Self-Correcting Agentic RAG via Memory-Grounded Failure Localization

Haoyang Zhong[1] **Yifei Sun**[2*] **Shiwei Liu**[1]
Jie Li[2] **Chunping Wang**[3] **Yang Yang**[1]
[1]Zhejiang University [2]School of Software Technology, Zhejiang University
[3]FinVolution Group
haoyang011@e.ntu.edu.sg wangchunping02@xinye.com
{yifeisun,shiweiliu,jie.li,yangya}@zju.edu.cn

Abstract

Agentic RAG, which decomposes a complex question into dependency-aware sub-questions, has become the dominant paradigm for multi-hop QA. Yet existing systems can only repair errors at the failed step or simply return to the previous step and retry, and thus fail to handle two pervasive failure modes: *inaccurate decomposition*, where the plan is flawed before execution begins, and *error propagation*, where an upstream mistake surfaces downstream. We propose that addressing both requires global self-correction, the ability to identify the root cause of a failure and repair it at the right location. To this end, we propose **LoCoRAG**, a globally self-correcting agentic RAG framework. LoCoRAG maintains a persistent dual memory through a two-stage thinker-generator pipeline, separating what the corpus contains from what the system has concluded. Then, a memory-grounded failure localization module attributes each failure to specific causes and pinpoints its true origin. Together, LoCoRAG drives a self-correcting global replan that selects a matched repair, from local refinement to upstream rollback with downstream regeneration. Across existing multi-hop QA benchmarks, LoCoRAG achieves SOTA performance on all datasets and metrics, with a 10.8 points average LLM-Acc gain over the strongest baseline on 3-hop-or-deeper scenarios.[1]

1 Introduction

Agentic RAG, in which retrieval-augmented LLMs issue queries and perform multi-step reasoning, has become the dominant paradigm for multi-hop question answering (QA) (Yao et al., 2023; Gao et al., 2023). Existing work has developed along two main lines. *Iterative RAG* (Trivedi et al., 2023; Jiang et al., 2023) interleaves retrieval with reasoning but does not decompose the question, thereby placing a heavy burden on the reasoning ability of the LLM. *Plan-based RAG* (Press et al., 2023; Khot et al., 2023; Zhu et al., 2026), in contrast, decomposes the question into a sequence of dependency-aware sub-questions, retrieves evidence by step, and composes the sub-answers into a final response. This structural transparency has made it popular for multi-hop QA (Press et al., 2023; Gao et al., 2023).

Existing plan-based RAG systems are organized around three agentic frameworks. Tree-based query decomposition methods (Cao et al., 2023; Chu et al., 2024; Jiang et al., 2025; Shi et al., 2026) expand multiple decomposition or retrieval trajectories and aggregate or vote over them. Node-level replanning methods (Verma et al., 2024; Zhang et al., 2025a) treat each sub-question as an independent unit and operate on one node at a time. Trajectory-level replanning methods (Chen et al., 2026; Zhu et al., 2026; Nguyen et al., 2025) take the execution trajectory as a whole and reason over the full sequence of sub-questions and intermediate outputs. These approaches improve robustness at different scopes, but do not provide corpus-grounded root-cause localization coupled with a cause-matched modification of the plan. ReAgent (Xinjie et al., 2025) is a complementary backtracking framework that performs local or global rollback when intermediate states conflict; it restores an earlier valid state and retries the existing decomposition, rather than diagnosing why a failure occurred or revising the decomposition itself. Thus, achieving global self-correction in plan-based RAG still requires confronting two fundamental questions.

