retrieval process as described in subsection 5.4 to ensure a fair comparison.

**EditCoT (Wang et al., 2024a):** We adhere to the original paper’s settings, conducting up to 4 rounds of CoT editing. The retrieval process is configured to operate solely on edit memory, as one of the official implementations, to ensure a fair comparison.

For all ICE methods, we use the official prompt templates and adapt them to fit the chat template format of each LLM. The LLMs we used are implemented by Huggingface Transformers library (Wolf et al., 2020).

## C Details of Efficiency Experiments

All efficiency experiments are conducted on a single NVIDIA A100 (40G) GPU. We exclude network latency from the reported inference time of RAE. In accordance with the official implementation, RAE employs float32 floating point numbers, while other methods use float16, as we observe a considerable performance degradation in RAE with half-precision. In our implementation, many operations in DecKER use batch processing, including the sampling of multiple masked reasoning paths and parallel filling processes.

## D Further Ablation Studies

### D.1 Performance with Different Edit Batch Sizes

We define Edit Batch Size as the number of questions in an editing batch. We conduct experiments on MQuAKE-300 and Llama-3.1-8B-Instruct, with batch sizes of 1, 10, and 100, and compare DecKER-Base with EditCoT. The results are shown in Figure 7. We observe that both methods experience a decline in performance as the batch size increases. However, our method is more robust and exhibits lower sensitivity to editbatch compared to EditCoT.

### D.2 Impact of Sample Number on DecKER-BoN

Table 4 illustrates the effect of the number of sampled masked paths ( $N$ ). Performance initially improves and then levels off as  $N$  increases. This indicates the effectiveness of the two-round evaluation while also suggesting that a larger  $N$  is not

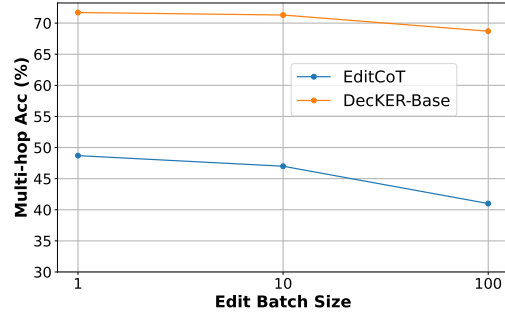


Figure 7: Edit Batch Size vs. Performance. The model is Llama-3.1-8B-Instruct and the dataset is MQuAKE-300.

Table 4: The relationship between sampling paths and performance in DecKER-BoN. The model is Llama-3.1-8B-Instruct and the dataset is MQuAKE-300.

<b>DecKER-Base</b>	66.0			
<b>DecKER-BoN</b>	$N = 2$	$N = 6$	$N = 10$	$N = 14$
	67.7	70.3	69.3	69.7

necessarily better, as an increased number of candidates can introduce noise and impact the selection process.

## E Case Study

We showcase two examples of Meta-Llama-3.1-8B-Instruct from MQuAKE-CF-3k-v2. Table 5 illustrates a 2-hop question involving two edits. None of the other three baselines can solve this problem. Although PokeMQA correctly infers "basketball player," the presence of editing knowledge in the context causes the model to revise its answer. RAE fails to accurately complete the retrieval in this example. EditCoT fails to preserve the reasoning framework of the original CoT, resulting in a disruption of the reasoning process, revealing its black-box nature. Our method closely matches the ground truth, with reasoning unaffected by editing knowledge, and accurately arrives at the final answer.

Table 6 is a 4-hop question involving three edits. While PokeMQA identifies the final answer "Chiavari" during the reasoning process, the model continues to decompose the next sub-question, trying to assess whether Chiavari is the capital of South Korea because it is editing knowledge. RAE’s retrieval error occurs in the intermediate step, leading to an incorrect result, demonstrating its lack of robustness. EditCoT’s final answer is correct, but its

Table 5: An example of a 2-hop problem on Meta-Llama-3.1-8B-Instruct. In the "Results" column, the red sections indicate where reasoning fails. We omit the display of intermediate steps, such as retrieval scores.

