Multi-Hop Knowledge Editing via Critic-Guided Multi-Agent Reasoning

Xudong Li[1], **Yuhang Tian**[1], **Dandan Song**[1*], **Zhijing Wu**[1], **Shuhao Zhang**[3], **Jun Yang**[1], **Yongyu Huo**[1], **Changzhi Zhou**[1], **Xinyu Zhang**[1], **Chenhao Li**[1], **Huipeng Ma**[1], **Luan Zhang**[1], **Yan Xu**[1], **Qian Liu**[2]

[1]Beijing Institute of Technology
[2]School of Science, University of Auckland
[3]Huazhong University of Science and Technology
demmon718@gmail.com, sdd@bit.edu.cn

Abstract

Knowledge within large language models (LLMs) inevitably lags behind an evolving world, motivating knowledge editing methods that update facts without expensive retraining. In multi-hop knowledge editing, models must not only recall updated facts but also correctly propagate them through multi-hop reasoning chains. However, most existing approaches rely on unidirectional, feed-forward pipelines, decomposing questions and retrieving edited facts in a rigid hop-wise sequence. This design is brittle: a minor retrieval error or logical mismatch at an early hop can become a *silent failure* that cascades to the final answer without an explicit recovery mechanism. To address this limitation, we propose **C**ritic-Guided Multi-**A**gent **R**easoning for Knowledge **E**diting (**CARE**), a framework for closed-loop post-edit reasoning. A CRITIC agent performs chain-level verification by checking both global coherence and step-wise correctness, and triggers bounded backtracking for iterative self-correction, while a SELECTOR agent supplies high-fidelity, low-noise candidate pools from the edit store to enable effective revision. Experiments on MQuAKE-2002 and MQuAKE-hard demonstrate that CARE effectively mitigates error propagation, achieving a new state-of-the-art.

1 Introduction

Large Language Models (LLMs) have shown strong reasoning capabilities (He et al., 2025; Cheng et al., 2025) but rely on static parametric knowledge that inevitably lags behind an evolving world (Zhu et al., 2025). This mismatch often leads to outdated or incorrect answers, motivating knowledge editing (KE) methods that update facts without expensive retraining (Meng et al., 2022). While early KE work focused on isolated, single-hop updates, recent research has shifted to the more challenging multi-hop knowledge editing (MHKE), where models must propagate edited facts through multi-hop reasoning chains rather than merely recall them (Zhong et al., 2023; Gu et al., 2024). To tackle MHKE, most state-of-the-art approaches adopt parameter-preserving strategies, retrieving edited knowledge at inference time from an external store (Liu et al., 2025). They typically decompose multi-hop queries into hop-wise sub-questions and retrieve relevant edited facts step by step. Recent work further suggests that knowledge-graph-based edit stores provide more structured and controllable support for such compositional reasoning than unstructured text memories (Lu et al., 2025; Chen et al., 2024; Fu et al., 2025).

Despite progress in decomposition and retrieval, the prevailing inference paradigm remains unidirectional and feed-forward (Figure 1A). In this setting, subtle deviations (e.g., retrieval or logic

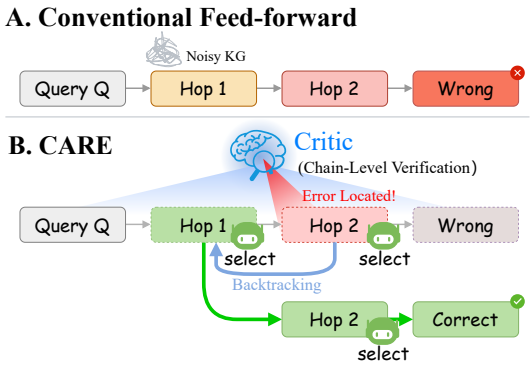


Figure 1: Conventional feed-forward multi-hop reasoning vs. CARE. (**A**) Feed-forward pipelines follow a fixed hop order; early noisy retrieval from the edit store can cascade into a wrong final answer. (**B**) CARE locates the error hop via a CRITIC and performs bounded backtracking, reselecting a better fact from the post-edit graph to recover the correct answer.

*Corresponding Author

Findings of the Association for Computational Linguistics: ACL 2026, pages 17259–17276

errors) often do not surface immediately, instead becoming *silent failures* that propagate downstream (Wang et al., 2024c). As subsequent hops condition on this compromised state, errors cascade unchecked, ultimately corrupting the final prediction without an explicit mechanism to locate and recover from the failure.

We address this limitation with **C**ritic-Guided Multi-**A**gent **R**easoning for Knowledge **E**diting (**CARE**), a closed-loop framework for post-edit reasoning (Figure 1B). CARE replaces one-pass, feed-forward inference with a feedback loop that checks and selectively revises the reasoning chain. A CRITIC agent performs chain-level verification by checking global coherence and step-wise correctness, pinpoints the breaking hop, and triggers bounded backtracking for iterative self-correction. Backtracking alone cannot produce a correct hop; it only enables reselection at the failure point, so recovery depends on the availability of reliable alternatives. To improve this recoverability, a SELECTOR agent retrieves high-fidelity, low-noise candidate pools from the edit store, making it more likely to substitute a corrected hop instead of carrying errors forward. For graph-based edit stores built from natural-language edit descriptions, we apply Edit-Anchored Triple Refinement (EATR) during store construction to reduce extraction noise. EATR uses the edit anchor triple as a constraint, keeping only candidate triples whose entities align with the anchor and whose relation is consistent with it, yielding a cleaner candidate pool for retrieval and revision. Extensive experiments demonstrate that CARE effectively mitigates error propagation and achieves state-of-the-art performance.

Our contributions are summarized as follows:

- We propose CARE, a critic-guided closed-loop framework: a CRITIC verifies the chain and triggers bounded backtracking, while a SELECTOR provides reliable alternatives.
- We propose EATR to denoise extracted triples with anchor-consistent endpoint alignment and relation consistency, yielding a cleaner post-edit graph for retrieval and revision.
- We achieve state-of-the-art performance on MQuAKE-2002 and MQuAKE-hard, demonstrating effective mitigation of error propagation in multi-hop knowledge editing.

2 Related Work

KE aims to efficiently update an LLM's factual memory without full retraining (Wang et al., 2024b; Zhang et al., 2024b; Durrani et al., 2025), which is often impractical at scale. Existing KE methods are commonly grouped into parameter-editing approaches (Meng et al., 2023; Mitchell et al., 2022a; Meng et al., 2022) (e.g., locating and modifying internal weights as in ROME/MEMIT) and parameter-preserving approaches that keep the backbone frozen and instead condition generation on externally stored edits (Mitchell et al., 2022b; Atri et al., 2025). MHKE is substantially harder because edits must propagate through dependent reasoning steps rather than being recalled in isolation. Early MHKE methods like MeLLo (Zhong et al., 2023) perform iterative decomposition-and-retrieval over edited facts, while subsequent work such as DeepEdit (Wang et al., 2024c) frames editing as constrained decoding to encourage coherent multi-hop reasoning and PokeMQA (Gu et al., 2024) decouples decomposition, answer generation, and conflict handling to reduce subtask interference. More recently, IRAKE (Liu et al., 2025) identifies "edit skipping" in multi-hop inference and mitigates it by retrieving candidate edits and prior decomposition records before generating sub-questions.

To better exploit relational structure, recent work builds graph-structured edit stores for neighborhood and path traversal (Zhang et al., 2024a). Methods like GMeLLo (Chen et al., 2024) integrate a knowledge graph memory for MHKE, and KEDKG (Lu et al., 2025) constructs a dynamic edited KG with fine-grained entity/relation filtering to ground generation. Graph-structured edit stores enable path-based retrieval, but most graph-based MHKE methods still follow a one-pass, hop-by-hop, feed-forward inference routine. In this paradigm, intermediate hops are rarely verified against the edited world, so early deviations (e.g., noisy retrieval or minor mismatches) can become *silent failures* that propagate downstream. As a result, feed-forward pipelines often fail to localize and revise early-hop errors, leading to incorrect final answers. To address this limitation, we propose CARE, a critic-guided closed-loop framework for post-edit multi-hop reasoning, where a CRITIC verifies chain-level coherence and triggers bounded backtracking for self-correction.