First, how should the root cause of a failure be localized? When a sub-question fails, the true failure does not necessarily reside at that step. It may stem from an ambiguous or under-decomposed sub-question (**inaccurate decomposition**), or have propagated from an upstream sub-answer (**error propagation**). Effective self-correction therefore requires localizing the failure to its true origin and

*Corresponding author.
[1]Our code is available at https://github.com/zjunet/LoCoRAG

Proceedings of the 2026 Conference on Empirical Methods in Natural Language Processing, pages 49629–49652
October 24-29, 2026 ©2026 Association for Computational Linguistics

Figure 1: A case in MuSiQue illustrating why simple replan is insufficient.

applying a repair whose scope matches the diagnosis: a local fix when the failure is local, a structural rewrite when the decomposition is flawed, and an upstream rollback when an earlier answer is propagated forward. Existing plan-based methods generally anchor correction at the observed failure step, while rollback-oriented methods such as ReAgent address conflict-triggered backtracking without changing the decomposition. As shown in Figure 1, the role inversion at r_2 stems from a decomposition flaw, while r_5 fails because the upstream entity Czechoslovakia is incorrectly carried forward; in either case, rewriting the query at the failed step cannot repair the plan.

Second, on what should correction be grounded? Existing diagnostic mechanisms inspect only the original question, the sub-questions, and the model's own reasoning trace, in isolation from the corpus. This question-and-trace-only view cannot distinguish whether a sub-question failed because the answer is absent from the corpus, because an upstream entity was identified incorrectly, or because the sub-question bundles too many hops. These methods are thus prone to *rewrite loops*, repeatedly rephrasing a query that is unanswerable as posed, because the true cause is invisible to them. As shown in Figure 1, when r_5 fails, counter-evidence pointing to the correct entity Yugoslavia already sits in the retrieved passages, yet any method that consults only the sub-question and the trace has no way to access it. Reliable diagnosis must instead be grounded in what the corpus actually provides. These two questions are not independent: knowing *where* to correct presupposes evidence about *what* actually went wrong, and that evidence can only come from the corpus, not the trace.

To address these two challenges, we propose **LoCoRAG**, a globally self-correcting agentic RAG framework built on the principle of *localize-then-correct*, in which **memory-grounded failure localization** drives a matched global repair. LoCoRAG maintains a persistent dual memory through a two-stage thinker-generator pipeline: an evidence memory $\mathcal{M}_E$ records all passages potentially relevant to the current or the original question, while a process memory $\mathcal{M}_X$ records the sub-answers committed so far, separating what the corpus contains from what the system has concluded. Grounded in both memories, memory-grounded failure localization attributes each failure to one of four causes—surface retrieval gap, ill-specified sub-question, under-decomposed sub-question, or upstream propagation error—and pinpoints the entry point where correction should begin. A **self-correcting global replan** then applies a matched repair from that entry: local refinement, sub-question rewriting, finer decomposition, or upstream rollback with downstream regeneration.

We evaluate LoCoRAG on multi-hop QA benchmarks against a broad range of baselines spanning graph-based, iterative, agentic, and RL-trained retrieval systems. LoCoRAG achieves the best performance across all datasets and metrics, under two base LLMs of substantially different scale and architecture. On MuSiQue and AgenticRAGTracer, the benchmarks that contain 3-hop or deeper questions, LoCoRAG improves LLM-Acc over the strongest baseline by **10.8** points on average. This is precisely the setting in which error propagation and decomposition flaws occur most often, and thus where global self-correction matters most.

Our contributions are summarized as follows:

- We identify that agentic RAG demands global self-correction to overcome error propagation and inaccurate decomposition, two failure modes that prior methods cannot resolve.
- We propose LoCoRAG, which leverages memory-grounded failure localization to enable a self-correcting global replan that provides robust multi-hop reasoning.
- Across four multi-hop QA benchmarks and two base LLMs, LoCoRAG achieves SOTA results, with a 10.8 points LLM-Acc gain on 3-hop-or-deeper scenarios.