<b>Question:</b> What is the occupation of the chairperson of the Palestine Liberation Organization?		
<b>Related Edits:</b> The chairperson of Palestine Liberation Organization is Mahmoud Abbas → Regina Ip Regina Ip works in the field of politician → basketball player		
<b>Ground Truth Reasoning Path:</b> The chairperson of Palestine Liberation Organization is Regina Ip. Regina Ip works in the field of basketball player.		
Method	Results	Correctness
PokeMQA	Entity of Question: Palestine Liberation Organization, an organization. Subquestion: Who is the chairperson of the Palestine Liberation Organization? Generated answer: The chairperson of Palestine Liberation Organization is Regina Ip. According to Generated answer, the entity of Subquestion is: Regina Ip Subquestion: What is the occupation of Regina Ip? Generated answer: Regina Ip works in the field of basketball player. According to Generated answer, the entity of Subquestion is: basketball player However, the chairperson of Palestine Liberation Organization is actually Mahmoud Abbas	Wrong
RAE	<b>Retrieved Facts:</b> Palestine Liberation Organization has chairperson named as Regina Ip. Regina Ip has the religion of Catholicism. Catholic Church has the religion of Catholicism. Catholic Church has chairperson named as Pope.  <b>Answer:</b> there is no information about regina ip being the chairperson of the palestine liberation organization	Wrong
EditCoT	<b>Original CoT:</b> The chairperson of the Palestine Liberation Organization is Mahmoud Abbas. Mahmoud Abbas is a public figure associated with the Palestinian National Authority. The Palestinian National Authority is a government entity, and its leaders are typically involved in politics. Therefore, Mahmoud Abbas, as the chairperson of the Palestine Liberation Organization, is likely a politician.  <b>Edited CoT:</b> The chairperson of the Palestine Liberation Organization is Regina Ip.	Wrong
Decker	<b>Masked Reasoning Path:</b> The chairperson of the Palestine Liberation Organization is [MASK 1]. The occupation of [MASK 1] is [MASK ANS].  <b>Filled Reasoning Path:</b> The chairperson of the Palestine Liberation Organization is Regina Ip. The occupation of Regina Ip is basketball player.	Right

reasoning process does not align with the ground truth. Our method accurately completes the reasoning and filling.

## F Licensing

Qwen2.5-7B-Instruct and Qwen2.5-14B-Instruct is released under the Apache License 2.0. Meta-Llama-3.1-8B-Instruct is released under the LLAMA 3.1 COMMUNITY LICENSE. MEMIT, PokeMQA, and Mello are released under the MIT license. Contriever is released under the CC BY-SA 4.0 License.

The datasets MQuAKE and RippleEdits are released under the MIT license. The objective of this research is to conduct an academic investigation, which is consistent with the stipulations outlined

in the associated licensing agreement.

## G Prompt Templates

### G.1 Prompt Templates in Analytical Experiments

We provide GPT-4o-mini with the following system prompt:

*"Extract the relation from the knowledge triples (subject, relation, object) involved in each sentence, and return a list of relations that is equal in length to the given list of sentences. You only need to provide a JSON structure that contains one key relations, whose value is the list of relations."*

In the user input section, we provide a list, with each element representing a step in a reasoning path. We utilize GPT-4o-mini's format decoding

function to ensure the output is a dictionary with the key "relations" and the value being the list of extracted relations.

## **G.2 Prompt Templates of DecKER**

Table 7, Table 8, Table 9, and Table 10 present the prompt templates used by DecKER. We provide a 5-shot prompt template, with the blue sections in the tables corresponding to the roles in the LLM’s chat template.



Table 6: An example of a 4-hop problem on Meta-Llama-3.1-8B-Instruct. In the "Results" column, the red sections indicate where reasoning fails. We omit the display of intermediate steps, such as retrieval scores.