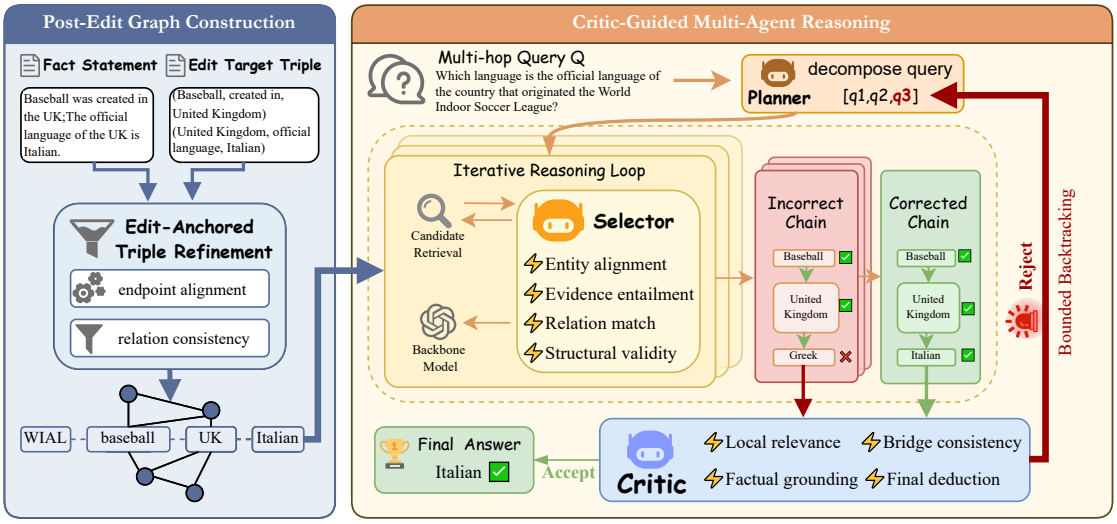


Figure 2: The CARE framework. (1) Post-edit graph construction with EATR for triple denoising. (2) Critic-guided multi-agent reasoning with chain verification and bounded backtracking.

3 Methodology

3.1 Problem Setting

Following prior work (Zhong et al., 2023; Gu et al., 2024; Shi et al., 2024), we represent each factual knowledge item as a triplet (s, r, o), where s, r, and o denote the subject, relation, and object, respectively. A factual edit updates the object from o to o^* while keeping (s, r) unchanged, denoted as $e = (s, r, o \rightarrow o^*)$. Multiple edits are stored in an edited memory $\mathcal{E} = \{e_1, \ldots, e_m\}$. In this work, we focus on the challenging multi-hop scenario (e.g., MQuAKE), where a question Q is answered by traversing a sequence of interdependent facts, denoted as a reasoning chain $C = [(s_1, r_1, o_1), \ldots, (s_H, r_H, o_H)]$, where H is the number of hops. In this chain, the object of hop j acts as the subject for hop $j{+}1$ (i.e., $o_j = s_{j+1}$), with o_H being the original answer. The modification of an intermediate fact leads to a post-edit chain $C^* = [(s_1, r_1, o_1), \ldots, (s_i, r_i, o_i^*), \ldots, (s_H^*, r_H, o_H^*)]$, where $(s_i, r_i, o_i) \rightarrow (s_i, r_i, o_i^*) \in \mathcal{E}$ is the applied edit, and the terminal entity o_H^* is the updated answer under the edited world.

Given a frozen LLM $\mathcal{M}$ and the edit set $\mathcal{E}$, knowledge editing aims to derive a conditionally edited model $\mathcal{M}^*$. Formally, the objective is to ensure $\mathcal{M}^*$ generates the target answer o_H^* by following the valid reasoning path consistent with C^*.

3.2 Overview of CARE

We propose CARE, a critic-guided closed-loop framework for multi-hop knowledge editing. Unlike prior feed-forward pipelines that execute hop-wise retrieval and reasoning without global verification, CARE performs chain-level verification and triggers bounded backtracking for iterative self-correction, preventing early-hop *silent failures* from cascading to the final answer under the post-edit world.

The overall framework of the proposed method is illustrated in Figure 2, which contains two synergistic stages. First, Post-Edit Graph Construction: we process fact statements and edit targets with EATR. By enforcing strict endpoint alignment and relation consistency, EATR denoises the extracted triples during store construction, which in turn helps the SELECTOR build a higher-fidelity, lower-noise candidate pool for downstream reselection. Second, Critic-Guided Multi-Agent Reasoning: inference starts with a PLANNER that decomposes the multi-hop query Q into a sequence of sub-questions (e.g., $[q_1, q_2, q_3]$). The system then enters an iterative loop where a SELECTOR retrieves and ranks candidate facts, while a CRITIC verifies global coherence and step-wise correctness of the assembled chain. When an Incorrect Chain is detected (e.g., selecting "Greek" in the figure), the CRITIC triggers bounded backtracking (red arrow) to revise the incorrect hop. This closed-loop correction steers the reasoning toward a Corrected Chain (e.g., "Italian"), ensuring the final an-

swer is derived from the edited knowledge.

3.3 Post-Edit Graph Construction with EATR

We build a post-edit graph store for downstream multi-hop reasoning. Given edits $\mathcal{E}$ and their text descriptions, we construct a graph $G_{\mathcal{E}}$ whose triples are consistent with the edited facts.

Edit descriptions may yield noisy or drifted triples under automatic extraction, which can later cascade in multi-hop reasoning; we therefore propose EATR to denoise triples during post-edit store construction. For each edit $e = (s, r, o \rightarrow o^*) \in \mathcal{E}$, we use the benchmark-provided anchor triple (s, r, o^*) only for store construction, while extracting diverse candidates from the edit description. Following prior graph-based editing pipelines (e.g., KEDKG), we apply REBEL-LARGE (a BART-based seq2seq relation extraction model) to obtain a candidate set $\mathcal{T}(e)$, and pair each triple with its evidence sentence to populate the SELECTOR's search index. We refine $\mathcal{T}(e)$ with respect to the anchor triple (s, r, o^*) using two constraints. First, we enforce endpoint alignment: after canonicalization, the head and tail entities must exactly match s and o^*. Second, we enforce relation consistency. Let r_{cand} denote the candidate relation and r the anchor relation. We encode both relation texts using the BGE-M3 sentence embedding model (`bge-m3`) to obtain $\mathbf{e}(\cdot)$, and keep a candidate only if

$$\cos(\mathbf{e}(r_{\text{cand}}), \mathbf{e}(r)) \geq \delta_{\text{rel}}. \quad (1)$$

Because candidates come from edit descriptions, we still retain diverse evidence phrasings for robust retrieval. This refinement reduces extraction noise and drift, keeping $G_{\mathcal{E}}$ faithful to the edit specification and enabling the SELECTOR to retrieve cleaner candidates during closed-loop reasoning.

3.4 Multi-Agent Collaborative Reasoning

Building upon the structurally refined graph $G_{\mathcal{E}}$, CARE performs multi-agent collaborative reasoning: (i) a PLANNER that decomposes the query Q into sequential sub-questions; (ii) a SELECTOR that retrieves and ranks candidate facts from $G_{\mathcal{E}}$ for each hop; and (iii) a CRITIC that verifies the assembled chain and triggers bounded backtracking when an intermediate hop violates Coherence. This division of labor mitigates early-hop *silent failures* via evidence-grounded hop selection and chain-level verification, preventing errors from cascading across hops. All prompts used for the PLANNER, SELECTOR, and CRITIC are provided in Appendix J.