2 Related Work

RAG for multi-hop QA RAG grounds LLMs with external documents, but one-shot retrieval is insufficient for multi-hop QA because later hops depend on intermediate entities absent from the surface query (Trivedi et al., 2023; Shao et al., 2023). Graph-based methods such as GraphRAG, HippoRAG2, HyGRAG and Youtu-GraphRAG (Edge et al., 2024; Gutiérrez et al., 2025; Zhong et al., 2026; Dong et al., 2025) organize the corpus into entity–relation structures and exploit graph traversal or community summarization to connect dispersed evidence, at the cost of expensive offline construction. Search-R1 and R1-Searcher instead train LLMs with reinforcement learning to decide when and how to search (Jin et al., 2025a; Song et al., 2025), but require extra training and still offer limited explicit control over recovery. Agentic RAG for multi-hop QA consist of iterative and plan-based RAG. IR-CoT, a classic iterative RAG method, alternates chain-of-thought generation with retrieval. Iter-RetGen retrieves from the current generated context (Trivedi et al., 2023; Shao et al., 2023), yet keep the reasoning state implicit and expose no stable plan for inspection or repair. Plan-based RAG decomposes a question into sub-questions before retrieval, giving each hop a local information need (Zhu et al., 2026; Nguyen et al., 2025; Shi et al., 2026); this line is closest to LoCoRAG, which, rather than improving retrieval structure or learning a search policy, makes plan execution self-correcting when intermediate answers or decompositions fail.

Plan-based RAG. Plan-based RAG differ mainly in the agentic framework. Tree-based methods—ProbTree, BeamAggR, RAG-Star, and RT-RAG (Cao et al., 2023; Chu et al., 2024; Jiang et al., 2025; Shi et al., 2026)—maintain alternative decompositions or retrieval paths and aggregate over them, but repair through redundancy rather than root-cause localization. A second family corrects at the sub-query itself: Plan*RAG executes atomic sub-questions for controllability (Verma et al., 2024), PAR-RAG adds a review loop that revises a weak sub-question or retrieval action (Zhang et al., 2025a). A third family enlarges the corrective scope: LogicRAG guides retrieval and composition with logical or graph-like structures (Chen et al., 2026), REAP couples a planner and evaluator to regenerate failed sub-queries or remaining steps (Zhu et al., 2026), and MA-RAG distributes planning, evidence extraction, verification, and synthesis across agents (Nguyen et al., 2025). These methods move beyond simple query rewriting, yet their correction is driven mainly by the question, plan, or trace—not by what the corpus actually contains. ReAgent (Xinjie et al., 2025) provides a complementary reversible multi-agent framework: a verifier and supervisor trigger local or global rollback when intermediate states conflict, restoring an earlier valid state and retrying the original decomposition. More general studies such as Who&When (Zhang et al., 2025b) and AgenTracer (Zhang et al., 2026) investigate which agent or step causes a failed trajectory, primarily as failure-attribution analysis. In contrast, LoCoRAG performs online, corpus-grounded diagnosis within a structured multi-hop plan and applies a cause-matched repair, including modification of the decomposition when necessary. It therefore instantiates memory-grounded failure localization with a self-correcting global replan for plan-based RAG.

3 Method

3.1 Overview

LoCoRAG is a self-correcting agentic RAG framework for multi-hop question answering. Figure 2 provides an overview of LoCoRAG. It contains four coupled modules: a decomposer that builds a working plan, a two-stage thinker-generator executor that runs the plan while writing dual memory, a memory-grounded localizer that diagnoses non-direct execution statuses, and a global replanner that updates only the affected part of the plan. Given a question q and a corpus $\mathcal{C}$, it maintains a revisable working plan $\mathcal{P} = \{r_1, \ldots, r_n\}$. Each requirement contains a sub-question and dependen-