<b>Question:</b> Which city is the capital of the country where the CEO of the developer of iOS 9 holds citizenship?		
<b>Related Edits:</b> The developer of iOS 9 is <u>Apple Inc.</u> → <u>NeXT Computer, Inc.</u> Steve Jobs holds citizenship in the country of <u>United States of America</u> → <u>South Korea</u> The capital of South Korea is <u>Seoul</u> → <u>Chiavari</u>		
<b>Ground Truth Reasoning Path:</b> iOS 9 was developed by NeXT Computer, Inc. The CEO of NeXT Computer, Inc. is Steve Jobs. Steve Jobs is a citizen of South Korea. The capital of South Korea is Chiavari.		
Method	Results	Correctness
PokeMQA	Entity of Question: iOS 9, a subclass of iOS. Subquestion: Which company is iOS 9 developed by? Generated answer: iOS 9 was developed by NeXT Computer, Inc.. According to Generated answer, the entity of Subquestion is: NeXT Computer, Inc. Subquestion: Who is the CEO of NeXT Computer, Inc.? Generated answer: The CEO of NeXT Computer, Inc. is Steve Jobs. According to Generated answer, the entity of Subquestion is: Steve Jobs Subquestion: What is the country of citizenship of Steve Jobs? Generated answer: Steve Jobs is a citizen of South Korea. According to Generated answer, the entity of Subquestion is: South Korea Subquestion: Which city is the capital of South Korea? Generated answer: The capital of South Korea is Chiavari. According to Generated answer, the entity of Subquestion is: Chiavari <b>Subquestion: Is Chiavari the capital of South Korea?</b> <b>Generated answer:</b>	Wrong
RAE	<b>Retrieved Facts:</b> iOS 9 was developed by NeXT. NeXT is located in the country of Switzerland. Switzerland has its capital city named as Bern. Bern capital of Switzerland.  <b>Answer:</b> <b>there is no information about the ceo of next</b>	Wrong
EditCoT	<b>Original CoT:</b> The CEO of Apple is Tim Cook. Tim Cook holds citizenship in the United States. The capital of the United States is Washington, D.C.  <b>Edited CoT:</b> The CEO of the developer of iOS 9 is Steve Jobs. Steve Jobs is a citizen of South Korea. The capital of South Korea is Chiavari.	Right
DecKER	<b>Masked Reasoning Path:</b> The developer of iOS 9 is [MASK 1]. The CEO of [MASK 1] is [MASK 2]. [MASK 2] holds citizenship in the country of [MASK 3]. The capital of [MASK 3] is [MASK ANS].  <b>Filled Reasoning Path:</b> The developer of iOS 9 is NeXT Computer, Inc.. The CEO of NeXT Computer, Inc. is Steve Jobs. Steve Jobs holds citizenship in the country of South Korea. The capital of South Korea is Chiavari.	Right

Table 7: The prompt template of masked reasoning path generation.

<p><b>System:</b> Generate a detailed chain of thought with each reasoning step labeled as [STEP]. Replace any specific knowledge, entities, dates, or locations that are not in the text of question with [MASK]. Focus only on the logical steps and flow. And then include the category of each [MASK *] in order in the chain of thought as [CATEGORY], to clarify the type of entity that should be filled in.</p> <p><b>User:</b> Question: What is the capital of the country where Plainfield Town Hall is located?</p> <p><b>Assistant:</b> [STEP] Plainfield Town Hall is located in the country of [MASK 1]. [STEP] The capital of [MASK 1] is [MASK ANS]. [CATEGORY] [MASK 1] [country] [MASK ANS] [city]</p> <p><b>User:</b> Question: In which country is the company that created Nissan 200SX located?</p> <p><b>Assistant:</b> [STEP] Nissan 200SX was created by [MASK 1]. [STEP] [MASK 1] is located in the country of [MASK ANS]. [CATEGORY] [MASK 1] [company] [MASK ANS] [country]</p> <p><b>User:</b> Question: Which continent is the country where the director of My House Husband: Ikaw Na! was educated located in?</p> <p><b>Assistant:</b> [STEP] The director of My House Husband: Ikaw Na! is [MASK 1]. [STEP] [MASK 1] was educated at [MASK 2]. [STEP] [MASK 2] is located in the country of [MASK 3]. [STEP] [MASK 3] is located in the continent of [MASK ANS]. [CATEGORY] [MASK 1] [person] [MASK 2] [school] [MASK 3] [country] [MASK ANS] [continent]</p> <p><b>User:</b> Question: Who is the spouse of the US president?</p> <p><b>Assistant:</b> [STEP] The US president is [MASK 1]. [STEP] The spouse of [MASK 1] is [MASK ANS]. [CATEGORY] [MASK 1] [person] [MASK ANS] [person]</p> <p><b>User:</b> Question: Who has ownership of the developer of the Chevrolet Corvette (C4)?</p> <p><b>Assistant:</b> [STEP] The developer of Chevrolet Corvette (C4) is [MASK 1]. [STEP] [MASK 1] is owned by [MASK ANS]. [CATEGORY] [MASK 1] [company] [MASK ANS] [person]</p> <p><b>User:</b> Question: <i>{A question}</i></p> <p><b>Assistant:</b></p>
---