3.4.1 Planner: Iterative Sub-question Decomposition

Given a multi-hop query Q, the PLANNER decomposes it into an ordered list of hop-wise sub-questions $\{q_i\}_{i=1}^{H}$, where each q_i targets an atomic fact lookup needed to answer Q. To make intermediate states explicit, we use a placeholder token `[ENT]` to represent the bridging entity between hops. For $i > 1$, q_i is instantiated from the previous hop output a_{i-1} by setting $[\text{ENT}] \leftarrow a_{i-1}$. To empower the PLANNER to generate such structured decompositions with correct placeholders, we fine-tune it on a dataset of query–plan pairs constructed from gold sub-question sequences with the `[ENT]` placeholder (construction details in Appendix D and training hyperparameters in Appendix G). Let $O = [q_1; \ldots; q_H]$ denote the linearized sub-question sequence (delimited by ";"). We optimize the PLANNER with the standard autoregressive objective:

$$\mathcal{L}_{\text{plan}}(\theta) = -\sum_{t=1}^{|O|} \log p_\theta\big(O_t \mid O_{<t}, Q\big), \quad (2)$$

where θ denotes the PLANNER parameters and O_t is the t-th token in O.

This placeholder mechanism makes entity transitions explicit, enabling hop-wise retrieval on $G_{\mathcal{E}}$ and supporting chain-level verification by checking entity continuity across hops. The detailed prompt for the PLANNER is provided in Figure 6.

3.4.2 Selector: Retrieval and Semantic Ranking

For each sub-question q_i, the SELECTOR follows a retrieve–then–rank pipeline: it first retrieves the top-K candidates from the 1-hop neighborhood of the current subject entity s_i in $G_{\mathcal{E}}$ by matching the sub-question against evidence sentences, and then returns a filtered, ranked pool $\mathcal{P}_i = \{p_{i,1}, \ldots, p_{i,K'}\}$ for critic-guided backtracking, where $K' \leq K$ denotes the number of candidates that survive filtering.

Evidence-grounded filtering and scoring. Coarse retrieval may still include off-target triples that can induce early-hop *silent failures*. The

SELECTOR therefore acts as a context-grounded verifier, scoring candidates using only the triple and its evidence sentence (overriding parametric memory); $\mathcal{P}_i$ is curated to be high-fidelity and low-noise so that backtracking can re-select a correct supporting fact rather than propagating errors downstream.

The SELECTOR operates under a strict evaluation protocol comprising four behavioral constraints: (1) **Entity alignment** (head matches s_i); (2) **Evidence entailment** (evidence directly answers q_i); (3) **Relation match** (relation fits the sub-question intent); and (4) **Structural validity** (reject self-loops where head = tail). The hop output is the tail of the top-ranked surviving candidate; if $\mathcal{P}_i$ is empty, we fall back to direct generation with the backbone LLM. The detailed prompt is provided in Figure 7.

3.4.3 Critic: Chain Verification with Bounded Backtracking

Given a preliminary chain $\mathcal{R} = [c_1^*, \ldots, c_H^*]$, the CRITIC acts as a chain verifier: it inspects both step-wise correctness and global coherence, pinpoints where the chain breaks, and triggers bounded backtracking to revise the failure hop rather than letting errors cascade to the final answer.

Verification protocol. Conditioned on the sub-question chain $\{q_i\}_{i=1}^{H}$, the selected hop-level facts $\mathcal{R}$, and the final predicted answer, the CRITIC follows a strict protocol: (1) **Local relevance check:** each c_i^* must directly answer q_i based on its evidence, without relying on parametric knowledge; (2) **Bridge consistency check:** the output entity of hop i must exactly match the input entity of hop $i{+}1$ ($o_i = s_{i+1}$); (3) **Factual grounding check:** Each hop's triple must be entailed by its associated evidence sentence. Fallback-generated hops must also be evidence-entailed, and the model must not use parametric knowledge; and (4) **Final deduction check:** the final answer must follow logically from the verified chain.

Bounded backtracking. If the CRITIC flags a failure at hop $\hat{i}$, it rejects the chain and backtracks to $\hat{i}$ with a retry budget $\tau_{\max} = 2$. Within this budget, CARE repairs the chain via two strategies: (1) Reselection. The CRITIC asks the SELECTOR to discard the current choice and try the next-best candidate in $\mathcal{P}_{\hat{i}}$, pruning the invalid branch before reassembling the remaining hops. (2) Fallback generation. If $\mathcal{P}_{\hat{i}}$ is exhausted, we switch hop $\hat{i}$ to direct generation with the backbone LLM grounded in local evidence to proceed. The detailed prompt for the CRITIC is provided in Figure 8.

4 Experiments

4.1 Experiment Setup

Datasets. To evaluate our framework, we utilize two established benchmarks derived from the original MQuAKE dataset: MQuAKE-2002 and MQuAKE-hard (Wang et al., 2024c; Zhong et al., 2023). The MQuAKE-2002 dataset is a curated version that removes instances where the ground-truth answers are invalidated by conflicting knowledge updates found in other samples. In contrast, MQuAKE-hard represents a more rigorous test set, specifically composed of instances that require reasoning over the maximum number of edited facts per query. Further specifications regarding these datasets can be found in Appendix A.

Evaluation Metrics. We follow prior work (Zhong et al., 2023; Gu et al., 2024; Wang et al., 2024c) and report multi-hop accuracy (denoted as Acc), i.e., whether the final answer matches the gold answer, and hop-wise accuracy (denoted as Hop-Acc), i.e., whether each hop selects the correct intermediate entity/fact along the gold chain. We evaluate under different edit scopes (1-edited, 100-edited, and All-edited), where Hop-Acc provides a finer-grained view of reasoning robustness beyond Acc.

Baselines. We compare CARE with representative multi-hop knowledge editing baselines: FT, ROME (Meng et al., 2022), MEMIT (Meng et al., 2023), MeLLo (Zhong et al., 2023), DeepEdit (Wang et al., 2024c), PokeMQA (Gu et al., 2024), IRAKE (Liu et al., 2025), and the graph-based method KEDKG (Lu et al., 2025). We omit "Reason-KE" (Wu et al., 2025) since it requires training the backbone, unlike our parameter-preserving, plug-and-play setting. We use LLaMA-3-8B and GPT-4o-mini as the backbone LLMs in our main experiments. Detailed descriptions of all baselines are provided in Appendix B.

Experimental Setup. We use a fine-tuned LLaMA-2-7B as the PLANNER and DeepSeek-V3 (Liu et al., 2024) as the SELECTOR and

Methods	MQuAKE-2002						MQuAKE-hard			
	1-edited		100-edited		All-edited		1-edited		All-edited	
	Acc	Hop-Acc	Acc	Hop-Acc	Acc	Hop-Acc	Acc	Hop-Acc	Acc	Hop-Acc
LLaMA-3-8B										
FT	23.30	-	1.97	-	0.80	-	3.10	-	1.16	-
FT_{CoT}	27.17	6.10	2.41	0.04	0.99	0.04	3.80	0.00	1.39	0.00
ROME	12.37	-	2.47	-	2.37	-	3.20	-	1.63	-
$ROME_{CoT}$	15.27	6.47	4.60	0.03	4.53	0.20	4.20	0.00	2.09	0.00
MEMIT	13.23	-	8.20	-	4.27	-	4.70	-	2.09	-
$MEMIT_{CoT}$	17.97	7.23	11.40	3.47	6.30	0.70	5.10	0.00	2.33	0.00
MeLLo	36.57	11.30	21.30	12.07	14.33	7.30	10.50	3.50	4.60	0.20
DeepEdit	40.22	12.43	32.20	16.20	17.76	9.50	14.50	2.40	8.80	0.30
PokeMQA	48.23	33.60	39.13	29.88	36.81	24.97	33.96	19.51	27.16	17.14
IRAKE	65.30	46.30	58.50	48.50	55.24	44.80	52.50	33.00	40.79	35.90
KEDKG	56.84	54.49	56.04	52.74	53.09	48.90	65.50	62.70	68.53	63.63
CARE (OURS)	**74.32**	**72.77**	**74.45**	**72.80**	**74.53**	**72.33**	**94.17**	**93.94**	**93.01**	**93.01**
GPT-4o-Mini										
MeLLo	33.20	8.20	22.90	11.50	17.20	5.40	8.40	1.20	4.70	0.40
DeepEdit	41.60	13.20	34.60	17.50	20.50	10.50	15.30	1.80	7.20	0.60
PokeMQA	51.23	34.70	38.33	27.36	34.73	23.77	34.10	21.50	26.15	15.03
IRAKE	56.26	42.51	51.50	46.58	46.75	42.11	41.35	24.81	32.13	24.71
KEDKG	66.88	65.38	62.58	60.93	65.03	62.88	66.43	63.40	62.93	58.74
CARE (OURS)	**87.11**	**86.76**	**85.95**	**85.60**	**86.06**	**85.66**	**94.17**	**95.10**	**93.01**	**92.07**

Table 1: Main results. The best scores for each base LLM are marked in bold. "-" denotes "not applicable".

