EAIR: Entity-aware Inference-Time Knowledge Routing for Multi-Hop Knowledge Editing

Jungyu Lee, Kunhui Lee, Gyun Lee, Seung-Hoon Na*
Graduate School of Artificial Intelligence,
Ulsan National Institute of Science and Technology
{steamy13133, nash}@unist.ac.kr

Abstract

Existing in-context editing (ICE) methods for multi-hop knowledge editing commonly suffer from *paraphrase sensitivity*, which refers to the phenomenon where these methods are not sufficiently robust to paraphrased multi-hop questions. To improve retrieval accuracy and knowledge routing to address paraphrase sensitivity, this paper proposes a novel entity-aware inference-time knowledge routing method, referred to as **EAIR**, which consists of four major steps: 1) **Entity-referential query decomposition**, which decomposes the original question into multiple entity-referential sub-question instructions; 2) **Entity-aware retrieval**, which leverages the previously reference-resolved topic entity in the retrieval step; 3) **Evidence-conditioned contrastive decoding**, which discourages the model from relying on its parametric knowledge and steers the model toward following retrieved edits; 4) **Reflection-based knowledge routing**, which additionally filters decoding results using refusal-style reflection to mitigate the risk introduced by contrastive decoding. Experimental results across the MQuAKE benchmark family and model scales show that EAIR achieves the highest strict case accuracy in 11 of 12 settings, substantially reducing paraphrase sensitivity. Our code is available at https://github.com/won12055/EAIR.

1 Introduction

Knowledge editing has emerged as a research area that addresses knowledge updating or injection in LLMs (Wang et al., 2024a; Yao et al., 2023). Existing approaches are commonly categorized into *parametric* and *non-parametric* methods (Yao et al., 2023). Parametric methods—such as *locate-then-edit* approaches (Dai et al., 2022; Meng et al., 2022, 2023), *parameter expansion* (Dong et al., 2022; Hartvigsen et al., 2023; Huang et al., 2023), and *meta-learning*-based methods (Mitchell et al., 2022a; Tan et al., 2024; Li et al., 2025)—have been extensively studied, however, these methods often interfere with the model's original knowledge and can degrade its inherent capabilities (Gu et al., 2024b; Yang et al., 2024). In contrast, non-parametric methods leverage the in-context learning capabilities of LLMs without modifying model parameters. Among them, *in-context knowledge editing (ICE)* expresses edits as context and relies on in-context learning to apply them (Zheng et al.,

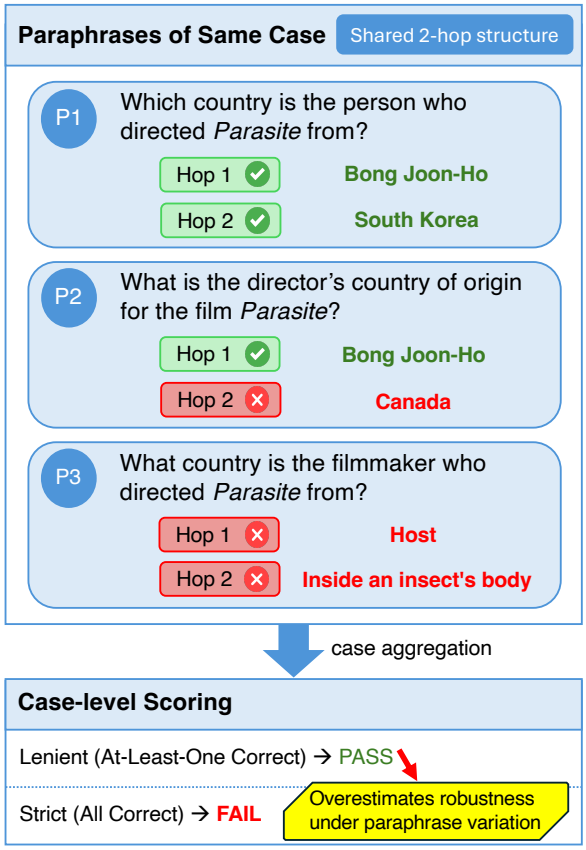

Figure 1: Paraphrase sensitivity in multi-hop QA: within a case, paraphrased queries can yield mixed correctness despite an identical 2-hop structure.

*Corresponding author

Findings of the Association for Computational Linguistics: ACL 2026, pages 17622–17645
July 2-7, 2026 ©2026 Association for Computational Linguistics

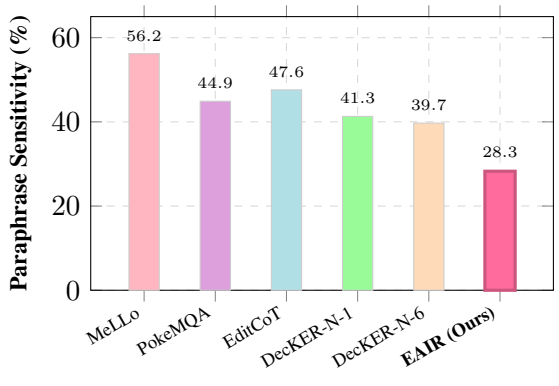

Figure 2: Paraphrase sensitivity on MQuAKE-T (Qwen 2.5-14B). EAIR achieves the lowest paraphrase sensitivity among evaluated methods.

2023; Shi et al., 2024b; Nafee et al., 2025).

Recently, ICE has been further advanced by incorporating chain-of-thought (CoT) reasoning (Wei et al., 2022) for multi-hop question answering (QA) (Zhao et al., 2024; Wu et al., 2025), yielding improved performance on MQuAKE (Zhong et al., 2023), a widely used benchmark for multi-hop knowledge editing. Despite these promising results, we find that existing CoT-based ICE methods are **not sufficiently robust to paraphrased multi-hop questions**, not only for DecKER (Wang et al., 2025b), but also for both classical and recent CoT-based ICE methods, including MeLLo (Zhong et al., 2023) and EditCoT (Wang et al., 2025a), as illustrated in Figure 1. In other words, these CoT-based ICE methods attain relatively high performance under what we refer to as ***lenient case accuracy***, which counts a case as correct if *any* paraphrase is answered correctly. However, their performance drops substantially under ***strict case accuracy***, which requires *all* paraphrases within a case to be answered correctly. We refer to the gap between ***lenient*** and ***strict*** case accuracy as ***paraphrase sensitivity***.

Under paraphrase variation, *retrieval* becomes more challenging, as surface variations can reduce the likelihood of retrieving relevant edited knowledge. This, in turn, increases the risk that ICE incorrectly *routes* the query to parametric knowledge by concluding that no relevant edits exist, even when updated knowledge is available. To improve *retrieval reliability* and *knowledge routing* under paraphrase variation, we propose **entity-aware inference-time knowledge routing (EAIR)**. Specifically, EAIR consists of four processing steps:

1. **Entity-referential query decomposition.** This step decomposes an original question into multiple *entity-referential sub-question instructions* using prompting-based methods, together with an initial *topic entity*. An entity-referential sub-question instruction is a sub-question form in which the topic entity may be specified either *explicitly* or *implicitly*.

2. **Entity-aware retrieval.** This step performs retrieval by explicitly conditioning on the entity resolved in the previous hop. Specifically, each entity-referential sub-question is first refined into a *reference-resolved sub-question* by appending the resolved entity, which is then used as the retrieval query. Since the resolved entity serves as the topic entity for the current sub-question, we first identify edits that explicitly mention this entity. The resulting set of *entity-matched edits* is treated as the initial candidate pool for retrieval.

3. **Evidence-conditioned contrastive decoding.** This step performs decoding by contrasting two responses from the same base LLM to a reference-resolved sub-question: (i) an *evidence-conditioned model*, which prepends the top-retrieved edits when generating an answer, and (ii) a *base (no-evidence) model*, which generates an answer without using any retrieved edits. This contrastive decoding strategy (Liu et al., 2021; Li et al., 2023; Shi et al., 2024a) is motivated by the tendency that LLMs often exhibit a strong bias toward parametric (pre-trained) knowledge.

4. **Reflection-based knowledge routing.** Retrieval errors can still occur for various reasons, leading the model to generate refusal-style responses *after contrastive decoding* even when conditioned on incorrect or irrelevant evidence. To address the risk from the retrieval error, this step rejects the *contrastive decoding* result when the generated output exhibits a *refusal-type* pattern rather than a valid answer form (Maskey et al., 2025; Madhusudhan et al., 2025).

Experimental results on the MQuAKE benchmark using three LLMs show that **EAIR** outperforms prior methods in 11 of 12 settings, achieving the highest strict case accuracy in these comparisons. These improvements are primarily achieved

by substantially reducing *paraphrase sensitivity*, as illustrated in Figure 2.

2 Preliminaries

2.1 Problem Setup

We formally describe multi-hop QA under knowledge editing in the MQuAKE setting (Zhong et al., 2023). Let M denote a base language model. At inference time, we are given a global set of *single-hop edit requests* $\mathcal{E} = \{(s_i, r_i, o_i \rightarrow o_i^*)\}_{i=1}^{n}$, where each tuple specifies a subject s_i, a relation r_i, an original object o_i, and its edited object o_i^*. That is, $o_i \rightarrow o_i^*$ replaces the object of the fact (s_i, r_i, o_i) with o_i^*. Each request can be equivalently expressed as a natural-language edit statement. For example, (United Kingdom, head of government, Boris Johnson $\rightarrow$ Rishi Sunak) corresponds to *"The head of government of the United Kingdom is Rishi Sunak."*

Given a multi-hop question q, the goal is to produce the final answer through a sequence of intermediate entities $(o_0, \ldots, o_k)$, where k is the number of hops in q, o_0 is the initial topic entity, and each hop depends on the previous-hop entity.

MQuAKE provides cases $\mathcal{C}$, where each case $\mathcal{Q}_j \in \mathcal{C}$ contains K paraphrased questions $\mathcal{Q}_j = \{q_{j,1}, \ldots, q_{j,K}\}$ that share a single ground-truth answer $a_j^\star$. The goal is to answer all paraphrases consistently under the same edit set $\mathcal{E}$.

2.2 Case-level Metrics

To formally define *paraphrase sensitivity*, we consider two case-level evaluation metrics for each paraphrase set $\mathcal{Q}_j$: *strict* and *lenient* case accuracy.

- **Strict case accuracy** ($\mathrm{Acc}_{\mathrm{strict}}$). A case is counted as correct under the *strict* criterion only if *all* paraphrased questions in $\mathcal{Q}_j$ are answered correctly.
- **Lenient case accuracy** ($\mathrm{Acc}_{\mathrm{lenient}}$). A case is counted as correct under the *lenient* criterion if *at least one* paraphrased question $q_{j,i} \in \mathcal{Q}_j$ is answered correctly.

Paraphrase sensitivity (PS).

$$\mathrm{PS} = \mathrm{Acc}_{\mathrm{lenient}} - \mathrm{Acc}_{\mathrm{strict}}. \quad (1)$$

Equivalently, PS measures the fraction of cases in which at least one paraphrase succeeds but not all do, reflecting case-level inconsistency across paraphrases.

Throughout this paper, strict case accuracy is the primary metric, and PS is reported as a supporting indicator of case-level consistency.

3 Method

EAIR is designed to reduce hop-wise error propagation by explicitly structuring **knowledge routing** between two information sources. For each sub-query, EAIR retrieves a compact neighborhood of candidate edits from a global edit set $\mathcal{E}$. At decoding time, EAIR determines whether generation should be guided by the retrieved edits (*with-evidence*) or by the model's parametric prior (*no-evidence*). This separation keeps retrieval lightweight while allowing the model to resolve evidence reliability precisely at the point where each hop answer is produced.

As a result, the overall architecture of the proposed **EAIR** is illustrated in Figure 3. EAIR consists of four main components: (i) *Entity-referential query decomposition*, (ii) *Entity-aware retrieval*, (iii) *Evidence-conditioned contrastive decoding*, and (iv) *Reflection-based knowledge routing*.

Pseudocode is provided in Appendix A, and a full step-by-step workflow is given in Appendix N. We additionally provide qualitative failure cases in Appendix D.

3.1 Entity-referential query decomposition: Decompose

EAIR first applies a hop planner. Given an input question q, the planner produces a tentative hop count $\tilde{k}$ and a sequence of *sub-question instructions* $(sq_1, \ldots, sq_{\tilde{k}})$.

Each sub-question instruction sq_t summarizes the sub-goal to be addressed at hop t and includes a topic entity that may be specified either explicitly or implicitly. In the implicit case, the entity is assumed to be obtained from the previous hop and is therefore referenced via a referential expression. For example, given the sub-question instruction *"identify the country of citizenship of that discoverer"*, the phrase *"that discoverer"* denotes an implicit entity that must be resolved from the previous-hop answer, such as *"Louis Pasteur."* If the initial hop decomposition fails, we set $sq_1 = q$.