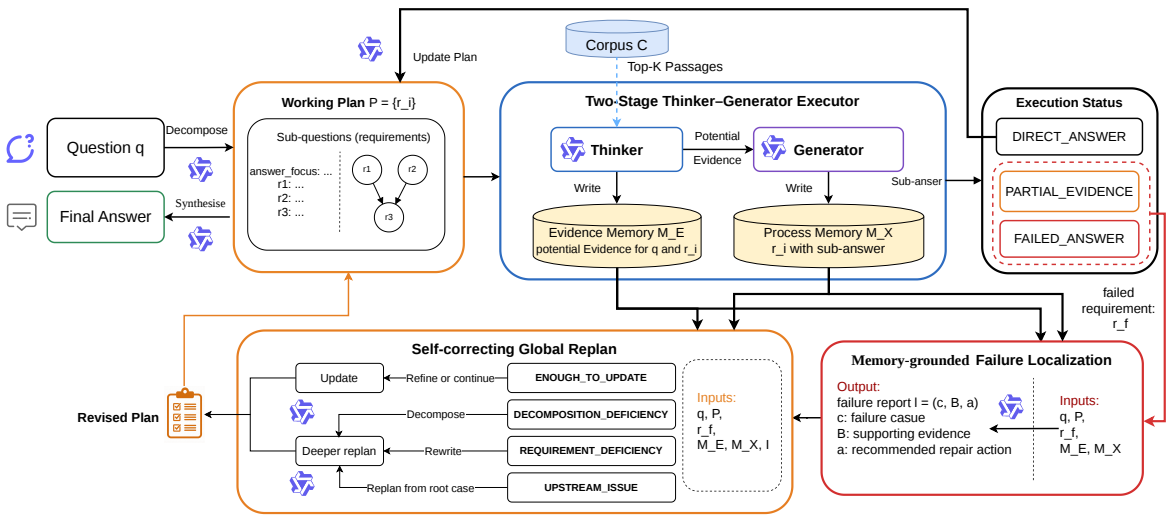

Figure 2: Overall framework of LoCoRAG.

cies on previous requirements:

$$r_i = (q_i, D_i), \tag{1}$$

where $D_i \subset \{r_1, \ldots, r_{i-1}\}$ denotes its dependency set. Execution repeatedly selects requirements whose dependencies have been resolved. Requirements with direct answers update downstream placeholders and support final synthesis; requirements with partial or failed statuses trigger failure localization and replanning. The loop stops when the plan is resolved, sufficient evidence has been gathered, or the replan iteration budget is reached.

3.2 Two-Stage Thinker-Generator Executor

At each iteration, LoCoRAG executes requirements whose dependencies are resolved. For an executable r_i, LoCoRAG first retrieves candidate passages from the corpus and then separates evidence collection from answer commitment. The Thinker is recall-oriented: it reads the original question, the current requirement, and the retrieved passages, and stores evidence that may support either the current hop or a downstream hop. The Generator is commitment-oriented: it uses the collected evidence to produce a sub-answer and an execution status. Formally,

$$\mathcal{R}_i = \text{Retrieve}_k(q_i, \mathcal{C}), \tag{2}$$

$$\mathcal{E}_i = \text{Thinker}(q, r_i, \mathcal{R}_i), \tag{3}$$

$$\tilde{\mathcal{M}}_E = \mathcal{M}_E \cup \{(r_i, e) : e \in \mathcal{E}_i\}, \tag{4}$$

$$(\hat{a}_i, s_i) = \text{Generator}(q, r_i, \mathcal{E}_i). \tag{5}$$

Here $\mathcal{R}_i$ is the top-k passages, and $\mathcal{E}_i$ contains quoted evidence spans. The status s_i belongs to {*DIRECT_ANSWER*, *PARTIAL_EVIDENCE*, *FAILED_ANSWER*}, distinguishing whether the current evidence fully answers the requirement, gives only partial evidence, or fails to anchor the answer. The memory update is then committed:

$$\mathcal{M}_E \leftarrow \tilde{\mathcal{M}}_E, \tag{6}$$

$$\mathcal{M}_X \leftarrow \mathcal{M}_X \cup \{(r_i, \hat{a}_i, s_i)\}. \tag{7}$$

Thus $\mathcal{M}_E$ records what the corpus supports, while $\mathcal{M}_X$ records the process what the system has concluded during execution. This separation is important for later diagnosis: the localizer can compare evidence with committed answers instead of treating every generated sub-answer as equally reliable.