CRITIC. We evaluate on MQuAKE-2002 and MQuAKE-hard under 1-edited, 100-edited, and All-edited settings. We build the post-edit graph with REBEL-LARGE and apply EATR with δ_{rel}=0.80. We use BM25 for candidate retrieval with K=3 in all experiments, as validated by the retrieval ablations in Appendix H. Additional experimental details, efficiency measurements, and hyperparameter sensitivity analyses are provided in Appendix C, Appendix E, and Appendix F.

4.2 Main Results

Table 1 shows the experimental results on the MQuAKE-2002 and MQuAKE-hard benchmarks. We draw the following observations:

Overall performance. Our method achieves the strongest performance on both datasets across all edit scopes and metrics. On MQuAKE-hard under the All-edited setting, it surpasses 93% in both Acc and Hop-Acc, improving over KEDKG by 24.48% and 29.38%, respectively, indicating effective mitigation of multi-hop error propagation.

Faithful intermediate reasoning. Many baselines exhibit a noticeable discrepancy between Acc and Hop-Acc, whereas our method keeps the two metrics closely aligned across settings, indicating more faithful intermediate reasoning chains. We further analyze this discrepancy in Section 5.2 (Figure 4).

Robustness to edit scaling. Under larger edit scopes (1-edited and All-edited), all baselines degrade noticeably, while our method remains substantially more stable across both datasets, demonstrating strong robustness to edit scaling and memory growth.

Backbone generalization. The choice of base LLM affects editing performance, with several baselines varying substantially across different backbones. In contrast, our method performs strongly across different backbones, suggesting good plug-and-play generalizability.

4.3 Ablation Study

We conduct ablation experiments with LLaMA-3-8B on MQuAKE-2002 and MQuAKE-hard by dis-

EATR	Selector	Critic	MQuAKE-2002						MQuAKE-hard			
			1-edited		100-edited		All-edited		1-edited		All-edited	
			Acc	Hop-Acc	Acc	Hop-Acc	Acc	Hop-Acc	Acc	Hop-Acc	Acc	Hop-Acc
×	✓	✓	56.99	54.34	57.90	55.45	55.69	52.39	63.40	61.30	67.59	63.63
✓	×	✓	62.73	60.03	55.45	53.15	54.94	50.84	69.69	68.29	58.27	54.77
✓	✓	×	68.18	64.18	66.15	63.25	65.98	63.08	91.37	89.44	90.20	87.34
✓	✓	✓	**74.32**	**72.77**	**74.45**	**72.80**	**74.53**	**72.33**	**94.17**	**93.94**	**93.01**	**93.01**

Table 2: Results of ablation study. "✓" and "×" denote the enabled and disabled modules, respectively.

abling one component at a time: *w/o EATR* builds the post-edit graph from raw extracted triples; *w/o Selector* removes evidence-grounded semantic verification and relies on coarse retrieval; and *w/o Critic* runs a feed-forward pipeline without chain-level verification/backtracking. Results are shown in Table 2.

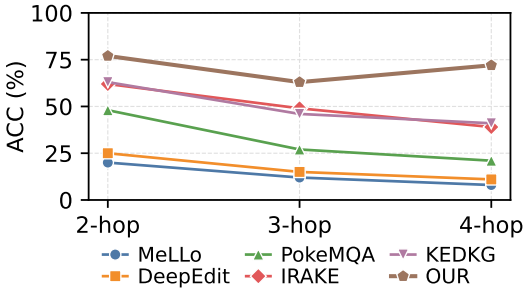
EATR. Disabling EATR notably hurts performance on MQuAKE-2002 under the 1-edited setting, where Acc drops by 17.33%, indicating that edit-anchored refinement is essential for constructing a faithful post-edit graph and safeguarding the SELECTOR's high-quality candidate pool.

Selector. Removing the SELECTOR drops MQuAKE-hard (All-edited) Acc to 58.27%, making the candidate pool more susceptible to noisy hop selection, which exacerbates downstream error propagation and reduces the CRITIC's room for backtracking-based revision.

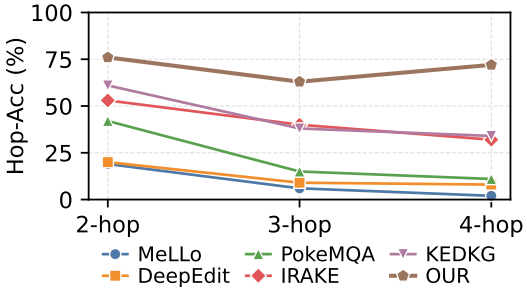
Critic. The CRITIC performs chain-level verification and triggers bounded backtracking to reject inconsistent chains and prevent *silent failures* from propagating. However, since it cannot generate new facts, its corrective gain depends on whether reliable alternatives exist in the candidate pool.

5 Analysis

We analyze the overall behavior of CARE along four dimensions. (1) robustness under increasing reasoning depth; (2) faithfulness of intermediate reasoning chains, reflected by the Acc–Hop-Acc gap; (3) CRITIC behavior, including chain-level error detection precision and correction efficacy with bounded backtracking; and (4) generalizability across diverse backbone LLMs. A qualitative case study illustrating the bounded backtracking behavior is provided in Appendix I. Unless otherwise specified, all analyses are conducted with LLaMA-3-8B as the backbone.



(a) multi-hop accuracy



(b) hop-wise accuracy

Figure 3: Acc and Hop-Acc on MQuAKE-2002 (All-edited) grouped by hop count across different KE methods.

5.1 Robustness to Reasoning Depth

Figure 3 reports Acc and Hop-Acc on MQuAKE-2002 (All-edited) grouped by hop count. We observe that most baselines degrade as hop count increases, whereas our method remains consistently stronger and degrades more slowly under deeper reasoning. This stability comes from our closed-loop design: the SELECTOR curates a high-quality candidate pool via evidence-grounded verification to reduce early-hop errors, and the CRITIC performs chain-level verification with bounded backtracking to prevent error propagation. Consequently, longer chains are less brittle and maintain higher Acc and Hop-Acc.

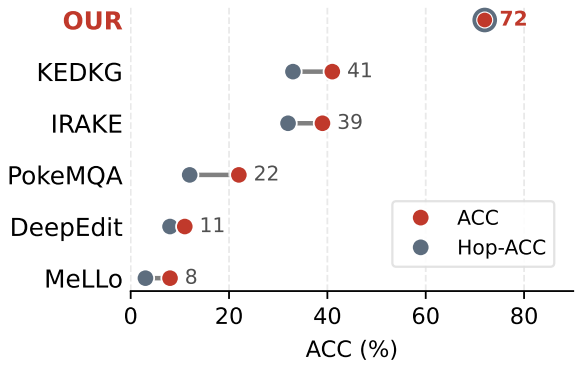


Figure 4: Acc (red) vs. Hop-Acc (gray) on 4-hop queries in MQuAKE-2002 (All-edited). A shorter connecting line indicates more faithful multi-hop reasoning.

Hop Count	Error Detection (Precision)	Effective Correction (Wrong → Right)
2-hop	86.9%	63.5%
3-hop	83.0%	68.8%
4-hop	78.0%	58.3%
Overall	**82.9%**	**63.5%**

Table 3: CRITIC effectiveness on MQuAKE-2002 (All-edited, by hop count). Error Detection (Precision) measures how often a chain flagged by the CRITIC is indeed wrong; Effective Correction reports the success rate of fixing detected failures within bounded backtracking.