In addition, we extract the initial topic entity o_0, by applying a lightweight entity-extraction prompt to the surface question q. If extraction fails, we set $o_0 = \epsilon$. Here, ϵ denotes an empty entity string.

Detailed hop-planner, entity-extraction prompt,

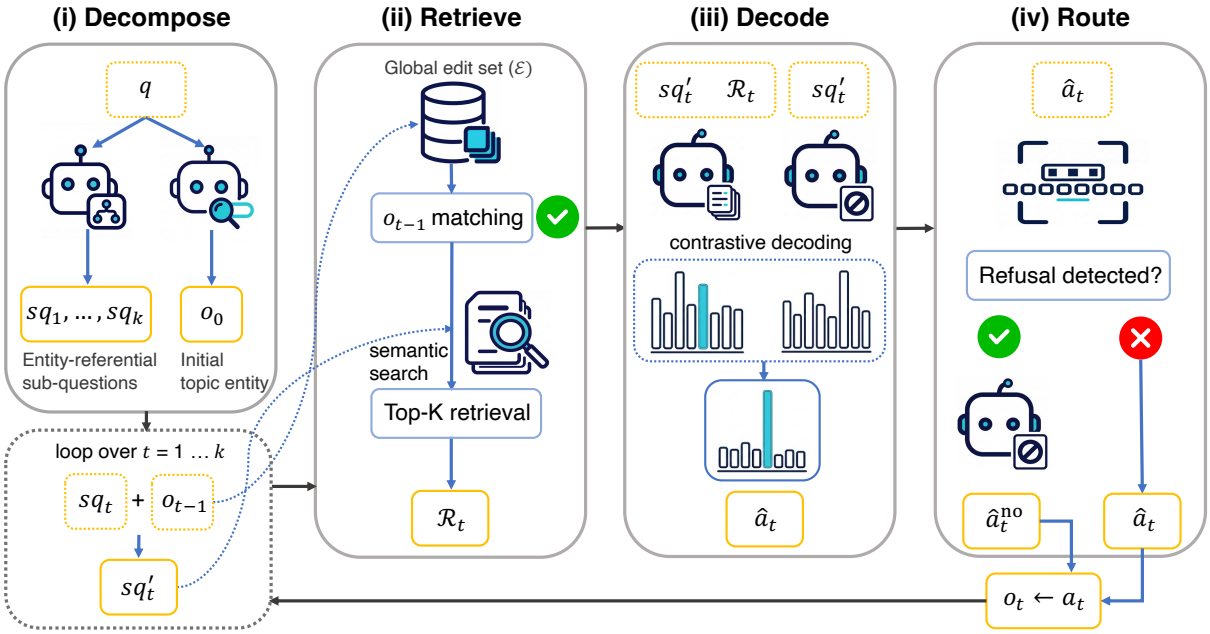

Figure 3: **EAIR** overview at a single hop t. (i) **Decompose** maps the surface question q to entity-referential sub-question instructions $(sq_1, \ldots, sq_k)$ and an initial topic entity o_0. (ii) **Retrieve** forms a reference-resolved query sq'_t by appending a short context from the previous-hop entity ($sq'_t = sq_t$ for t=1, else $sq_t \oplus \mathrm{Ctx}(o_{t-1})$), filters the global edit set $\mathcal{E}$ via exact entity matching, and performs dense Top-K retrieval to obtain $\mathcal{R}_t$. (iii) **Decode** runs the base LLM on two prompts—an *evidence-conditioned* prompt that prepends $\mathcal{R}_t$ to sq'_t and a *no-evidence* prompt that asks sq'_t alone—and applies contrastive decoding to obtain a candidate hop answer $\hat{a}_t$. (iv) **Route** applies a rule-based refusal predicate to $\hat{a}_t$ and, when refusal-style markers are detected, falls back to a no-evidence decoding output $\hat{a}_t^{\text{no}}$; otherwise it accepts $\hat{a}_t$. The accepted answer is fed forward as the next-hop entity ($o_t \leftarrow a_t$) the procedure is iterated for $t = 1, \ldots, k$. A full step-by-step end-to-end workflow is provided in Appendix N.

and analysis are provided in Appendix B and K.

3.2 Entity-aware Retrieval: Retrieve

At the t-th hop, the sub-question instruction sq_t is resolved by incorporating the previous hop answer o_{t-1} when $t > 1$. Specifically, we form a *reference-resolved sub-question*, denoted by sq'_t, which is used as the retrieval query:

$$sq'_t = \begin{cases} sq_t, & t = 1, \\ sq_t \oplus \mathrm{Ctx}(o_{t-1}), & t > 1, \end{cases} \quad (2)$$

where $\oplus$ denotes string concatenation and $\mathrm{Ctx}(\cdot)$ formats a short contextual suffix.

Entity exact matching. To improve retrieval accuracy, we first apply *entity exact matching* to filter the edit set by selecting only those edits that explicitly mention the topic entity o_{t-1}. Formally, the entity-matched edit subset is defined as

$$\tilde{\mathcal{E}}(o_{t-1}) = \{\, e_i \in \mathcal{E} \mid o_{t-1} \text{ appears in } e_i \,\}. \quad (3)$$

If $o_{t-1} = \epsilon$, we disable entity filtering by setting $\tilde{\mathcal{E}}(o_{t-1}) = \emptyset$.

Entity-matched candidate pool. Based on the entity-matched edit set, we construct the candidate pool $\hat{\mathcal{E}}_t$ as follows:

$$\hat{\mathcal{E}}_t = \begin{cases} \tilde{\mathcal{E}}(o_{t-1}), & \text{if } \tilde{\mathcal{E}}(o_{t-1}) \neq \emptyset, \\ \mathcal{E}, & \text{otherwise}, \end{cases}, \; t > 1. \quad (4)$$

At $t = 1$, if the initial entity is available ($o_0 \neq \epsilon$) but no edit matches it, we treat the query as unaffected by knowledge edits and skip retrieval by setting $\hat{\mathcal{E}}_1 = \emptyset$. If $o_0 = \epsilon$, we do not apply this $t = 1$ exception and instead retrieve from the full edit set $\mathcal{E}$.

Dense retrieval. Given a candidate pool, we apply dense retrieval between the *reference-resolved sub-question* sq'_t and each candidate edit $e \in \hat{\mathcal{E}}_t$ using Contriever (Izacard et al., 2022). Specifically, we obtain embeddings $\mathsf{Emb}(sq'_t)$ and $\mathsf{Emb}(e)$, and score each candidate by the dot product:

$$s_t(e) = \mathsf{Emb}(sq'_t)^\top \mathsf{Emb}(e), \qquad e \in \hat{\mathcal{E}}_t. \quad (5)$$

We then select the K_{ev} highest-scoring edits as the hop-wise retrieved neighborhood:

$$\mathcal{R}_t = \mathrm{TopK}_{K_{\text{ev}}}(\hat{\mathcal{E}}_t;\, \boldsymbol{s}_t), \qquad \mathcal{R}_t \subseteq \mathcal{E}, \quad (6)$$

where $\boldsymbol{s}_t = \{ s_t(e) \}_{e \in \hat{\mathcal{E}}_t}$.

We provide an in-depth analysis of retrieval signals and evidence retrieval quality in Appendix L.

3.3 Evidence-conditioned contrastive decoding: Decode

At hop t, given the *reference-resolved sub-question* sq'_t and the retrieved neighborhood $\mathcal{R}_t$ (§3.2), we construct two prompts: (i) an *evidence* prompt that prepends $\mathcal{R}_t$ before sq'_t, and (ii) a *no-evidence* prompt that asks sq'_t without any retrieved evidence (Appendix B).

We run the base LLM on both prompts and apply contrastive decoding over the evidence-conditioned and no-evidence distributions (Krause et al., 2021; Yang and Klein, 2021). If $\mathcal{R}_t = \emptyset$, we decode using the no-evidence prompt only.

Formally, let $\boldsymbol{\ell}_\tau^{\text{ev}}, \boldsymbol{\ell}_\tau^{\text{no}} \in \mathbb{R}^{|\mathcal{V}|}$ denote the logit vectors at decoding step τ under the evidence and no-evidence prompts, respectively, where $\mathcal{V}$ is the vocabulary. We define $\boldsymbol{p}_\tau^{\text{ev}} = \text{softmax}(\boldsymbol{\ell}_\tau^{\text{ev}})$ and $\boldsymbol{p}_\tau^{\text{no}} = \text{softmax}(\boldsymbol{\ell}_\tau^{\text{no}})$.

Contrastive decoding (Liu et al., 2021; Li et al., 2023; Shi et al., 2024a) computes the steered logits using a contrast strength $\alpha > 0$ as

$$\tilde{\boldsymbol{\ell}}_\tau = (1+\alpha)\,\boldsymbol{\ell}_\tau^{\text{ev}} - \alpha\,\boldsymbol{\ell}_\tau^{\text{no}}. \tag{7}$$

3.4 Reflection-based knowledge routing: Route

When the retrieved edits $\mathcal{R}_t$ are irrelevant to the sub-question sq'_t, the *evidence* run often produces *refusal-style* statements about insufficient or inconsistent evidence instead of a valid single-entity answer. Thus, if $\hat{a}_t$ is refusal-style, we discard it and answer the hop using the *no-evidence* run. Otherwise, we accept $\hat{a}_t$ as the hop answer. An in-depth analysis of this reflection-based routing behavior is provided in Appendix M.

4 Experiments

Benchmarks and global edit set. We evaluate on the MQuAKE benchmark family, including the counterfactual variants MQuAKE-CF-3k-v2 and CF-3k-Remastered, as well as the Temporal variants (T and T-Remastered) (Zhong et al., 2023, 2025). For each benchmark, we build a single global edit set $\mathcal{E}$ by pooling edit statements from all cases and share $\mathcal{E}$ across all queries. The resulting edit-set sizes are 2,764 (CF-3k-v2), 2,791 (CF-3k-Remastered), and 96 for each Temporal split (T and T-Remastered). Dataset statistics are provided in Appendix E.

Models. We evaluate Llama 3.1-8B-Instruct (Grattafiori et al., 2024), Qwen 2.5-7B-Instruct, and Qwen 2.5-14B-Instruct (Qwen et al., 2025). All methods are evaluated without updating model parameters.

Baselines. We compare against MeLLo (Zhong et al., 2023), PokeMQA (Gu et al., 2024a), EditCoT (Wang et al., 2025a), and DecKER (Wang et al., 2025b). We use the authors' released code and follow their reported protocols; when unspecified, we keep default settings. Details are in Appendix C.

Metrics and implementation. In MQuAKE, each case contains K=3 paraphrases (Zhong et al., 2023). We report strict and lenient case accuracy; paraphrase sensitivity (PS) is the Lenient–Strict gap. For retrieval, we use Contriever (Izacard et al., 2022) with dot-product scoring and K_{ev}=8.

For decoding stability, we use greedy contrastive decoding (Eq. 7) with a single fixed strength parameter α=0.3 for all main results across the MQuAKE benchmark family. We fix these hyperparameters in the main experiments for comparability and report further analysis for K_{ev} and α in Appendix G and Appendix H.

4.1 Main results

Table 1 summarizes strict and lenient case accuracy across four MQuAKE variants (CF-3k-v2, CF-3k-Remastered, T, and T-Remastered) and three model scales. All methods are evaluated with one generation per paraphrased question, except DecKER-N-6 which uses multiple generations (Wang et al., 2025b). EAIR achieves the highest strict case accuracy in 11 of 12 model–benchmark settings, demonstrating robust case-level consistency under a strict objective.

On the counterfactual variants, EAIR consistently improves strict case accuracy over all baselines across model scales, with the largest gains observed on the smallest model. EAIR also remains competitive on Lenient, indicating that the strict case accuracy improvements are not obtained by sacrificing overall answerability.

On the Temporal variants, EAIR remains strong and achieves the best strict case accuracy in 5 of 6 settings. The only exception is Llama 3.1-8B on T-Remastered, where PokeMQA is higher by 0.97 points.