Table 8: The prompt template of conflict detection by LLMs.

<p><b>System:</b> Determine whether the provided Fact sentence supports filling the MASKed part in the given Sentence. Only output Yes or No.</p> <p><b>User:</b> Fact: Club Nouveau is a group. Sentence: Club Nouveau originated from [MASK 1]. <b>Assistant:</b> No</p> <p><b>User:</b> Fact: Paris is the capital of France. Sentence: The capital of France is [MASK ANS]. <b>Assistant:</b> Yes</p> <p><b>User:</b> Fact: The Venus de Milo is located in the Louvre Museum. Sentence: The Mona Lisa is located in the [MASK 2] Museum. <b>Assistant:</b> No</p> <p><b>User:</b> Fact: The first person to get to the South Pole is Roald Amundsen. Sentence: The first person to walk on the moon was [MASK 3]. <b>Assistant:</b> No</p> <p><b>User:</b> Fact: The Python programming language was created by Guido van Rossum. Sentence: The creator of Python is [MASK 1]. <b>Assistant:</b> Yes</p> <p><b>User:</b> Fact: <i>{A retrieved fact.}</i> Sentence: <i>{A masked step.}</i> <b>Assistant:</b></p>
--

Table 9: The prompt template of entity filling.

<p><b>System:</b> Given the first masked entity without any analysis. The answer you give must match the given type.</p> <p><b>User:</b> Type of the masked entity: country. Sentence: Club Nouveau originated from [MASK 1]. <b>Assistant:</b> United States</p> <p><b>User:</b> Type of the masked entity: country. Sentence: The country where the creator of the Chevrolet Corvette (C4) is located is [MASK ANS]. <b>Assistant:</b> United States</p> <p><b>User:</b> Type of the masked entity: place. Sentence: The Mona Lisa is located in the [MASK 2] Museum. <b>Assistant:</b> Louvre</p> <p><b>User:</b> Type of the masked entity: person. Sentence: The first person to walk on the moon was [MASK 3]. <b>Assistant:</b> Neil Armstrong</p> <p><b>User:</b> Type of the masked entity: person. Sentence: The creator of Python is [MASK 1]. <b>Assistant:</b> Guido van Rossum</p> <p><b>User:</b> Type of the masked entity: <i>{Type of the masked entity.}</i> Sentence: <i>{A masked sentence.}</i> <b>Assistant:</b></p>
---

Table 10: The prompt template of determining if entity types match, i.e. the KI Eval.

<p><b>System:</b> You are given an entity along with its supposed types. Your task is to determine whether the entity matches the type it has been assigned. Only output Yes or No. Yes if the entity matches the type, No otherwise.</p> <p><b>User:</b> Entity: New York Assigned Type: country <b>Assistant:</b> No</p> <p><b>User:</b> Entity: Apple Inc. Assigned Type: company <b>Assistant:</b> Yes</p> <p><b>User:</b> Entity: Steve Jobs Assigned Type: city <b>Assistant:</b> No</p> <p><b>User:</b> Entity: baseball Assigned Type: position <b>Assistant:</b> No</p> <p><b>User:</b> Entity: USD Assigned Type: currency <b>Assistant:</b> Yes</p> <p><b>User:</b> Entity: <i>{An entity.}</i> Assigned Type: <i>{A type.}</i> <b>Assistant:</b></p>
--