5.2 Faithfulness of Reasoning Chains

As noted in the main results, many baselines show a noticeable gap between Acc and Hop-Acc, while our method keeps them closely aligned. We focus on MQuAKE-2002 (All-edited) and restrict to 4-hop queries, visualizing the dumbbell plot in Figure 4, where a longer line indicates stronger hop-level inconsistency and potential error propagation. Figure 4 shows that several baselines reach the correct final answer despite unreliable intermediate hops (large Acc–Hop-Acc gaps). In contrast, our method yields consistently smaller gaps, suggesting that its gains come from more coherent chains supported by evidence-grounded hop selection and closed-loop verification, rather than brittle intermediate hops.

5.3 Mechanism Analysis of the Critic

To validate our critic-guided chain-level verification with bounded backtracking, we analyze two capabilities of the CRITIC: (1) how accurately it identifies erroneous reasoning chains, and (2) how effectively it repairs them through closed-loop revision. Table 3 reports results on MQuAKE-2002 by hop count. Error Detection (Precision) measures how often a flagged chain is indeed incorrect, and Effective Correction measures the success rate of fixing detected failures within the retry budget. The CRITIC achieves consistently high detection precision (82.9% overall), indicating that it reliably flags genuinely problematic chains rather than disrupting correct ones. Although precision decreases slightly as hop count increases (86.9% → 78.0%), it remains strong for 4-hop queries where error propagation is more likely. Moreover, the CRITIC corrects 63.5% of detected failures overall, showing that bounded backtracking can recover from intermediate-hop errors and mitigate downstream error propagation when reliable alternatives exist.

Backbone	Method	MQuAKE-2002		MQuAKE-hard	
		Acc	Hop-Acc	Acc	Hop-Acc
Open-Source Models					
LLaMA-2-7B	KEDKG	63.04	61.80	68.53	63.63
	OURS	**86.31**	**85.11**	**94.63**	**93.93**
Qwen2.5-7B	KEDKG	60.19	57.69	67.60	63.63
	OURS	**82.36**	**81.26**	**94.63**	**94.17**
DeepSeek-V3.2	KEDKG	64.88	62.55	66.90	62.24
	OURS	**87.01**	**86.56**	**94.17**	**93.70**
Closed-Source Models					
GPT-4o	KEDKG	61.58	58.84	67.60	62.24
	OURS	**83.97**	**83.41**	**94.63**	**94.41**

Table 4: Performance comparison between our method (CARE) and the strongest graph-based baseline (KEDKG) across different backbone models, reporting All-edited results on MQuAKE-2002 and MQuAKE-hard.

5.4 Generalizability Across Backbones

Finally, we investigate the generalizability of CARE across different foundational models under the All-edited setting. As shown in Table 4, we compare our method against the strongest knowledge graph-based baseline, KEDKG, using LLaMA-2-7B, Qwen2.5-7B, DeepSeek-V3.2 and GPT-4o. The results demonstrate that CARE consistently outperforms KEDKG regardless of the backbone, exhibiting stable gains across diverse model families and scales. Crucially, our framework functions as a plug-and-play solution that can be directly applied to mainstream LLMs without parameter modifications. These improvements persist when scaling to larger backbones such as GPT-4o and DeepSeek-V3.2, confirming that

CARE is effective beyond smaller open-source models and can robustly benefit large-scale systems as well.

6 Conclusion

We present CARE, a critic-guided closed-loop framework for multi-hop knowledge editing over graph-structured edit stores. By combining a CRITIC for chain-level verification with bounded backtracking and a SELECTOR for evidence-grounded semantic verification, CARE mitigates early-hop errors and downstream error propagation. We also propose EATR to construct a faithful post-edit graph and reduce extraction noise, improving the quality of candidates for revision. Experiments on MQuAKE-2002 and MQuAKE-hard demonstrate state-of-the-art performance, smaller Acc–Hop-Acc gaps, and stronger robustness under deeper reasoning.

Limitations

Our work has the following limitations. First, CARE adopts a closed-loop multi-agent reasoning design (CRITIC/SELECTOR) with bounded backtracking, which introduces additional inference overhead and token/resource consumption compared to feed-forward pipelines. However, our experiments show that this extra cost yields substantially better multi-hop editing accuracy and more stable intermediate reasoning, making the trade-off worthwhile in practice. Second, although CARE performs strongly on multi-hop knowledge editing benchmarks, its effectiveness in other high-stakes domains (e.g., finance or law) remains unexplored. Future work will focus on scaling CARE to larger and noisier edit stores and extending it to broader settings beyond the knowledge-triple formulation.

Ethics Statement

In this work, we utilize publicly available datasets and do not collect any personally identifiable information. All datasets and models are employed in strict compliance with their intended purposes and respective licenses. The primary goal of this research is to improve the reliability and coherence of multi-hop knowledge editing in Large Language Models; we explicitly condemn any malicious manipulation of facts or potential misuse.

Acknowledgments

This work was supported by the National Key Research and Development Program of China (Grant No. 2024YFE0210800) and the National Natural Science Foundation of China (Grant No. 62476025). This work was also supported by the Fundamental Research Funds for the Central Universities.

References

Yash Kumar Atri, Ahmed Alaa, and Thomas Hartvigsen. 2025. Lifelong model editing with graph-based external memory. In *Findings of the Association for Computational Linguistics: ACL 2025*, pages 13336–13352.

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

Fengxiang Cheng, Haoxuan Li, Fenrong Liu, Robert van Rooij, Kun Zhang, and Zhouchen Lin. 2025. Empowering llms with logical reasoning: A comprehensive survey. *arXiv preprint arXiv:2502.15652*.

Nadir Durrani, Basel Mousi, and Fahim Dalvi. 2025. Editing across languages: A survey of multilingual knowledge editing. *CoRR*, abs/2505.14393.

Junfeng Fang, Houcheng Jiang, Kun Wang, Yunshan Ma, Jie Shi, Xiang Wang, Xiangnan He, and Tat-Seng Chua. 2025. Alphaedit: Null-space constrained knowledge editing for language models. In *The Thirteenth International Conference on Learning Representations, ICLR 2025, Singapore, April 24-28, 2025*. OpenReview.net.

Jinhu Fu, Kun Wang, Chongye Guo, Junfeng Fang, Wentao Zhang, and Sen Su. 2025. Knowledge graph-driven memory editing with directional interventions. In *Findings of the Association for Computational Linguistics: EMNLP 2025*, pages 4860–4874.

T Gao, X Yao, and Danqi Chen. 2021. Simcse: Simple contrastive learning of sentence embeddings. In *EMNLP 2021-2021 Conference on Empirical Methods in Natural Language Processing, Proceedings*.

Aaron Grattafiori, Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Alex Vaughan, and 1 others. 2024. The llama 3 herd of models. *arXiv preprint arXiv:2407.21783*.

Hengrui Gu, Kaixiong Zhou, Xiaotian Han, Ninghao Liu, Ruobing Wang, and Xin Wang. 2024.

Pokemqa: Programmable knowledge editing for multi-hop question answering. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 8069–8083.

Tao He, Hao Li, Jingchang Chen, Runxuan Liu, Yixin Cao, Lizi Liao, Zihao Zheng, Zheng Chu, Jiafeng Liang, Ming Liu, and 1 others. 2025. Breaking the reasoning barrier a survey on llm complex reasoning through the lens of self-evolution. In *Findings of the Association for Computational Linguistics: ACL 2025*, pages 7377–7417.

Binyuan Hui, Jian Yang, Zeyu Cui, Jiaxi Yang, Dayiheng Liu, Lei Zhang, Tianyu Liu, Jiajun Zhang, Bowen Yu, Keming Lu, and 1 others. 2024. Qwen2. 5-coder technical report. *arXiv preprint arXiv:2409.12186*.

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.

Xiaopeng Li, Shanwen Wang, Shasha Li, Shezheng Song, Bin Ji, Jun Ma, and Jie Yu. 2025. Rethinking the residual distribution of locate-then-editing methods in model editing. *CoRR*, abs/2502.03748.