Method	CF-3k-v2		T		CF-3k-RM		T-RM	
	Strict	Lenient	Strict	Lenient	Strict	Lenient	Strict	Lenient
Qwen 2.5-7B-Instruct								
MeLLo	0.53	9.77	14.83	68.74	0.40	9.37	21.14	71.89
PokeMQA	9.80	38.30	24.95	69.65	14.23	42.03	50.59	84.98
EditCoT	7.07	40.27	22.86	75.54	7.83	41.20	43.88	84.17
DecKER-N-1	9.87	42.70	31.69	68.04	11.03	40.23	40.40	76.34
DecKER-N-6	15.33	49.60	38.33	70.29	16.63	48.27	52.52	80.31
EAIR (Ours)	**31.56**	**54.76**	**56.20**	**85.33**	**35.40**	**50.23**	**68.83**	**85.94**
Llama 3.1-8B-Instruct								
MeLLo	7.80	31.17	16.22	70.82	7.33	26.83	26.50	70.23
PokeMQA	13.30	40.77	39.45	85.44	20.63	43.57	**70.76**	**92.76**
EditCoT	8.83	42.70	19.06	74.33	12.13	45.03	33.32	85.44
DecKER-N-1	24.97	57.97	44.22	80.67	29.03	52.27	60.94	82.56
DecKER-N-6	28.47	**60.17**	43.74	81.48	31.27	**53.83**	62.02	85.78
EAIR (Ours)	**31.26**	59.40	**57.60**	**87.47**	**35.06**	52.16	69.79	83.10
Qwen 2.5-14B-Instruct								
MeLLo	8.93	37.70	22.06	78.27	11.07	40.03	36.70	86.86
PokeMQA	6.47	30.90	23.13	67.99	12.07	34.40	45.17	81.17
EditCoT	8.73	41.47	34.85	82.44	9.70	42.90	62.98	92.17
DecKER-N-1	22.13	50.20	40.20	81.53	26.20	50.00	62.34	89.11
DecKER-N-6	25.80	**57.13**	44.22	83.94	28.20	**54.57**	68.88	90.88
EAIR (Ours)	**27.16**	55.93	**60.54**	**88.81**	**32.30**	49.80	**77.78**	**91.84**

Table 1: Main performance comparison on MQuAKE and MQuAKE-RM (Remastered) datasets. Strict and lenient case accuracy is reported in % (K_{ev}=8, α=0.3).

DecKER-N-6 (DecKER-BoN) may remain competitive on Lenient because its best-of-N inference can increase the chance that at least one paraphrase succeeds, whereas Strict additionally requires cross-paraphrase consistency. In contrast, PokeMQA's gains may be more setting-dependent, reflecting potential sensitivity to its decomposition and intermediate-answering pipeline, while EAIR's Strict improvements can appear comparatively stable across model scales and benchmark variants.

4.2 Analysis

Paraphrase sensitivity. Paraphrase sensitivity (PS) is defined as the Lenient–Strict gap and is used as a supporting indicator of case-level consistency across paraphrases. PS can be misleading when both Strict and Lenient are near zero, since a small gap may simply reflect rare successes on any paraphrase. This pattern appears for weak methods such as MeLLo (Zhong et al., 2023) on the counterfactual splits, where low PS reflects low overall success rather than consistency.

When accuracy is non-trivial, PS is more informative because it reflects within-case inconsistency: succeeding on some paraphrases but failing on others. Table 2 reports Strict and PS and compares EAIR against DecKER-N-6, the strongest baseline under the Strict objective. Across all model–benchmark settings, EAIR improves strict case accuracy and reduces PS, consistent with more stable case-level behavior across paraphrases.

Efficiency. We measure wall-clock latency per case to quantify inference-time overhead. As shown in Table 3, EAIR yields the lowest latency among all methods, while DecKER slows down substantially as the number of generation attempts increases. Appendix F provides a detailed cost breakdown of EAIR.

Performance Breakdown by Hop Count. Figure 4 reports strict case accuracy by hop count on Qwen 2.5-7B. CF-3k-v2 is imbalanced ($N_{2/3/4} = 1{,}135/1{,}136/729$), while CF-3k-Remastered is balanced ($N = 1{,}000$ per hop).

Across both splits, baselines differ mainly on 2-hop cases, where performance varies substantially

Model	Benchmark	EAIR (Ours)		DecKER-N-6		Improvement (Δ)	
		Strict	PS	Strict	PS	Strict ↑	PS ↓
Qwen 2.5-7B	CF-3k-v2	31.56	23.20	15.33	34.27	**+16.23**	**−11.07**
	T	56.20	29.13	38.33	31.96	**+17.87**	**−2.83**
	CF-3k-RM	35.40	14.83	16.63	31.64	**+18.77**	**−16.81**
	T-RM	68.83	17.11	52.52	27.79	**+16.31**	**−10.68**
Llama 3.1-8B	CF-3k-v2	31.26	28.14	28.47	31.70	**+2.79**	**−3.56**
	T	57.60	29.87	43.74	37.74	**+13.86**	**−7.87**
	CF-3k-RM	35.06	17.10	31.27	22.56	**+3.79**	**−5.46**
	T-RM	69.79	13.31	62.02	23.76	**+7.77**	**−10.45**
Qwen 2.5-14B	CF-3k-v2	27.16	28.77	25.80	31.33	**+1.36**	**−2.56**
	T	60.54	28.27	44.22	39.72	**+16.32**	**−11.45**
	CF-3k-RM	32.30	17.50	28.20	26.37	**+4.10**	**−8.87**
	T-RM	77.78	14.06	68.88	22.00	**+8.90**	**−7.94**

Table 2: Strict case accuracy and paraphrase sensitivity (PS) on MQuAKE variants. PS is computed as $Lenient - Strict$. EAIR improves strict case accuracy and reduces PS (improves consistency) across all benchmarks compared to DecKER-N-6.

Method	Relative Speed	Time (s)
MeLLo		6.42
DecKER-N-6		4.94
EditCoT		4.15
PokeMQA		3.27
DecKER-N-1		2.62
EAIR (Ours)		**2.46**

Table 3: Wall-clock latency per case on MQuAKE-CF-3k-v2 using Llama 3.1-8B on a single NVIDIA H200 GPU, averaged over the first 100 cases (out of 3,000).

Metric	Base	–Entity	–GlobalFB	–Refusal
Qwen 2.5-7B-Instruct				
Strict	**31.56**	24.70	31.10	30.96
Lenient	**54.76**	48.83	53.67	53.06
Llama 3.1-8B-Instruct				
Strict	**31.26**	25.46	30.83	27.53
Lenient	**59.40**	58.06	57.80	49.83
Qwen 2.5-14B-Instruct				
Strict	**27.16**	24.90	26.00	25.80
Lenient	**55.93**	54.30	53.60	49.46

Table 4: Ablation study of EAIR on MQuAKE-CF-3k-v2 (case accuracy, %). **–Entity**: Disabling entity exact matching during retrieval (§3.2). **–GlobalFB**: Removing the global similarity-search fallback when no entity match is found (§3.2). **–Refusal**: Disabling reflection-based knowledge routing via refusal markers (§3.4).

by method. At 3 and 4 hops, baseline strict case accuracy largely converges at similarly low levels, suggesting shared difficulty in propagating edits through longer chains. In contrast, EAIR can retain a clear advantage not only at 2 hops but also on 3- and 4-hop cases, maintaining separation where other methods converge. This pattern is consistent with EAIR improving strict-case consistency on longer chains, rather than benefiting only from easier 2-hop cases. Detailed hop-count values are reported in Appendix I.

Ablations. Table 4 summarizes the contribution of each component in EAIR. Our analysis yields three key findings: (1) Entity matching is essential for consistency: Disabling entity matching (–Entity) leads to the most significant drop in strict case accuracy across all models. This confirms that entity-based filtering acts as a high-precision anchor that suppresses spurious retrievals. (2) Global fallback provides robust recovery: While entity matching handles most cases, the consistent performance decrease in –GlobalFB indicates its necessity as a secondary recovery mechanism. (3) Reflection prevents reasoning failures: Removing reflection-based knowledge routing (–Refusal) causes a substantial decline in lenient case accuracy. This suggests that the refusal logic serves as a functional safeguard, discouraging the model from committing to incorrect evidence when retrieved edits are irrelevant. Appendix J further breaks these results down by hop count, showing how the effect of each safeguard appears separately on 2-, 3-, and

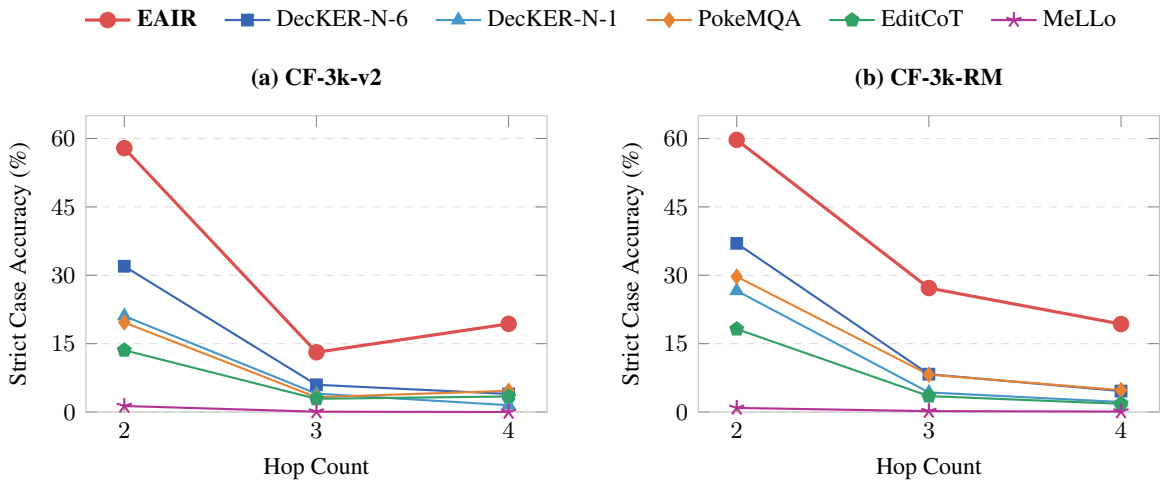

Figure 4: Performance breakdown by reasoning hop count on Qwen 2.5-7B-Instruct for the MQuAKE-CF variants (CF-3k-v2 and CF-3k-Remastered).

4-hop questions and providing a reference comparison against DecKER-N-6.

5 Related Work

Knowledge editing: parametric vs. non-parametric. Knowledge editing methods are commonly divided into *parametric* approaches (Meng et al., 2022) that update model weights and *non-parametric* approaches (Mitchell et al., 2022b) that keep parameters fixed. We focus on non-parametric *in-context* editing without parameter updates (Zheng et al., 2023).

Multi-hop in-context editing and paraphrase consistency. Multi-hop question answering has been widely studied in prior work (Yang et al., 2018; Cohen et al., 2024). MQuAKE frames multi-hop knowledge editing with multiple paraphrased queries per case and evaluates multi-hop accuracy in a lenient manner, exposing paraphrase-level instability under hop-wise pipelines (Zhong et al., 2023). Representative multi-hop in-context editing methods build on decomposition and stepwise control: MeLLo (Zhong et al., 2023) and PokeMQA (Gu et al., 2024a) retrieve and inject edits along hop trajectories, EditCoT (Wang et al., 2025a) intervenes on intermediate reasoning traces, and DecKER (Wang et al., 2025b) decouples planning from knowledge injection with multi-trace selection. These pipelines can remain sensitive to paraphrase-induced variation in hop cues and retrieved candidates, motivating methods that improve case-level consistency under paraphrase variation.

Decoding-time control for edited evidence. Context-aware decoding (CAD) (Shi et al., 2024a) contrasts evidence-conditioned and no-evidence token distributions at decoding time to increase faithfulness to provided evidence. DeCK (Bi et al., 2025) adapts contrastive decoding (Li et al., 2023) as a *plug-in* rule that strengthens edited evidence during generation, while DeepEdit (Wang et al., 2024b) frames editing as constrained decoding via step-level search. Complementary to prior decoding-time control (Dathathri et al., 2020; Chuang et al., 2024), we target multi-hop retrieval where evidence reliability is uncertain: we treat retrieval as candidate generation and use evidence-conditioned contrastive decoding with reflection-based routing to improve case-level consistency.

6 Conclusion

We formalize **paraphrase sensitivity** in multi-hop in-context knowledge editing and treat **strict** case accuracy as the primary objective for case-level consistency under paraphrase variation. We propose **EAIR**, which combines entity-referential query decomposition, entity-aware retrieval, evidence-conditioned contrastive decoding, and reflection-based knowledge routing. Across multiple LLM families and MQuAKE variants, EAIR improves **strict** case accuracy while reducing paraphrase sensitivity.