3.3 Memory-Grounded Failure Localization

Failures can arise from the current retrieval step, an underspecified sub-question, an overly coarse decomposition, or an unsupported upstream answer. LoCoRAG distinguishes these cases by diagnosing against both memories instead of applying a fixed repair. As shown in Figure 2, localization is invoked when the executor returns *PARTIAL_EVIDENCE* or *FAILED_ANSWER*. The input therefore includes not only the failed requirement but also the accumulated evidence memory $\mathcal{M}_E$ and process memory $\mathcal{M}_X$.

Given q, the current plan $\mathcal{P}$, the failed requirement r_f, both memories, and its execution trace $\mathcal{I}_f = (r_f, \mathcal{E}_f, \hat{a}_f, s_f)$, the localizer produces a failure report

$$\ell = \text{Localize}(q, \mathcal{P}, r_f, \mathcal{M}_E, \mathcal{M}_X, \mathcal{I}_f) \tag{8}$$

$$= (c, \mathcal{B}, a), \tag{9}$$

where c is the failure cause, $\mathcal{B}$ is the supporting evidence used for the diagnosis, and a is a recommended repair action that conditions the subsequent replan. $\mathcal{B}$ jointly checks three signals: whether completed answers in $\mathcal{M}_X$ are supported by $\mathcal{M}_E$ and consistent with q; whether r_f is atomic and well specified; and whether upstream outputs correctly propagate into the failed step. The resulting cause is one of four types: *ENOUGH_TO_UPDATE*, where existing evidence suffices to continue or lightly refine the plan; *REQUIREMENT_DEFICIENCY*, where the current sub-question is ambiguous or missing constraints; *DECOMPOSITION_DEFICIENCY*, where the step should be split into smaller hops; and *UPSTREAM_ISSUE*, where an earlier unsupported answer causes the downstream failure. This contrast between $\mathcal{M}_E$ and $\mathcal{M}_X$ lets the localizer attribute the failure to what the corpus supports, rather than only to the surface failed output.

3.4 Self-Correcting Global Replan

After localization, LoCoRAG replans from the diagnosed entry point instead of restarting from a fixed position. Given $\ell = (c, \mathcal{B}, a)$, the replanner sets r_f to the failed requirement for current-step errors, or to the earliest upstream root cause identified by $\mathcal{B}$. The revised plan is then returned to the working plan in Figure 2, so validated parts can continue without being regenerated. We use $A(r_f)$ to denote the region that should be revised:

$$A(r_f) = \{r_j \in \mathcal{P} : r_j = r_f \text{ or } r_f \rightarrow^* r_j\}, \quad (10)$$

where $r_f \rightarrow^* r_j$ means that r_j depends directly or transitively on r_f. In implementation, this region is not computed as a separate deterministic preprocessing step. Instead, the LLM replanner identifies the affected requirements and produces a complete revised plan in the same call, using $A(r_f)$ as a conceptual description of the intended correction scope. Validated requirements outside the affected region and all evidence in $\mathcal{M}_E$ are provided as context so that they can be preserved. The repair branch follows the localized cause. If $c =$ *ENOUGH_TO_UPDATE*, LoCoRAG applies UPDATE: it absorbs partial evidence to continue or slightly refine sub-question without structural replanning. For the other three causes, LoCoRAG enters a deeper replan stage. There, *REQUIREMENT_DEFICIENCY* triggers REWRITE to add missing constraints or disambiguation, *DECOMPOSITION_DEFICIENCY* triggers DECOMPOSE to replace a coarse step with smaller dependent requirements, and *UPSTREAM_ISSUE* triggers REPLANFROMROOTCAUSE to roll back to the upstream source and regenerate the downstream plan. This keeps correction bounded in cost: the LLM may modify downstream dependencies when necessary, but it is instructed to preserve validated requirements that are outside the diagnosed scope. Given the shared replan input $\mathbf{z}$, the revised plan is selected by the localized cause:

$$\mathbf{z} = (q, \mathcal{P}, r_f, \ell, \mathcal{M}_E, \mathcal{M}_X). \quad (11)$$

$$\mathcal{P}' = \begin{cases} \text{Update}(\mathbf{z}), & c = \textit{ENOUGH_TO_UPDATE}, \\ \text{DeeperReplan}_a(\mathbf{z}), & \text{otherwise}. \end{cases} \quad (12)$$