Aixin Liu, Bei Feng, Bing Xue, Bingxuan Wang, Bochao Wu, Chengda Lu, Chenggang Zhao, Chengqi Deng, Chenyu Zhang, Chong Ruan, and 1 others. 2024. Deepseek-v3 technical report. *arXiv preprint arXiv:2412.19437*.

Yi Liu, Xiangrong Zhu, Xiangyu Liu, Wei Wei, and Wei Hu. 2025. Avoiding knowledge edit skipping in multi-hop question answering with guided decomposition. *arXiv preprint arXiv:2509.07555*.

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

Kevin Meng, David Bau, Alex Andonian, and Yonatan Belinkov. 2022. Locating and editing factual associations in gpt. *Advances in neural information processing systems*, 35:17359–17372.

Kevin Meng, Arnab Sen Sharma, Alex J Andonian, Yonatan Belinkov, and David Bau. 2023. Mass-editing memory in a transformer. In *ICLR*.

Eric Mitchell, Charles Lin, Antoine Bosselut, Chelsea Finn, and Christopher D Manning. 2022a. Fast model editing at scale. In *International Conference on Learning Representations*.

Eric Mitchell, Charles Lin, Antoine Bosselut, Christopher D Manning, and Chelsea Finn. 2022b. Memory-based model editing at scale. In *International Conference on Machine Learning*, pages 15817–15831. PMLR.

Stephen Robertson, Hugo Zaragoza, and 1 others. 2009. The probabilistic relevance framework: Bm25 and beyond. *Foundations and Trends® in Information Retrieval*, 3(4):333–389.

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.

Peng Wang, Zexi Li, Ningyu Zhang, Ziwen Xu, Yunzhi Yao, Yong Jiang, Pengjun Xie, Fei Huang, and Huajun Chen. 2024a. WISE: rethinking the knowledge memory for lifelong model editing of large language models. In *Advances in Neural Information Processing Systems 38: Annual Conference on Neural Information Processing Systems 2024, NeurIPS 2024, Vancouver, BC, Canada, December 10 - 15, 2024*.

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

Yiwei Wang, Muhao Chen, Nanyun Peng, and Kai-Wei Chang. 2024c. Deepedit: Knowledge editing as decoding with constraints. *CoRR*, abs/2401.10471.

Yuchen Wu, Liang Ding, Li Shen, and Dacheng Tao. 2025. Robust knowledge editing via explicit reasoning chains for distractor-resilient multi-hop qa. *arXiv preprint arXiv:2509.01468*.

Mengqi Zhang, Xiaotian Ye, Qiang Liu, Pengjie Ren, Shu Wu, and Zhumin Chen. 2024a. Knowledge graph enhanced large language model editing. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pages 22647–22662.

Ningyu Zhang, Yunzhi Yao, Bozhong Tian, Peng Wang, Shumin Deng, Mengru Wang, Zekun Xi, Shengyu Mao, Jintian Zhang, Yuansheng Ni, and 1 others. 2024b. A comprehensive study of knowledge editing for large language models. *arXiv preprint arXiv:2401.01286*.

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

Zhiyuan Zhu, Yusheng Liao, Zhe Chen, Yuhao Wang, Yunfeng Guan, Yanfeng Wang, and Yu Wang. 2025.

Evolvebench: A comprehensive benchmark for assessing temporal awareness in llms on evolving knowledge. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 16173–16188.

A Dataset Statistics

We evaluate CARE on two multi-hop knowledge editing benchmarks derived from MQuAKE: MQuAKE-2002 and MQuAKE-hard (Wang et al., 2024c; Zhong et al., 2023). We do not use MQuAKE-3k for edit-store scaling evaluation because cross-case conflicting edits can invalidate gold answers when many edits are stored together; MQuAKE-2002 removes such conflicts, and MQuAKE-hard keeps the most challenging cases (all 4-hop with 4 edits). Each instance contains edits (`requested_rewrite`) and three paraphrased multi-hop questions (`questions`); a case is correct if the model answers any question correctly under the post-edit world. We report 1-edited, 100-edited, and All-edited settings by varying the size of the external edit store to test robustness as memory size increases. Table 5 summarizes dataset statistics, and Table 6 reports hop distributions.

Dataset	#Inst.	Avg. hops	Avg. edits	#Conflicts
MQuAKE-3k	3,000	3.0	2.0	998
MQuAKE-2002	2,002	2.7	2.2	0
MQuAKE-hard	429	4.0	4.0	0

Table 5: Dataset summary. #Conflicts counts instances whose gold answers are invalidated by cross-case conflicting edits.

#Edits/case	2-hop	3-hop	4-hop	Total
MQuAKE-2002				
1	479	71	7	557
2	487	244	20	751
3	0	310	116	426
4	0	0	268	268
All	966	625	411	2,002
MQuAKE-hard				
4	0	0	429	429

Table 6: Hop distribution conditioned on the number of edited facts per case.

B Baselines

We compare CARE with representative baselines spanning three paradigms of knowledge editing: parameter-editing, parameter-preserving retrieval and reasoning, and graph-based editing with structured stores.

Parameter-editing baselines. These methods inject new knowledge by directly updating the backbone LLM parameters (Fang et al., 2025; Li et al., 2025; Wang et al., 2024a). We include FT (fine-tuning on edit targets) and two representative locate-and-edit approaches, ROME and MEMIT (Meng et al., 2022, 2023). In addition, we report a CoT variant for each method (e.g., FT_{CoT}, ROME_{CoT}, $\text{MEMIT}_{\text{CoT}}$), where the model is prompted to produce a step-by-step multi-hop reasoning chain before the final answer, enabling a more direct comparison with retrieval- and reasoning-based approaches.

Parameter-preserving retrieval and reasoning baselines. These methods keep the backbone frozen and condition generation on retrieved evidence from an external edit store at inference time (Mitchell et al., 2022b). We include MeLLo (Zhong et al., 2023) and DeepEdit (Wang et al., 2024c), which decompose multi-hop queries into sub-questions and iteratively retrieve supporting evidence for compositional reasoning; PokeMQA (Gu et al., 2024), which leverages retrieved edited facts and structured prompting to generate step-by-step answers; and IRAKE (Liu et al., 2025), which targets the edit skipping issue by retrieving candidate edits and prior decomposition records before generating sub-questions, making the decomposition better aligned with the available edits in the store.

Graph-based editing and reasoning baselines. These approaches explicitly organize edits into graph-structured stores and perform neighborhood/path retrieval to support multi-hop inference (Chen et al., 2024; Lu et al., 2025; Zhang et al., 2024a). We include KEDKG (Lu et al., 2025), a strong graph baseline that constructs an edited knowledge graph and performs constrained graph traversal for multi-hop reasoning. Compared to text-based memories, graph representations are typically more controllable and compositional under edit scaling and memory growth.

C Experimental Setup

We evaluate CARE on MQuAKE-2002 and MQuAKE-hard across 1-edited, 100-edited, and

All-edited settings by varying the size of the external edit store. To test robustness across model families, we conduct experiments using multiple backbone LLMs, including LLaMA-3-8B (Grattafiori et al., 2024) and GPT-4o-mini in our main results (Table 1), and further include LLaMA-2-7B, Qwen2.5-7B-Instruct (Hui et al., 2024), DeepSeek-V3.2 (Liu et al., 2024), and GPT-4o (Table 4).