7 Limitations

We evaluate EAIR on the MQuAKE benchmark family (Zhong et al., 2023, 2025), which supports

strict case-level scoring under paraphrase variation. While the remastered variants reduce some artifacts, they remain within the same benchmark family; thus, EAIR's out-of-family generalization is not yet fully characterized. Extending strict-scoring evaluations to additional multi-hop settings would strengthen conclusions about robustness.

In addition to benchmark coverage, EAIR inherits limitations from its modular pipeline design. EAIR uses a lightweight, prompt-based hop planner with simple fallbacks when decomposition fails. This design is practical but can be sensitive to decomposition errors, which may propagate across hops. EAIR reduces such propagation by combining entity-aware retrieval, evidence-conditioned contrastive decoding, and reflection-based knowledge routing. Strengthening the decomposition module is complementary to EAIR and may further improve strict case-level consistency.

8 Acknowledgments

This work was supported by the Institute of Information & Communications Technology Planning & Evaluation (IITP) grant funded by the Korea government (MSIT) (No. RS-2020-II201336, Artificial Intelligence Graduate School Support (UNIST)) and by an IITP grant funded by the Korea government (MSIT) (No. RS-2023-00216011, Development of Artificial Complex Intelligence for Conceptually Understanding and Inferring like Human).

References

Baolong Bi, Shenghua Liu, Lingrui Mei, Yiwei Wang, Junfeng Fang, Pengliang Ji, and Xueqi Cheng. 2025. Decoding by contrasting knowledge: Enhancing large language model confidence on edited facts. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 17198–17208, Vienna, Austria. Association for Computational Linguistics.

Yung-Sung Chuang, Yujia Xie, Hongyin Luo, Yoon Kim, James R. Glass, and Pengcheng He. 2024. Dola: Decoding by contrasting layers improves factuality in large language models. In *The Twelfth International Conference on Learning Representations*.

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.

Damai Dai, Li Dong, Yaru Hao, Zhifang Sui, Baobao Chang, and Furu Wei. 2022. Knowledge neurons in pretrained transformers. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 8493–8502, Dublin, Ireland. Association for Computational Linguistics.

Sumanth Dathathri, Andrea Madotto, Janice Lan, Jane Hung, Eric Frank, Piero Molino, Jason Yosinski, and Rosanne Liu. 2020. Plug and play language models: A simple approach to controlled text generation. In *International Conference on Learning Representations*.

Qingxiu Dong, Damai Dai, Yifan Song, Jingjing Xu, Zhifang Sui, and Lei Li. 2022. Calibrating factual knowledge in pretrained language models. *Preprint*, arXiv:2210.03329.

Aaron Grattafiori, Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Alex Vaughan, Amy Yang, Angela Fan, Anirudh Goyal, Anthony Hartshorn, Aobo Yang, Archi Mitra, Archie Sravankumar, Artem Korenev, Arthur Hinsvark, and 542 others. 2024. The llama 3 herd of models. *Preprint*, arXiv:2407.21783.

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*, pages 16801–16819, Miami, Florida, USA. Association for Computational Linguistics.

Tom Hartvigsen, Swami Sankaranarayanan, Hamid Palangi, Yoon Kim, and Marzyeh Ghassemi. 2023. Aging with grace: Lifelong model editing with discrete key-value adaptors. In *Advances in Neural Information Processing Systems*, volume 36, pages 47934–47959. Curran Associates, Inc.

Zeyu Huang, Yikang Shen, Xiaofeng Zhang, Jie Zhou, Wenge Rong, and Zhang Xiong. 2023. Transformer-patcher: One mistake worth one neuron. In *The Eleventh International Conference on Learning Representations*.

Gautier Izacard, Mathilde Caron, Lucas Hosseini, Sebastian Riedel, Piotr Bojanowski, Armand Joulin, and Edouard Grave. 2022. Unsupervised dense information retrieval with contrastive learning. *Transactions on Machine Learning Research*.

Ben Krause, Akhilesh Deepak Gotmare, Bryan McCann, Nitish Shirish Keskar, Shafiq Joty, Richard Socher,

and Nazneen Fatema Rajani. 2021. GeDi: Generative discriminator guided sequence generation. In *Findings of the Association for Computational Linguistics: EMNLP 2021*, pages 4929–4952, Punta Cana, Dominican Republic. Association for Computational Linguistics.

Xiang Lisa Li, Ari Holtzman, Daniel Fried, Percy Liang, Jason Eisner, Tatsunori Hashimoto, Luke Zettlemoyer, and Mike Lewis. 2023. Contrastive decoding: Open-ended text generation as optimization. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 12286–12312, Toronto, Canada. Association for Computational Linguistics.

Xiaopeng Li, Shasha Li, Xi Wang, Shezheng Song, Bin Ji, Shangwen Wang, Jun Ma, Xiaodong Liu, Mina Liu, and Jie Yu. 2025. Emsedit: Efficient multi-step meta-learning-based model editing. *Preprint*, arXiv:2508.04012.

Alisa Liu, Maarten Sap, Ximing Lu, Swabha Swayamdipta, Chandra Bhagavatula, Noah A. Smith, and Yejin Choi. 2021. DExperts: Decoding-time controlled text generation with experts and anti-experts. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 6691–6706, Online. Association for Computational Linguistics.

Nishanth Madhusudhan, Sathwik Tejaswi Madhusudhan, Vikas Yadav, and Masoud Hashemi. 2025. Do LLMs know when to NOT answer? investigating abstention abilities of large language models. In *Proceedings of the 31st International Conference on Computational Linguistics*, pages 9329–9345, Abu Dhabi, UAE. Association for Computational Linguistics.

Utsav Maskey, Mark Dras, and Usman Naseem. 2025. Steering over-refusals towards safety in retrieval augmented generation. *Preprint*, arXiv:2510.10452.

Kevin Meng, David Bau, Alex J Andonian, and Yonatan Belinkov. 2022. Locating and editing factual associations in GPT. In *Advances in Neural Information Processing Systems*.

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*.

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. *Preprint*, arXiv:2206.06520.

Mahmud Wasif Nafee, Maiqi Jiang, Haipeng Chen, and Yanfu Zhang. 2025. Dynamic retriever for in-context knowledge editing via policy optimization. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pages 16744–16757, Suzhou, China. Association for Computational Linguistics.

Qwen, :, An Yang, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chengyuan Li, Dayiheng Liu, Fei Huang, Haoran Wei, Huan Lin, Jian Yang, Jianhong Tu, Jianwei Zhang, Jianxin Yang, Jiaxi Yang, Jingren Zhou, and 25 others. 2025. Qwen2.5 technical report. *Preprint*, arXiv:2412.15115.

Weijia Shi, Xiaochuang Han, Mike Lewis, Yulia Tsvetkov, Luke Zettlemoyer, and Wen-tau Yih. 2024a. Trusting your evidence: Hallucinate less with context-aware decoding. In *Proceedings of the 2024 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 2: Short Papers)*, pages 783–791, Mexico City, Mexico. Association for Computational Linguistics.

Yucheng Shi, Qiaoyu Tan, Xuansheng Wu, Shaochen Zhong, Kaixiong Zhou, and Ninghao Liu. 2024b. Retrieval-enhanced knowledge editing in language models for multi-hop question answering. *Preprint*, arXiv:2403.19631.

Chenmien Tan, Ge Zhang, and Jie Fu. 2024. Massive editing for large language models via meta learning. *Preprint*, arXiv:2311.04661.

Changyue Wang, Weihang Su, Qingyao Ai, Yichen Tang, and Yiqun Liu. 2025a. Knowledge editing through chain-of-thought. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pages 10684–10704, Suzhou, China. Association for Computational Linguistics.

Changyue Wang, Weihang Su, Qingyao Ai, Yujia Zhou, and Yiqun Liu. 2025b. Decoupling reasoning and knowledge injection for in-context knowledge editing. In *Findings of the Association for Computational Linguistics: ACL 2025*, pages 24543–24562, Vienna, Austria. Association for Computational Linguistics.

Song Wang, Yaochen Zhu, Haochen Liu, Zaiyi Zheng, Chen Chen, and Jundong Li. 2024a. Knowledge editing for large language models: A survey. *Preprint*, arXiv:2310.16218.

Yiwei Wang, Muhao Chen, Nanyun Peng, and Kai-Wei Chang. 2024b. 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*.

Yuchen Wu, Liang Ding, Li Shen, and Dacheng Tao. 2025. Robust knowledge editing via explicit reasoning chains for distractor-resilient multi-hop QA. In *Findings of the Association for Computational Linguistics: EMNLP 2025*, pages 14578–14586, Suzhou, China. Association for Computational Linguistics.

Kevin Yang and Dan Klein. 2021. FUDGE: Controlled text generation with future discriminators. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 3511–3535, Online. Association for Computational Linguistics.

Wanli Yang, Fei Sun, Jiajun Tan, Xinyu Ma, Du Su, Dawei Yin, and Huawei Shen. 2024. The fall of ROME: Understanding the collapse of LLMs in model editing. In *Findings of the Association for Computational Linguistics: EMNLP 2024*, pages 4079–4087, Miami, Florida, USA. Association for Computational Linguistics.

Zhilin Yang, Peng Qi, Saizheng Zhang, Yoshua Bengio, William Cohen, Ruslan Salakhutdinov, and Christopher D. Manning. 2018. HotpotQA: A dataset for diverse, explainable multi-hop question answering. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 2369–2380, Brussels, Belgium. Association for Computational Linguistics.

Yunzhi Yao, Peng Wang, Bozhong Tian, Siyuan Cheng, Zhoubo Li, Shumin Deng, Huajun Chen, and Ningyu Zhang. 2023. Editing large language models: Problems, methods, and opportunities. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 10222–10240, Singapore. Association for Computational Linguistics.

Zihao Zhao, Yuchen Yang, Yijiang Li, and Yinzhi Cao. 2024. RippleCOT: Amplifying ripple effect of knowledge editing in language models via chain-of-thought in-context learning. In *Findings of the Association for Computational Linguistics: EMNLP 2024*, pages 6337–6347, Miami, Florida, USA. Association for Computational Linguistics.

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

Shaochen Zhong, Yifan Lu, Lize Shao, Bhargav Bhushanam, Xiaocong Du, Yixin Wan, Yucheng Shi, Daochen Zha, Yiwei Wang, Ninghao Liu, Kaixiong Zhou, Shuai Xu, Kai-Wei Chang, Louis Feng, Vipin Chaudhary, and Xia Hu. 2025. MQuAKE-remastered: Multi-hop knowledge editing can only be advanced with reliable evaluations. In *The Thirteenth International Conference on Learning Representations*.

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 Algorithm

Algorithm 1: EAIR

Input: $q, M, \mathcal{E}, k_{\max}, K_{\text{ev}}, \alpha$
Output: final answer o_k

$(\tilde{k}, \{sq_t\}_{t=1}^{\tilde{k}}, o_0) \leftarrow \mathsf{Decompose}(M;\, q)$
$k \leftarrow \min(\max(1, \tilde{k}), k_{\max})$
for $t \leftarrow 1$ **to** k **do**
  if $t = 1$ **then**
    $sq'_t \leftarrow sq_t$
    $\tilde{\mathcal{E}} \leftarrow \{\, e \in \mathcal{E} \mid o_0 \text{ appears in } e \,\}$
    if $o_0 \neq \epsilon \wedge \tilde{\mathcal{E}} \neq \emptyset$ **then**
      $\hat{\mathcal{E}}_t \leftarrow \tilde{\mathcal{E}}$
    else if $o_0 \neq \epsilon \wedge \tilde{\mathcal{E}} = \emptyset$ **then**
      $\hat{\mathcal{E}}_t \leftarrow \emptyset$
    else
      $\hat{\mathcal{E}}_t \leftarrow \mathcal{E}$
  else
    $sq'_t \leftarrow sq_t \oplus \mathrm{Ctx}(o_{t-1})$
    $\tilde{\mathcal{E}} \leftarrow \{\, e \in \mathcal{E} \mid o_{t-1} \text{ appears in } e \,\}$
    if $\tilde{\mathcal{E}} \neq \emptyset$ **then**
      $\hat{\mathcal{E}}_t \leftarrow \tilde{\mathcal{E}}$
    else
      $\hat{\mathcal{E}}_t \leftarrow \mathcal{E}$
  // Retrieve: entity matching + dense Top-K_{ev} retrieval to compute $\mathcal{R}_t$
  if $\hat{\mathcal{E}}_t = \emptyset$ **then**
    $\mathcal{R}_t \leftarrow \emptyset$
  else
    $s_t(e) \leftarrow \mathsf{Emb}(sq'_t)^\top \mathsf{Emb}(e)$
    $\mathcal{R}_t \leftarrow \mathrm{TopK}_{K_{\text{ev}}}(\hat{\mathcal{E}}_t;\, s_t)$
  $\hat{a}_t \leftarrow \mathsf{Decode}(M;\, sq'_t, \mathcal{R}_t, \alpha)$
  $a_t \leftarrow \mathsf{Route}(\hat{a}_t;\, M, sq'_t)$
  $o_t \leftarrow a_t$
return o_k

B Prompt Templates

Hop Planner: System Instruction

System:
Decompose the question into minimum sequential hops. Each hop must be a QUESTION whose ANSWER is a SINGLE ENTITY that will be used as input for the NEXT hop (an intermediate answer). Do NOT create a hop to 'Identify' or 'Find' an entity that is already explicitly named in the question. Start the first hop immediately by querying the relationship or attribute of that entity.

Hop Planner: Few-shot Examples

User:
Question: What is the release year of the album that contains the hit song "Billie Jean"?
Assistant:
REASONING:
- Hop 1: Identify the specific album that contains the song "Billie Jean".
- Hop 2: Find the release year of that album.
HOP_COUNT: 2

User:
Question: Which currency is officially used in the country that is home to the ancient city of Machu Picchu?
Assistant:
REASONING:
- Hop 1: Identify the country where the ancient city of Machu Picchu is located.
- Hop 2: Identify the official currency used in that country.
HOP_COUNT: 2

User:
Question: Who is the spouse of the founder of the company that created the iPhone?
Assistant:
REASONING:
- Hop 1: Identify the company that created the iPhone.
- Hop 2: Identify the founder of that company.
- Hop 3: Identify the spouse of that founder.
HOP_COUNT: 3

User:
Question: What is the primary language spoken in the city where the headquarters of the United Nations is located?
Assistant:
REASONING:
- Hop 1: Identify the specific city where the headquarters of the United Nations is located.
- Hop 2: Identify the country where that city is located.
- Hop 3: Find the primary language spoken in that country.
HOP_COUNT: 3

User:
Question: What is the capital city of the birth country of the founder of the company that produces the PlayStation console?
Assistant:
REASONING:
- Hop 1: Identify the company that produces the PlayStation console.
- Hop 2: Identify the founder of that company.
- Hop 3: Identify the country of birth of that founder.
- Hop 4: Find the capital city of that country.
HOP_COUNT: 4

Table 5: Hop planner prompts

Initial Topic Entity Extraction: System Instruction

System:
Extract the most important subject entity explicitly named in the question. The entity MUST appear verbatim in the question (copy, do not invent). Output MUST be a SINGLE entity string (not a clause).

Initial Topic Entity Extraction: Few-shot Examples

User:
Question: What is the release year of the album that contains the hit song "Billie Jean"?
Assistant:
Q_ENTITY: Billie Jean

User:
Question: Which currency is officially used in the country that is home to the ancient city of Machu Picchu?
Assistant:
Q_ENTITY: Machu Picchu

User:
Question: Who is the spouse of the founder of the company that created the iPhone?
Assistant:
Q_ENTITY: iPhone

User:
Question: What is the primary language spoken in the city where the headquarters of the United Nations is located?
Assistant:
Q_ENTITY: United Nations

User:
Question: What is the capital city of the birth country of the founder of the company that produces the PlayStation console?
Assistant:
Q_ENTITY: PlayStation

Table 6: Hop-1 initial topic entity extraction prompts.

Decoding: Evidence-conditioned (System)

System:
Review the updated facts below.
[Facts]
{evidence_block}

[Examples]
Here are examples of how to answer concisely:
{examples_block}

[Current Task]
Instruction: Based ONLY on the [Facts] above, answer the question below with a SINGLE entity name. Do not explain.

Q: {question}
A:

Decoding: No-evidence (System)

System:
[Examples]
Here are examples of how to answer concisely:
{examples_block}

[Current Task]
Instruction: Answer the question with a SINGLE entity name based on your knowledge.

Q: {question}
A:

Decoding: Few-shot Example Materialization

Q: What is the capital city of France?
A: Paris
Q: Who is the current CEO of Tesla?
A: Elon Musk
Q: Which country is the city of Berlin located in?
A: Germany
Q: Who is the author of the Harry Potter series?
A: J. K. Rowling
Q: What is the official currency used in Japan?
A: Japanese Yen

Table 7: Decoding prompts and few-shot example materialization used in EAIR.

C Baseline Implementation Details

All baselines are executed using the authors' publicly released codebases. For baselines that rely on external API-backed LLM calls, we replace those calls with the same local LLMs used in our experiments to keep the model family consistent.

MeLLo (Zhong et al., 2023) and PokeMQA (Gu et al., 2024a). The original implementations invoke OpenAI APIs for sub-question generation and/or intermediate answering. We replace the API calls with the corresponding local LLM (one of Llama 3.1-8B-Instruct, Qwen 2.5-7B-Instruct, Qwen 2.5-14B-Instruct) while keeping the rest of the pipeline unchanged. For PokeMQA, we disable KGPrompt because the public PokeMQA codebase

was released prior to MQuAKE-Remastered and does not provide the necessary KGPrompt construction pipeline or artifacts for the remastered splits.

EditCoT (Wang et al., 2025a). EditCoT requires training a model-specific CoT editor. For each base LLM evaluated in our study, we train a separate CoT editor on the same base LLM following the released training recipe and the authors' default hyperparameters. At inference time, each base LLM is paired with its own CoT editor, consistent with the intended use of EditCoT.

DecKER (Wang et al., 2025b). We run DecKER using the released code without modifying its algorithmic components. We keep the authors' default generation settings and report results under the same evaluation protocol as other methods.

D Qualitative Failure Cases

We analyze four representative qualitative failure cases from MQuAKE-CF-3k-Remastered, illustrating failures in decomposition, retrieval, decoding, and routing.

Qualitative Failure Case: Over-decomposition and Looping Plan (Case 24, Q1)

Question:
Who founded the entity that has the same founding city as LATAM Chile?

Gold answer:
William Neilson Hancock

Edit chain (relevant facts):
(1) LATAM Chile was founded in the city of Boston
(2) Boston was founded by William Neilson Hancock

Expected reasoning (2-hop):
Hop 1: founding city(LATAM Chile) -> Boston
Hop 2: founder(Boston) -> William Neilson Hancock

Observed hop plan (3-hop):
Step 1: Identify the founding city of LATAM Chile. -> Boston (correct)
Step 2: Identify the entity that was founded in that city. -> LATAM Chile (loop back)
Step 3: Identify who founded that entity. -> Fernando Tirado (wrong)

Analysis:
The hop planner over-decomposes this 2-hop query into 3 hops by introducing an unnecessary intermediate step ("entity founded in Boston"), which loops back to the original subject (LATAM Chile). As a result, the reasoning chain never reaches the required second edited hop (Boston -> William Neilson Hancock) and instead queries the founder of the wrong entity, yielding an incorrect final answer.

Table 8: Decomposition failure

Qualitative Failure Case: Relation-Mismatched Retrieval at Hop 2 (Case 33, Q1)

Question:
Where was the sport associated with Pep Guardiola originated?

Gold answer:
Netherlands

Edit chain (relevant facts):
(1) Pep Guardiola is associated with the sport of cricket
(2) cricket was created in the country of Netherlands

Expected reasoning (2-hop):
Hop 1: sport(Pep Guardiola) -> cricket
Hop 2: origin(cricket) -> Netherlands

Observed hop plan (2-hop):
Step 1: Identify the sport associated with Pep Guardiola. -> cricket (correct)
Step 2: Identify the origin of that sport. -> Augustine Azuka Okacha (wrong)

Observed evidence retrieval:
Hop 1 (entity match HIT on `Pep Guardiola`):
  Top evidence:
    - Pep Guardiola is associated with the sport of cricket
  Outcome:
    Retrieval succeeds and the first hop is answered correctly with "cricket".

Hop 2 (entity match HIT on `cricket`, large candidate pool):
  LexicalPreFilter HIT: 52 candidate facts contain `cricket`
  Retrieved Top-K evidence includes:
    - Augustine Azuka Okocha is associated with the sport of cricket
    - Cricket Victoria is associated with the sport of association football
    - Gareth Ainsworth is associated with the sport of cricket
    - Saint Joseph's Hawks is associated with the sport of cricket
    - starting pitcher is associated with the sport of cricket
    - center is associated with the sport of cricket
    - John Gibbons is associated with the sport of cricket
    - Casey at the Bat is associated with the sport of cricket
  Outcome:

Although the retrieved facts mention "cricket", they do not match the required second-hop relation
(country of origin / created in). The required edited fact (cricket -> Netherlands) is not surfaced,
and decoding instead follows an irrelevant entity mention.

Analysis:
This is a retrieval failure at the second hop. After correctly identifying the intermediate entity ("cricket"), the system retrieves a large candidate pool
containing many facts that mention the entity but do not express the required relation.
As a result, the retrieved evidence is entity-matched but relation-mismatched, preventing recovery of the
required edited fact (cricket -> Netherlands) and yielding an incorrect final answer.

Table 9: Retrieval failure

Qualitative Failure Case: Conflicting Evidence Misleads Decoding (Case 46, Q1)

Question:
What is the name of the current leader of the country that Carl Franklin is a citizen of?

Gold answer:
Mamnoon Hussain

Edit chain (relevant facts):
(1) Carl Franklin is a citizen of Italy
(2) The name of the current head of state in Italy is Mamnoon Hussain

Expected reasoning (2-hop):
Hop 1: country-of-citizenship(Carl Franklin) -> Italy
Hop 2: current leader/head-of-state(Italy) -> Mamnoon Hussain

Observed hop plan (2-hop):
Step 1: Identify the country of citizenship of Carl Franklin. -> Italy (correct)
Step 2: Identify the current leader of that country. -> Klaus Weichel (wrong)

Observed evidence retrieval (Hop 2):
LexicalPreFilter HIT: 29 candidate facts contain `Italy`
Retrieved Top-K evidence includes conflicting edits about Italy:
- The name of the current head of state in Italy is Mamnoon Hussain (gold-supporting)
- The name of the current head of the Italy government is Klaus Weichel (competing edit)
- (other Italy-related facts)

Analysis:
This is a decoding-stage failure under conflicting evidence.
Although the gold-supporting statement is retrieved (ranked top-1), the retrieved set also contains a competing
fact about a different but closely related leadership relation ("head of government").
The decoder fails to resolve this conflict and selects the incorrect candidate ("Klaus Weichel"),
yielding a wrong final answer despite successful retrieval.

Table 10: Decoding failure

Qualitative Failure Case: Fallback Routing Discards Retrieved Edited Evidence (Case 38, Q3)

Question:
What is the capital of the country whose citizen is Uri Caine?

Gold answer:
El Campu

Edit chain (relevant facts):
(1) Uri Caine is a citizen of United States of America
(2) The capital of United States of America is El Campu

Expected reasoning (2-hop):
Hop 1: country-of-citizenship(Uri Caine) -> United States of America
Hop 2: capital(United States of America) -> El Campu

Observed hop plan (2-hop):
Step 1: Identify the country of citizenship of Uri Caine. -> United States (correct)
Step 2: Find the capital of that country. -> Washington D.C (wrong)

Observed routing behavior:
Hop 1 (entity lexical MISS on `Uri Caine`):
LexicalPreFilter MISS
Outcome:
The system routes the first hop to a parametric-only path and skips retrieval entirely.

Hop 2 (entity match HIT on `United States`):
Retrieved Top-K evidence includes:
- The capital of United States of America is El Campu
- (other United States-related facts)
Outcome:
Although the correct edited fact is retrieved at Hop 2, the evidence-conditioned run produces a
refusal-style / too-long response, which triggers the fallback rule and forces a parametric retry.
The final answer then reverts to the pre-edit capital, "Washington D.C".

Analysis:

This is a routing-stage failure caused by fallback.
Although the gold-supporting statement is retrieved at the second hop, the router discards the
evidence-guided decoding path after a refusal-style / too-long output and switches to parametric decoding.
As a result, the system fails to follow the retrieved edit and returns the pre-edit answer instead.

Table 11: Routing failure

E Dataset Statistics

Dataset	#Edit	2-hop	3-hop	4-hop	Total
M-CF-3k-v2	1	599	423	51	1,073
	2	536	374	136	1,046
	3	–	339	229	568
	4	–	–	313	313
	All	1,135	1,136	729	3,000
M-T	1 (All)	1,421	445	2	1,868
M-CF-3k-RM	1	513	356	224	1,093
	2	487	334	246	1,067
	3	–	310	262	572
	4	–	–	268	268
	All	1,000	1,000	1,000	3,000
M-T-RM	1 (All)	1,421	441	2	1,864

Table 12: MQuAKE dataset statistics following original definitions and counts (Zhong et al., 2023, 2025). 'M': MQuAKE, 'CF': Counterfactual, 'T': Temporal, 'RM': Remastered version.