CARE is instantiated as a multi-agent workflow. The Planner uses a fine-tuned LLaMA-2-7B to decompose multi-hop questions with the `[ENT]` placeholder. The Selector and Critic are implemented with DeepSeek-V3 to perform evidence-grounded triple scoring and chain-level verification/backtracking, respectively. To construct the post-edit graph $G_{\mathcal{E}}$, we extract candidate triples and their evidence sentences from edit descriptions using REBEL-LARGE. We then perform entity linking by mapping entity mentions to Wikidata QIDs using spaCy (`en_core_web_md`) augmented with an `entityLinker` pipeline component; when linking fails, we fall back to surface-form entity strings. We then apply edit-anchored refinement with strict endpoint alignment and relation consistency filtering, retaining candidates only if $\cos(\mathbf{e}(r_c), \mathbf{e}(r)) \geq \delta_{\text{rel}}$, where $\delta_{\text{rel}} = 0.80$ is selected on the development set. We compute relation embeddings with the BGE-M3 sentence embedding model (`bge-m3`) for the relation consistency filtering in EATR. At inference time, for each hop, we enumerate the 1-hop neighborhood of the current entity in $G_{\mathcal{E}}$ and retrieve candidates via BM25 matching between the sub-question and evidence sentences, retaining the top-K candidates (with $K = 3$) for the Selector to rank under an evidence-only protocol. The Critic executes global chain-consistency checks and triggers bounded backtracking with a retry budget $\tau_{\max} = 2$. All agent calls use deterministic decoding (temperature $= 0$) to minimize variance. All experiments are conducted on a server equipped with 8× NVIDIA RTX A6000 GPUs.

D Training Dataset Construction

We adopt the training protocol from KEDKG to train the PLANNER component for multi-hop question decomposition. We use MQuAKE-CF as the source pool and perform strict test-set decontamination to avoid overlap with our evaluation benchmarks (MQuAKE-2002 and MQuAKE-hard). Specifically, we filter out any training case that shares the same subject–relation pair (s, r) with any case in the test sets, ensuring that the PLANNER does not observe the same edited facts during training. Each MQuAKE-CF case provides three paraphrased multi-hop questions Q and their corresponding sub-question sequences $\{q_i\}_{i=1}^{H}$. To train a decomposition model that is robust to entity transitions, we keep the first sub-question q_1 unchanged and replace the subject entity in the remaining sub-questions with a special placeholder token (e.g., `[ENT]`) to model entity transitions. We then linearize the sub-questions with ; as the delimiter and construct the PLANNER training set:

$$\mathcal{D}_{\text{plan}} = \{(Q,\ [q_1; \ldots; q_H])\}. \tag{3}$$

Hyper-parameters	PLANNER (LLM)
Epochs	3
Seq. length	2048
Learning rate	1e-5
Batch size	1
Optimizer	AdamW
Weight decay	0.01

Table 7: Training hyper-parameters for the Planner used for question decomposition.

Retriever	K	Recall@K (%)	Latency (ms)	Acc (%)
BM25	3	90.4	0.14	93.0
	5	90.9	0.15	92.5
	7	89.9	0.15	90.9
SimCSE	3	90.8	22.9	92.0
	5	90.7	23.5	92.3
	7	90.2	24.0	92.0
Contriever	3	89.9	23.6	91.8
	5	89.5	23.1	92.0
	7	90.2	24.3	91.3

Table 8: Retriever and top-K ablation study for the Selector's coarse retrieval stage on MQuAKE-hard (All-edited). Latency (ms) is the average retrieval latency per query for returning the top-K candidates (excluding downstream ranking/verification). Metrics are reported with one decimal place.

E Efficiency and Cost Analysis

Since CARE introduces closed-loop multi-agent reasoning and bounded backtracking, it is important to quantify the associated computational overhead. To this end, we report both module-level cost breakdowns and end-to-end efficiency

Module	Latency (ms)	Input Tokens	Output Tokens	Tokens
EATR	1305.4	0.0	0.0	0.0
Planner	11025.6	426.3	454.6	1090.9
Selector	10673.9	3299.5	41.8	3341.3
Critic	5094.4	2120.5	93.9	2214.4
Backbone	1298.7	279.4	52.4	331.8

Table 9: Module-level cost breakdown of CARE on MQuAKE-hard with $\tau_{\max} = 2$. All values are averaged per question.

Method	Accuracy	Hop-Acc.	Avg Latency (ms)	Tokens
KEDKG	0.655	0.627	13839.4	1310.4
CARE ($\tau_{\max} = 2$)	0.942	0.939	30029.8	6978.4

Table 10: End-to-end efficiency comparison on MQuAKE-hard. Tokens are averaged per question.

on MQuAKE-hard. Unless otherwise specified, all values in this section are averaged per question, and latency is measured from the start of question processing to the final answer generation. In our setup, the Planner (LLaMA-2-7B) and the backbone (LLaMA-3-8B) are served locally via vLLM on a single RTX A6000, while the Selector and Critic use the official DeepSeek-V3 API. All local components across methods are measured on the same single A6000 GPU, and API latency is included in the reported end-to-end runtime. Table 9 presents the module-level cost breakdown of CARE with $\tau_{\max} = 2$. The additional overhead is mainly introduced by the Planner and Selector, which dominate both latency and token consumption, while the backbone itself accounts for only a relatively small portion of the total cost. This result suggests that the extra computation primarily comes from CARE's verification-and-revision mechanism, rather than from the frozen backbone model. Table 10 further compares CARE with KEDKG in terms of end-to-end efficiency. As expected, CARE is substantially more expensive than the feed-forward baseline in both latency and token usage. However, this overhead is accompanied by large gains in both Accuracy and Hop-Accuracy. We therefore view the added cost as a reasonable trade-off for the substantial improvement in post-edit multi-hop reasoning quality.

F Hyperparameter Sensitivity

We study the sensitivity of CARE to two key hyperparameters on MQuAKE-hard: the bounded backtracking budget $\tau_{\max}$ and the EATR relation consistency threshold δ_{rel}. Table 11 shows that increasing $\tau_{\max}$ from 1 to 2 improves both Accuracy and Hop-Accuracy, indicating that a small amount of bounded revision is beneficial. However, setting $\tau_{\max} = 3$ further increases latency and token usage without improving performance, so we use $\tau_{\max} = 2$ by default. Table 12 shows that smaller δ_{rel} values retain more noisy triples, while overly strict thresholds may remove useful paraphrastic variants. Among the tested values, $\delta_{\text{rel}} = 0.8$ yields the best overall performance and is therefore used in the main experiments.

G Details about Training

We fine-tune the PLANNER on $\mathcal{D}_{\text{plan}}$ using a standard autoregressive objective over the linearized sub-question sequences. To prevent test-set leakage, we perform strict data decontamination by removing any training instances that share the same subject–relation pair (s, r) with the MQuAKE-2002 and MQuAKE-hard test sets. We then split the remaining MQuAKE-CF data into 80% training and 20% validation. Table 7 summarizes the hyper-parameters. Unless otherwise specified, we train for 3 epochs with AdamW, learning rate 1×10^{-5}, weight decay 0.01, batch size 1, and maximum sequence length 2048.

H Ablations on Selector Retrieval

We study the Selector's coarse candidate retrieval stage by varying (i) the retriever (BM25 vs. dense retrievers) and (ii) the top-K size of the retrieved candidate pool. Specifically, BM25 is a sparse lexical retriever over evidence sentences, while SimCSE and Contriever are dense sentence-embedding retrievers that rank candidates by vector similarity (Robertson et al., 2009; Gao et al., 2021; Izacard et al., 2022). All experiments are conducted on MQuAKE-hard under the All-edited setting, which is the most challenging scenario with the largest edit store. Table 8 reports Recall@K, Acc, and retrieval latency. Here, Latency (ms) denotes the average wall-clock time per query for retrieving the top-K candidates from the graph store (excluding downstream ranking/verification). Acc is computed from the final answers produced by the full pipeline given the retrieved candidate pool, rather than measuring the retrieval step alone. Overall, BM25 achieves comparable (or slightly better) Recall@K than dense retrievers while being substantially faster, and performance peaks with a small candidate budget. Based on these results, we use BM25 with K=3 as the de-

Edited Knowledge Facts:
World Indoor Soccer League is associated with baseball; Baseball was created in the United Kingdom; The official language of the United Kingdom is Italian.

KEDKG:

Question:
Which language is the official language of the country that originated the World Indoor Soccer League?
Entity: World Indoor Soccer League

Subquestion1: Which sport is World Indoor Soccer League associated with?
Retrieved: baseball. ✓
Generated answer:
World Indoor Soccer League is associated with baseball.