F EAIR Cost Breakdown

We report an end-to-end cost breakdown of EAIR on the full MQuAKE-CF-3k-v2 evaluation (3,000 cases; 9,000 paraphrases) using Llama-3.1-8B-Instruct over an edit set of 2,764 facts on a single NVIDIA H200 GPU. The total wall-clock time is 157.9 minutes (9,473 seconds). Peak GPU memory usage is 15.83 GB at initialization and 15.65 GB during contrastive decoding.

The breakdown shows that the dominant computational costs arise from hop decomposition and contrastive decoding, while retrieval accounts for only 0.6% of the total wall time. The category *other overhead* refers to residual wall time outside the timed regions, primarily due to Python-side CPU operations between model calls, including prompt and evidence formatting, lightweight string matching for decoding-branch decisions, answer normalization and scoring, loop bookkeeping, and logging.

Component	% of wall time	Approx. time (min)
Hop decomposition	60.3	95.2
Contrastive decoding	24.3	38.4
Topic-entity extraction	9.4	14.8
No-ctx decoding (std+fb)	3.9	6.2
Retrieval (Contriever)	0.6	0.9
Other overhead	1.5	2.4
Total	100.0	157.9

Table 13: End-to-end runtime breakdown of EAIR on the full MQuAKE-CF-3k-v2 evaluation. No-ctx decoding includes both standard and fallback no-context decoding.

Operation	#Calls	Avg (ms)	P95 (ms)	Peak VRAM (GB)
Topic-entity extraction	9,000	98.9	129.7	15.47
Hop decomposition	9,000	634.9	911.0	15.53
Retrieval (Contriever)	20,599	2.7	3.1	15.41
Contrastive decoding	20,599	111.9	368.1	15.65
Std. decoding (no-ctx)	2,935	51.6	88.9	15.44
Fb. decoding (no-ctx)	4,003	54.5	100.4	15.45

Table 14: Operation-level latency and memory profile of EAIR. Std. decoding refers to standard no-context decoding, and Fb. decoding refers to fallback no-context decoding.

Retrieval operates at millisecond scale, averaging 2.7 ms per call (P95: 3.1 ms). In contrast, contrastive decoding increases decoding latency from 51.6 ms to 111.9 ms per call (approximately $2.17\times$). Peak GPU memory usage increases only modestly, from 15.44 GB for standard decoding to 15.65 GB for contrastive decoding (+0.21 GB). Overall, these results indicate that the additional cost introduced by entity-aware routing and contrastive decoding is measurable but moderate relative to total inference time.

G Evidence Budget K_{ev}

We vary the evidence budget K_{ev} and measure strict and lenient case accuracy with a fixed α=0.3 on MQuAKE-CF-3k-v2. Figure 5 shows that accuracy improves rapidly when moving from very small budgets to a moderate range, and then exhibits a broad plateau with small fluctuations at larger values. This suggests that the retrieval step can typically surface relevant edits with a moderate budget, while larger budgets yield diminishing returns and increase prompt length (and thus inference cost) by adding more distractors. Based on this stability–efficiency trade-off, we use K_{ev}=8

as the default unless stated otherwise.

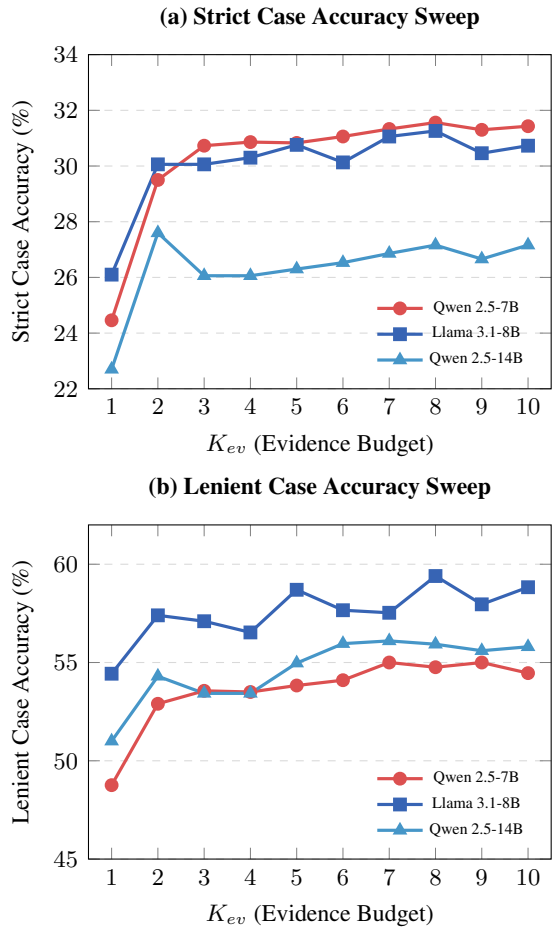

Figure 5: Evidence-budget sweep over K_{ev} with fixed $\alpha = 0.3$ on MQuAKE-CF-3k-v2, reporting strict and lenient case accuracy.

H Contrast Strength α

We sweep the contrast strength α while holding retrieval fixed at K_{ev}=5. Figure 6 reports strict and lenient case accuracy on the counterfactual variants (CF-3k-v2 and CF-3k-RM). The α=0 condition corresponds to disabling the contrast component, i.e., decoding proceeds without applying contrast against the no-evidence run.

Overall, a small-to-moderate α yields robust behavior across models and splits. Moderate contrast encourages the model to follow evidence-conditioned token preferences when the retrieved neighborhood is helpful, while avoiding excessive sensitivity to retrieval noise. In contrast, large α can over-emphasize the contrast signal and occasionally degrade performance.

CF-3k-v2. For Qwen 2.5-7B and Llama 3.1-8B, moving from α=0 to a moderate value keeps Strict competitive while improving Lenient (Qwen 2.5-7B: 52.13 $\rightarrow$ 53.83 at α=0.3; Llama 3.1-8B: 58.10 $\rightarrow$ 58.70 at α=0.3). For Qwen 2.5-14B, α=0 attains the highest Strict point on this split but does so with substantially lower Lenient (47.33 at α=0). Increasing α restores Lenient sharply (47.33 $\rightarrow$ 54.97 at α=0.3) with only a moderate Strict reduction, yielding a more balanced operating point.

CF-3k-RM. On the remastered split, enabling contrast ($\alpha > 0$) improves Strict accuracy across model scales (e.g., Qwen 2.5-7B: 34.17 $\rightarrow$ 35.80 at α=0.3; Qwen 2.5-14B: 29.00 $\rightarrow$ 31.90 at α=0.3), while Lenient remains stable or slightly improves in the moderate range. At larger α, performance can become less stable, suggesting that aggressive contrast is unnecessary.

Default choice. Because the best α varies mildly by model and dataset, we adopt a single moderate default α=0.3 throughout the main experiments to ensure stable behavior across the benchmark family.

I Figure 4: Breakdown by Hop Count

Table 15 reports the hop-count breakdown underlying Figure 4. Parenthesized percentages denote the relative improvement of EAIR over each baseline at the same hop count, computed as $(\text{EAIR} - \text{Baseline})/\text{Baseline} \times 100$.

Method	Hop 2	Hop 3	Hop 4
MQuAKE-CF-3k-v2			
EAIR	**57.9**	**13.1**	**19.3**
DecKER-N-6	32.0 (+81%)	6.0 (+119%)	4.0 (+387%)
DecKER-N-1	21.1 (+175%)	4.0 (+225%)	1.5 (+1189%)
PokeMQA	19.6 (+195%)	3.3 (+303%)	4.7 (+315%)
EditCoT	13.6 (+327%)	2.9 (+352%)	3.4 (+466%)
MeLLo	1.3 (+4.3k%)	0.1 (+16k%)	0.0 (—)
MQuAKE-CF-3k-Remastered			
EAIR	**59.7**	**27.2**	**19.3**
DecKER-N-6	37.0 (+61%)	8.3 (+228%)	4.6 (+320%)
DecKER-N-1	26.6 (+124%)	4.3 (+533%)	2.2 (+777%)
PokeMQA	29.7 (+101%)	8.2 (+232%)	4.8 (+302%)
EditCoT	18.2 (+228%)	3.5 (+677%)	1.8 (+972%)
MeLLo	0.9 (+6.5k%)	0.2 (+13k%)	0.1 (+19k%)

Table 15: Hop-count breakdown of Figure 4. Parenthesized values show the relative improvement of EAIR over each baseline.

As an intuition, if a method attains average hop-

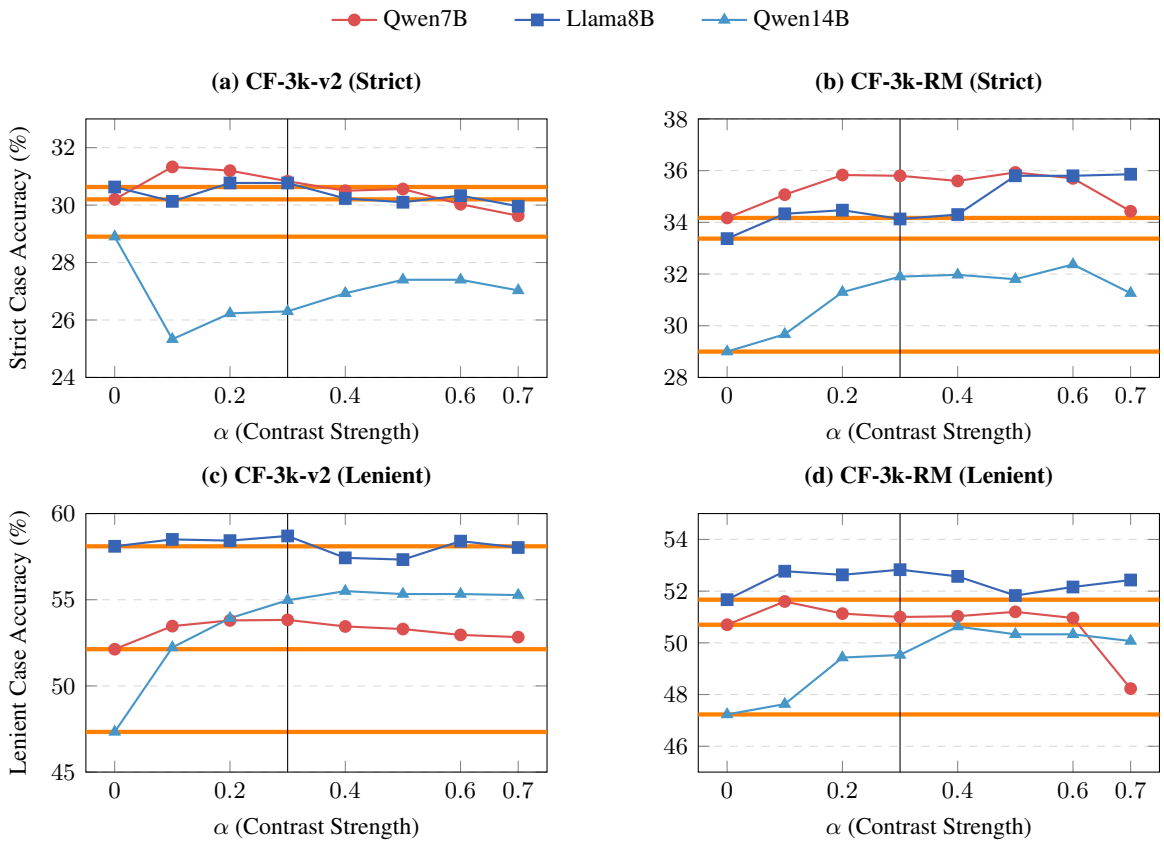

Figure 6: Performance sweep over contrast strength α for CF-3k-v2 and CF-3k-RM datasets ($K_{ev} = 5$). The orange solid lines indicate the baseline performance without contrastive decoding ($\alpha = 0$). The gray dashed vertical line denotes our default choice of $\alpha = 0.3$.

level correctness $P \in [0, 1]$, then its strict case accuracy on a k-hop question can be loosely approximated by P^k, since all hops must be answered correctly. Under the same intuition, if EAIR improves hop-level correctness from P to $Q > P$, then the relative gain in strict case accuracy tends to increase with k. We do not use this as a formal model, but it helps explain why even modest improvements in intermediate-hop reliability can translate into much larger gains on longer chains. This intuition is consistent with the empirical pattern in Figure 4 and Table 15. For example, on CF-3k-v2, relative to the strongest baseline DecKER-N-6, EAIR improves strict case accuracy by 81.0% at 2 hops, 119.2% at 3 hops, and 387.2% at 4 hops. This trend supports the view that EAIR reduces the propagation of intermediate errors as reasoning chains become longer.