Subquestion2: Which country was baseball created in?
Retrieved: United Kingdom. ✓
Generated answer:
Baseball was created in the United Kingdom.

Subquestion3: What is the official language of United Kingdom?
Retrieved: Greek ✗
Generated answer:
The official language of the United Kingdom is Greek.

Final answer: Greek. ✗

CARE (Ours):

Question:
Which language is the official language of the country that originated the World Indoor Soccer League?

Subquestion1: Which sport is World Indoor Soccer League associated with?
Selector: baseball (1.0). ✓
Generated answer:
World Indoor Soccer League is associated with baseball.

Subquestion2: Which country was baseball created in?
Selector: United Kingdom(1.0). ✓
Generated answer:
Baseball was created in the United Kingdom.

Subquestion3: What is the official language of United Kingdom?
Selector: Greek(1.0) ✗ Italian(0.9) ✓
Generated answer:
The official language of the United Kingdom is Greek.
Critic: step3 error!!! Revert to Subquestion 3.

Subquestion3: What is the official language of United Kingdom?
Selector: Greek(discard) ✗ Italian(0.9) ✓
Generated answer:
The official language of the United Kingdom is Italian.
Final answer: Italian. ✓

Figure 5: Case study comparing the feed-forward baseline KEDKG (Left) with CARE (Right), where the CRITIC detects the erroneous hop and triggers bounded backtracking for correction.

$\tau_{\max}$	Accuracy	Hop-Acc.	Mean Latency (ms)	P50 Latency (ms)	P90 Latency (ms)	Tokens
1	0.928	0.918	28694.2	25932.9	37643.2	6106.7
2	0.942	0.939	30029.8	25785.4	48488.3	6978.4
3	0.932	0.925	32916.8	25651.3	63859.3	7779.5

Table 11: Sensitivity to the bounded backtracking budget $\tau_{\max}$ on MQuAKE-hard. Tokens are averaged per question.

δ_{rel}	Accuracy	Hop-Acc.
0.5	0.909	0.897
0.6	0.918	0.909
0.7	0.918	0.911
0.8	0.942	0.939
0.9	0.914	0.909

Table 12: Sensitivity to the EATR relation consistency threshold δ_{rel} on MQuAKE-hard.

fault configuration in our main experiments.

I Case Study

We present a qualitative case study to illustrate how CARE recovers from intermediate-hop failures in the post-edit world, where edited facts may conflict with locally plausible retrieval candidates. Figure 5 compares CARE with a strong graph-based baseline (KEDKG) on a 3-hop query. The post-edit knowledge contains the following facts: (i) World Indoor Soccer League is associated with baseball; (ii) Baseball was created in the United Kingdom; and (iii) the official language of the United Kingdom is Italian.

The query asks: "Which language is the official language of the country that originated the World Indoor Soccer League?" Answering it requires chaining league → sport → origin country → official language.

KEDKG retrieves correct evidence for hop 1 and hop 2 but fails at hop 3 by selecting a compet-

ing candidate (Greek) that conflicts with the edited fact. Because the baseline does not perform chain-level verification or backtracking, the erroneous intermediate choice propagates to the final prediction, yielding an incorrect answer (Greek). In contrast, CARE uses the SELECTOR to rank candidate facts at each hop while preserving alternatives, and applies the CRITIC for chain-level verification. At hop 3, both Greek and Italian appear plausible under local evidence. CARE may initially follow the top-ranked option, but the CRITIC flags the inconsistency at hop 3 and triggers bounded backtracking to re-select an alternative. After discarding the inconsistent candidate, CARE selects Italian and produces a coherent chain consistent with the post-edit world, leading to the correct final answer.

J Prompts of CARE

For reproducibility, we provide the full prompts used by CARE for the PLANNER, SELECTOR, and CRITIC agents in Figures 6–8.

PLANNER Prompt (Question Decomposition)

Instruction:
You need to divide a multi-hop question into several sub-questions.

Examples:

Question: Who is the head of state of the country where Rainn Wilson holds a citizenship?
Sub-questions:
- What is the country of citizenship of Rainn Wilson?
- What is the name of the current head of state in `[ENT]`?

Question: Who is the spouse of the head of state in United States of America?
Sub-questions:
- Who is the head of state in United States of America?
- Who is the spouse of `[ENT]`?

Question: What is the capital city of the country of citizenship of Ivanka Trump's spouse?
Sub-questions:
- Who is Ivanka Trump's spouse?
- What is the country of citizenship of `[ENT]`?
- What is the capital city of `[ENT]`?

Question: What is the name of the founder of the headquarters where the creators of Mortal Kombat II are located?
Sub-questions:
- Who is the developer of Mortal Kombat II?
- Which city is the headquarter of `[ENT]` located in?
- Who founded `[ENT]`?

Question: On which continent is the country of citizenship of the founder of the manufacturer of iPhone 5 situated?
Sub-questions:
- Which company is iPhone 5 produced by?
- Who is the founder of `[ENT]`?
- What is the country of citizenship of `[ENT]`?
- On which continent is `[ENT]`?

Input Question: ≪*QUESTION*≫
Sub-questions:

Figure 6: Prompt for PLANNER.

SELECTOR Prompt (Knowledge Triple Evaluation)

Task:
You are evaluating knowledge graph triples for a sub-question. Your task is to judge whether a candidate triple can be used as a direct answer, based ONLY on the triple itself and the evidence sentence.

Strict Rules:

- **Entity Match:** The main entity in the sub-question must appear as the correct entity (usually the head) in the triple. If not, score = 0.
- **Sentence-based Check:** If the evidence sentence explicitly states the answer (even if worded differently), score high. If not, score 0.
- **Relation Consistency:** The relation must be compatible with the question type (e.g., "located in" for location questions). Irrelevant relations get score 0.
- **Ignore Reality:** Do NOT use external knowledge. Treat the evidence sentence as authoritative, even if factually false in the real world.
- **Self-Reference:** Self-referential triples (tail == head) must receive score 0.

Output Format:
Return ONLY a JSON list: `[{"id": ..., "score": ...}]`

Input Data:
Sub-question: `{question_text}`
Candidate triple:

- Head: `{head_text}`
- Relation: `{relation_text}`
- Tail: `{tail_text}`
- Evidence sentence: `{evidence_text}`

Figure 7: Prompt for SELECTOR

CRITIC Prompt (Logic Audit)

Role & Task:
You are an expert Logic Auditor. Your task is to validate multi-hop reasoning chains in a COUNTERFACTUAL knowledge editing dataset.

Global Rules (Critical):

- IGNORE REALITY: You are in an alternate universe. The provided "Steps/Facts" are the ONLY truth.
- STRICT CONTINUITY: The entity produced as the answer in Step N must be the EXACT input entity used in Step $N + 1$.
- LOGICAL RELEVANCE: Each Fact must directly answer its own sub-question.

Audit Process:
For each chain, evaluate internally:

- Local Relevance Check: Does Fact i answer Sub-question i?
- Bridge Consistency Check: Does output of Step i match input of Step $i + 1$?
- Final Deduction Check: Does the Final Answer follow logically?

Output Format:
Return ONLY a raw JSON object: `{ "valid": boolean, "problem_steps": [list of indices], "explanation": "..." }`

Examples:

[Example 1 — VALID]
Original question: Who is the current CEO of the company that created the iPhone?
Sub-questions:

- Which company created the iPhone?
- Who is the CEO of `[ENT]`?

Chain facts:

- Step 1: iPhone –[created_by]→ McDonald's (Output: McDonald's)
- Step 2: McDonald's –[CEO]→ Ronald McDonald (Output: Ronald McDonald)

Final predicted answer: Ronald McDonald
Audit Output: `{ "valid": true, "problem_steps": [], "explanation": "The chain is logically consistent based on the provided counterfactual facts." }`

[Example 2 — INVALID]
Original question: Which political party is led by the president of the country where the Golden Gate Bridge is located?
[... omitted for brevity ...]

Task Data: [Input reasoning chain to be audited]

Figure 8: Prompt for CRITIC.