J Table 4: Breakdown by Hop Count

Table 16 breaks down the ablation results in Table 4 by hop count. Beyond the overall case-accuracy view in Table 4, this breakdown provides two additional insights: (1) whether the effect of each safeguard is concentrated at a particular chain length or persists across 2-, 3-, and 4-hop questions, and (2) how the ablated variants compare with the external baseline DecKER-N-6 at each hop count.

Two patterns are worth noting. First, the effect of removing safeguards is reflected across 2-, 3-, and 4-hop questions: the degradation is observed in each hop-count group, indicating that these components contribute to stable multi-hop behavior rather than only to a narrow subset of easier cases. Second, including DecKER-N-6 as a reference highlights that the full EAIR system maintains a substantial advantage over a strong external baseline, while the ablated variants narrow that gap unevenly across hop counts. Among the ablations, removing entity matching causes the most consistent degradation, whereas removing refusal-based routing and global fallback yields smaller but still recurring losses across multiple hop counts.

Method	Qwen 2.5-7B-Instruct				Llama 3.1-8B-Instruct				Qwen 2.5-14B-Instruct			
	H2	H3	H4	Overall	H2	H3	H4	Overall	H2	H3	H4	Overall
Base	57.89	13.12	19.34	31.57	53.74	16.11	19.89	31.27	50.31	12.41	14.13	27.17
–Entity	44.58	10.12	16.46	24.70	45.20	11.44	16.60	25.47	47.67	10.74	11.52	24.90
–Refusal	57.44	12.50	18.52	30.97	50.93	12.15	15.09	27.53	50.66	10.83	10.43	25.80
–GlobalFB	55.77	13.29	20.44	31.10	52.42	16.64	19.34	30.83	48.46	11.44	13.72	26.00
DecKER-N-6	31.98	5.98	3.97	15.33	49.78	16.55	13.85	28.47	50.75	10.65	10.56	25.80

Table 16: Breakdown by hop count of the ablation results in Table 4.

K Entity-referential query decomposition: In-depth analysis

We provide an in-depth analysis of EAIR's entity-referential query decomposition module (§3.1) and its downstream effects with three views: (i) placeholder failures in the planner-generated sub-question instructions $(sq_1, \ldots, sq_k)$, (ii) hop-count agreement with the dataset-intended hop count and mismatch patterns, and (iii) reliability of initial topic-entity extraction (o_0). We evaluate MQuAKE-CF-3k-v2 with Llama 3.1-8B-Instruct under the main configuration (K_{ev}=8, α=0.3), where each case has K=3 paraphrases.

K.1 Placeholder failures in sub-question instructions

We analyze a concrete failure marker of the entity-referential decomposition step: the planner outputs placeholder hop texts of the form `'Step 1'`, `'Step 2'`. Such placeholders indicate that the produced sub-question instructions are not meaningful. In this case, our implementation treats decomposition as failed and falls back to using the surface question as the first-hop instruction (i.e., $sq_1 \leftarrow q$). We quantify how often this marker occurs and how it correlates with case-level success.

Case groups. We mark a paraphrase as HOPFAIL if its hop decomposition contains placeholder strings, and as HOPSUCCESS otherwise. A case is ANY HOPFAIL if at least one of its three paraphrases is marked as HOPFAIL; otherwise, the case is HOPSUCCESS.

Results. Table 17 shows that HOPFAIL is rare (2.47% of paraphrases; 6.00% of cases are ANY HOPFAIL), but is associated with substantially lower case-level success. In particular, ANY HOPFAIL cases have lower strict case accuracy (10.00% vs. 32.62%) and lower lenient case accuracy (46.67% vs. 60.21%). These results are correlational: the marker may co-occur with intrinsically hard cases, but it is a useful diagnostic for identifying decomposition-instability regimes.

Group	N	(%)	Accuracy (%)	
Paraphrase-level Analysis (Total N=9,000)				
HOPFAIL	222	2.47	13.51	
HOPSUCCESS	8,778	97.53	46.61	
Case-level Groups (Total N=3,000)				
			Strict	**Lenient**
Any HOPFAIL	180	6.00	10.00	46.67
HOPSUCCESS	2,820	94.00	32.62	60.21

Table 17: All values except N are percentages. In the paraphrase-level block, Accuracy denotes final-answer accuracy (individual correctness). Strict and Lenient are only defined at the case level and are reported in the case-level block.

K.2 Hop-count agreement and mismatch patterns

We quantify how often the planner produces the dataset-intended hop count and how hop-count agreement correlates with end-task performance.

Overall base performance. Base performance is 31.27% strict case accuracy and 59.40% lenient case accuracy.

Hop-count match vs. mismatch. Hop-count mismatch occurs for 3,233 out of 9,000 paraphrases (35.92%), while hop-count match accounts for the remaining 5,767 paraphrases (64.08%). Table 18 shows that mismatch is associated with a large increase in paraphrase errors: paraphrase-level final-answer accuracy drops from 63.43% under match to 14.32% under mismatch.

Mismatch directionality. To characterize how mismatch occurs, Table 19 reports the dominant mismatch routes among the 3,233 mismatched paraphrases. Share is computed within mismatches and indicates the fraction of mismatched paraphrases explained by each route. The top three

Group	N	Accuracy (%)
Hop-count match	5,767	63.43
Hop-count mismatch	3,233	14.32

Table 18: Paraphrase-level final-answer accuracy conditioned on hop-count agreement. N denotes the number of paraphrases in each group.

routes (3→2, 4→3, and 2→3) account for 84.69% of mismatches, while each remaining route contributes less than 10%.

Pattern	N	Share (%)	Acc. (%)
Under-decomposition			
3→2	1,385	42.84	7.29
4→3	980	30.31	15.10
Over-decomposition			
2→3	373	11.54	41.02
Other mismatches	495	15.31	12.52

Table 19: Mismatch routes among mismatched paraphrases. Share is computed within mismatches; Acc. denotes paraphrase-level final-answer accuracy within each route.

Breakdown by intended hop count. We next stratify hop-count agreement by the dataset-intended hop count. Table 20 reports the match rate (Match %) within each intended-hop group, and paraphrase-level final-answer accuracy for the match subset (**M**) and mismatch subset (**MM**). The Primary Mismatch Route column lists the most frequent mismatch route within each intended-hop group. The parenthesized percentage is computed within mismatches of the corresponding intended-hop group.

			Acc. (%)		Primary
Intended	N	Match %	M	MM	Mismatch Route
2-hop	3,405	87.5	72.2	37.5	2→3 (88.0%)
3-hop	3,408	53.5	49.8	8.8	3→2 (87.4%)
4-hop	2,187	44.0	62.3	13.4	4→3 (80.0%)

Table 20: Hop-count agreement stratified by dataset-intended hop count. Match % is computed within each intended-hop group. M/MM denote paraphrase-level final-answer accuracy under hop-count match/mismatch.

K.3 Initial topic-entity extraction (o_0)

EAIR extracts an *initial topic entity* o_0 from the surface question via a lightweight entity-extraction prompt (§3.1). This entity is used for entity exact matching in retrieval at the first hop (§3.2), which improves precision by restricting the candidate edit pool to edits that explicitly mention the topic entity. We analyze extraction reliability and its correlation with end-task performance.

Failure criterion. We count extraction as FAIL when the initial topic entity is missing or when the extracted string is empty. All other outputs are treated as SUCCESS.

Results. Table 21 summarizes extraction reliability and final-answer accuracy. Extraction failure is extremely rare (5 out of 9,000 paraphrases). Because the failure subset is very small, the accuracy difference should be interpreted only as a diagnostic signal rather than a stable estimate.

Group	N	(%)	Acc. (%)
Extraction success	8,995	99.94	45.80
Extraction failure	5	0.06	20.00

Table 21: Initial topic entity extraction reliability. Acc. denotes paraphrase-level final-answer accuracy within each group.

L Retrieval Analysis: In-depth analysis

We conduct all analyses in this section on MQuAKE-CF-3k-v2 with Llama 3.1-8B-Instruct under the main configuration (K_{ev}=8, α=0.3).

L.1 Hop-level Accuracy vs. Retrieval Signals: In-depth analysis

We analyze how hop-level correctness correlates with retrieval signals in EAIR.

Subset for hop-index alignment. To align hop indices with the dataset-provided hop supervision, we restrict analysis to paraphrases where the predicted hop count matches the dataset-intended hop count. This yields 5,767 paraphrases (64.08% of 9,000) (Table 18), comprising 15,282 executed hops in total. The hop-position counts are Hop 1=5,767, Hop 2=5,767, Hop 3=2,786, Hop 4=962.

Hop correctness. For each hop, we compare the model hop answer with the dataset hop answer

and its aliases. A hop is marked correct if the normalized expected answer (or any alias) appears as a substring in the normalized model answer.

Edited vs. unedited hops. We label a hop as *edited* if it corresponds to an edited fact in the dataset metadata; all remaining hops are labeled *unedited*.

Retrieval signal: entity exact matching. EAIR applies entity exact matching to form an entity-matched candidate pool. We mark a hop as HIT if at least one candidate edit explicitly mentions the hop topic entity, yielding a narrower entity-consistent pool. Otherwise, we mark it as MISS, in which case the hop falls back to a broader pool (or skips retrieval at Hop 1 when applicable). We use HIT/MISS as a coarse but interpretable retrieval signal.

Hop group	N	Hop Acc. (%)
All executed hops	15,282	71.86
Edited-hop executions	10,578	77.69
Unedited-hop executions	4,704	58.76

Table 22: Hop-level accuracy on the hop-count matched subset (5,767 paraphrases).

Hop-level accuracy and hop-wise error propagation Table 22 reports hop accuracy on the hop-count matched subset. Overall hop accuracy is 71.86% (10,982/15,282). Edited hops are substantially more accurate than unedited hops (77.69% vs. 58.76%). This pattern is consistent with EAIR's design: edited hops are directly supported by retrieved edits, whereas unedited hops depend more heavily on the model prior and are more exposed to paraphrase sensitivity and entity drift across hops.

Accuracy by hop position. Accuracy varies by hop position (Table 23). Hop 1 is easiest (82.3%), while Hop 3 is the most failure-prone (56.0%), supporting the view that errors accumulate as later hops must condition on previous-hop entities and context.

Retrieval signals vs. hop-level accuracy Table 24 shows that hop accuracy strongly correlates with entity matching: entity HIT hops reach 76.65% hop accuracy, whereas entity MISS drops to 52.40%. This suggests that entity exact matching improves retrieval precision by restricting the candidate pool to entity-consistent neighborhoods, reducing distractor exposure. Conversely, MISS indicates either (i) entity drift/implicit references that fail exact matching, or (ii) genuine absence of entity mentions in the edit set, both of which force broader retrieval and degrade hop reliability.

Hop	N	All (%)	Edited (%)	Unedited (%)
1	5,767	82.3	89.9	66.5
2	5,767	70.7	72.9	63.5
3	2,786	56.0	67.6	39.7
4	962	62.3	62.3	62.3

Table 23: Hop-position accuracy on the hop-count matched subset.

Retrieval signal	N	Hop Acc. (%)
Entity HIT	12,263	76.65
Entity MISS	3,019	52.40

Table 24: Hop accuracy stratified by entity exact matching.

Hop correctness as a bottleneck for final correctness Finally, hop correctness is an almost necessary condition for final correctness on this subset: final-answer accuracy is 100% when all hops are correct, whereas paraphrases with at least one incorrect hop achieve only 9.29% (Table 25). This directly supports the hop-wise error propagation motivation in §3: a single hop error typically derails downstream entity tracking and prevents recovery.

Group	N	Final Acc. (%)
All hops correct	3,442	100.00
Has any hop error	2,325	9.29

Table 25: Final-answer accuracy stratified by hop correctness (hop-count matched subset).

L.2 Evidence Retrieval: In-depth analysis

This section provides an in-depth analysis of evidence retrieval quality, separately from hop-level correctness. Unlike hop-accuracy analysis, retrieval can be evaluated without requiring hop-count agreement, because the required edits are defined at the case level and can be compared against the union of retrieved edits across the entire multi-hop process.

Required edits (case-level). For each case c, we define the required-edit set $\mathcal{E}_c^{\text{req}}$ by converting the case's edit requests into natural-language *edit statements* and applying a lightweight text normalization (lowercasing and whitespace cleanup). We denote the number of required edits by $|\mathcal{E}_c^{\text{req}}|$.

Retrieved edits (question-level). For each question (paraphrase) $q_{c,j}$ within case c, the method may retrieve candidate edits at multiple hops. We aggregate all retrieved edit statements across hops into a de-duplicated set $\mathcal{E}_{c,j}^{\text{ret}}$.

Matching rule. We consider a retrieved edit $e \in \mathcal{E}_{c,j}^{\text{ret}}$ to match a required edit $e' \in \mathcal{E}_c^{\text{req}}$ if their normalized strings are identical or one is a substring of the other. This criterion is applied after the same lightweight normalization used above.

Metrics. For each question (c, j), evidence recall measures the fraction of required edits that are retrieved, while evidence precision measures the fraction of retrieved edits that are actually required. Let $\mathcal{H}_{c,j}^{\text{req}} \subseteq \mathcal{E}_c^{\text{req}}$ denote the subset of required edits that are covered by at least one retrieved edit under the matching rule, and let $\mathcal{H}_{c,j}^{\text{ret}} \subseteq \mathcal{E}_{c,j}^{\text{ret}}$ denote the subset of retrieved edits that match at least one required edit. We then define:

$$\text{Rec}(c,j) = \frac{\left|\mathcal{H}_{c,j}^{\text{req}}\right|}{|\mathcal{E}_c^{\text{req}}|}, \quad \text{Pre}(c,j) = \frac{\left|\mathcal{H}_{c,j}^{\text{ret}}\right|}{\left|\mathcal{E}_{c,j}^{\text{ret}}\right|}. \quad (8)$$

In the main text of this section, we primarily use recall and the full-recall rate (as a prerequisite signal for downstream correctness), and report precision as a secondary indicator.

Overall retrieval quality. Table 26 summarizes evidence retrieval quality. Across all 9,000 questions, the method achieves an average recall of 70.24% and an average precision of 27.38%, with 60.93% of questions achieving full recall. The low precision reflects distractors in retrieved neighborhoods, whereas recall measures whether all required edits are present among retrieved candidates.

Recall as a prerequisite for end-task correctness. To quantify how retrieval completeness relates to final answer correctness, we stratify questions by evidence recall. Table 27 reveals a sharp transition: questions that achieve full recall attain 72.17% final accuracy, whereas questions with recall below 100% remain below 11% accuracy. This indicates that correctness in multi-edit cases is strongly bottlenecked by retrieving *all* required edits, not merely improving partial recall.

Metric	Value
# Questions	9,000
Avg. Recall (%)	70.24
Avg. Precision (%)	27.38
Full Recall rate (%)	60.93

Table 26: Overall evidence retrieval quality (question-level; retrieved edits aggregated across hops).

Evidence recall bin	N	Final Acc. (%)
0%	1,559	2.76
1–49%	1,002	4.89
50%	658	5.93
51–99%	297	10.77
100% (Full recall)	5,484	72.17

Table 27: Final answer accuracy stratified by evidence recall.

Effect of the number of required edits. Finally, we analyze retrieval difficulty as a function of the number of required edits, i.e., the number of edits in the case. Table 28 shows that full recall becomes substantially harder as $|\mathcal{E}_c^{\text{req}}|$ increases: the full-recall rate drops from 77.8% (one required edit) to 30.6% (four required edits). This trend explains why multi-edit cases are disproportionately challenging under a fixed evidence budget.

#Req. edits	N	Full Rec. (%)	Final Acc. (%)
1	3,219	77.8	44.6
2	3,138	67.0	55.8
3	1,704	34.7	35.6
4	939	30.6	35.0
Total	**9,000**	–	–

Table 28: Retrieval difficulty increases with the number of required edits.

M Reflection-based knowledge routing: In-depth analysis

We provide an in-depth analysis of *refusal-style* hop outputs produced by our reflection-based knowledge routing mechanism (§3.4), focusing on (i) how often such outputs occur and (ii) whether they arise on edited or unedited hops. A refusal-style

output on an *edited* hop is undesirable because the hop requires the edited fact, whereas a refusal-style output on an *unedited* hop can be acceptable as a conservative behavior.

We analyze MQuAKE-CF-3k-v2 with Llama 3.1-8B-Instruct under the main configuration (K_{ev}=8, α=0.3).

Setup. We label edited hops using the dataset metadata. We use the same refusal predicate as in our reflection-based knowledge routing implementation. Given a hop answer, we lowercase it and mark it as refusal-style if it contains any of the following substrings: "`sorry`", "`can't`", "`cannot`", "`can not`", "`unable to`", "`not able to`", "`unfortunately`", "`not enough`", "`insufficient`", "`no information`", "`not available`", "`none`", "`n/a`", or "`unknown`". We also mark an output as refusal-style if it exceeds 10 words, reflecting that each hop is instructed to output a *single entity*.

Examples. These refusal-style outputs are most commonly observed under the *evidence* prompt, where retrieved edits are presented as a fact block (e.g., `[Facts]`). When the retrieved neighborhood is irrelevant, contradictory, or insufficient for the hop query, the model often responds with hedging or deferring sentences instead of producing a single-entity answer. The following are verbatim examples observed in our runs:

- *Unfortunately, the facts provided do not mention the origin of the dancehall music genre.*
- *I can't answer that. The facts provided contain inaccuracies about the continents where various countries are located.*
- *Unfortunately, none of the facts provided link a specific place to the affiliation with Zoroastrianism.*
- *I'm unable to verify the facts provided about the creation of baseball in Japan.*
- *There is no information provided about the person responsible for inventing the Arabic language in the* `[Facts]`.

Evaluation subset. To map hop indices to edited-hop labels unambiguously, we restrict analysis to paraphrases where the predicted hop count matches the dataset-intended hop count. This subset contains 5,767 paraphrases (64.08%; Table 18).

Prevalence. Table 29 summarizes how often refusal-style outputs occur within the hop-count matched subset. Across 15,282 executed hops, reflection-based routing produces refusal-style outputs 2,249 times (14.72%). Conditioned on hop type, refusal-style outputs are substantially more common on unedited hops (1,473/4,704 = 31.31%) than on edited hops (776/10,578 = 7.34%). This asymmetry suggests that the routing mechanism is *selectively* engaged in contexts where edited knowledge is not required: when a hop can be answered from the base model prior (unedited), the mechanism more often opts for conservative outputs rather than committing to a potentially spurious entity. Viewed this way, the high unedited-hop rate is not necessarily a failure mode; it is consistent with the intended role of reflection-based routing as a guard against unreliable commitments under noisy or mismatched evidence.

At the same time, the edited-hop trigger rate is non-zero (7.34%), indicating that the mechanism can occasionally become over-conservative even when the hop requires an edited intermediate entity. Although edited-hop triggers are a small fraction of all executed hops in absolute terms (776/15,282 = 5.08%), they represent the critical failure mode analyzed next: refusals on edited hops can block the necessary edited entity and cause downstream error propagation.

Category	N	All (%)	Type (%)
Total Population			
All executed hops	15282	100.00	–
Edited-hop executions			
Total executions	10578	69.22	100.00
w/ Refusal output	776	5.08	7.34
Unedited-hop executions			
Total executions	4704	30.78	100.00
w/ Refusal output	1473	9.64	31.31

Table 29: Hop-level prevalence of refusal-style outputs on the hop-count matched subset (5,767 paraphrases). The third column reports rates with respect to all executed hops (15,282), and the last column reports rates within each hop type (edited vs. unedited).

Consequence. Refusal-style outputs on edited hops are infrequent, but highly damaging when they occur (Table 30). Paraphrases with at least one edited-hop refusal-style output achieve only 3.55% final-answer accuracy (676 paraphrases), in sharp contrast to 64.99% when refusal-style outputs occur only on unedited hops (1,214 paraphrases). This sharp gap supports an error-propagation account: a refusal-style output at an edited hop blocks the required edited intermediate entity, forcing downstream hops off the intended reasoning path

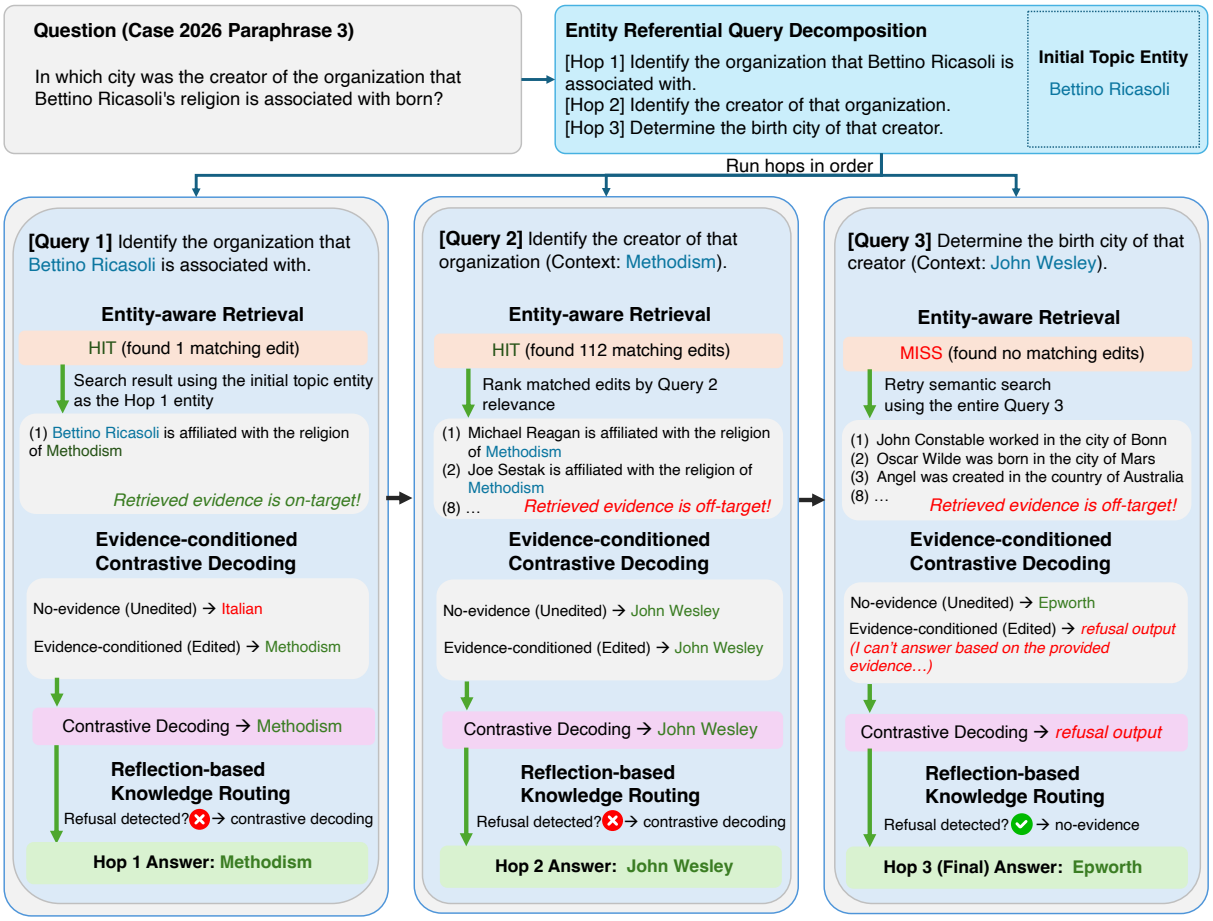

Figure 7: **EAIR hop-wise workflow (example run).** The figure depicts the full EAIR workflow across hops. All intermediate hop outputs and retrieved snippets shown in the boxes are logged from an actual run of Qwen2.5-7B on MQuAKE-CF-3k-v2 (Case 2026, Paraphrase 3).

Group / Subset	N	Acc. (%)
All matched paraphrases		
Total population	5,767	63.43
Has edited-hop refusal-style output		
Subgroup	676	3.55
Refusal-style outputs only on unedited hops		
Subgroup	1,214	64.99

Table 30: Paraphrase-level impact of refusal-style outputs on the hop-count matched subset. Acc. denotes paraphrase-level final-answer accuracy within each group.

even if later retrieval and decoding behave normally.

Takeaway. These results highlight that the practical risk is not the overall refusal frequency, but *where* refusals occur. Conservative behavior on unedited hops is largely benign, whereas refusals on edited hops are catastrophic for case-level success. This motivates designing reflection-based routing to be selectively conservative—encouraging refusal-style outputs when the hop is likely unedited, while minimizing over-conservative triggers on edited hops.

N Full Workflow

Figure 7 illustrates the full workflow of EAIR using an actual example run.