Thus *ENOUGH_TO_UPDATE* yields a lightweight updated full plan, while all other localized causes invoke action-conditioned deeper replanning whose affected region is identified internally by the LLM. The selected $\mathcal{P}'$ is then applied to $\mathcal{P}$, and execution resumes from the newly executable requirements. This gives LoCoRAG a variable correction scope: local when the error is local, structural when the plan is flawed, and upstream-aware when the failure is propagated.

4 Experiments

4.1 Settings

We evaluate on three widely used multi-hop QA benchmarks: **2WikiMultiHopQA** (Ho et al., 2020), **MuSiQue** (Trivedi et al., 2022), and **HotpotQA** (Yang et al., 2018). We follow the same question sampling and corpus construction protocol as HippoRAG2 (Gutiérrez et al., 2025) and Youtu-GraphRAG (Dong et al., 2025) to ensure a fair comparison. We report **Cover-EM**, **F1**, and **LLM-Acc**: Cover-EM checks whether the prediction contains the gold answer, F1 measures token-level overlap, and LLM-Acc uses DeepSeek-V3.2 (DeepSeek-AI, 2025) as an LLM judge to assess semantic correctness. We compare LoCoRAG with representative retrieval-augmented QA systems spanning five families: single-turn methods (Inference-only, Naive RAG), graph-based retrieval (HippoRAG2 (Gutiérrez et al., 2025), LogicRAG (Chen et al., 2026), Youtu-GraphRAG (Dong et al., 2025)), iterative retrieval (IR-CoT (Trivedi et al., 2023), Iter-RetGen (Shao et al., 2023)), agentic and plan-based RAG (REAP (Zhu et al., 2026), MA-RAG (Nguyen et al., 2025), RT-RAG (Shi et al., 2026)), and RL-trained

Category	Method	2WikiMultiHopQA			MuSiQue			HotpotQA		
		cover-EM	F1	LLM-Acc	cover-EM	F1	LLM-Acc	cover-EM	F1	LLM-Acc
Single	Inference-only	34.5	34.30	36.5	9.3	13.85	18.9	32.2	38.01	45.2
	Naive-RAG	54.0	53.31	58.7	23.0	26.21	30.7	64.2	67.38	78.7
Graph-Based	HippoRAG2	75.0	71.51	79.1	35.9	40.71	48.8	67.2	72.27	83.9
	LogicRAG	77.0	75.26	81.3	30.4	38.81	44.7	58.0	66.64	78.7
	Youtu-GraphRAG	82.9	17.48	85.0	28.5	9.37	38.5	73.6	21.36	85.6
Tune-Based	Search-R1-7B	67.3	66.95	71.9	37.5	45.97	51.0	64.1	73.06	81.0
	R1-Searcher-7B	74.0	72.23	77.8	42.4	48.61	52.3	70.0	76.14	84.7
	REAP-8B	73.5	74.68	80.6	35.4	41.33	46.4	63.0	71.15	80.3
Iterative\ Agentic	REAP-prompt	72.7	52.96	72.2	38.7	33.85	45.8	70.4	59.27	81.7
	IR-CoT	81.0	54.03	78.5	35.9	27.74	42.3	68.3	63.88	81.3
	Iter-RetGen	61.6	59.63	65.4	31.2	33.88	40.3	69.4	72.95	83.5
	MA-RAG	69.5	65.19	72.3	34.7	36.91	42.6	59.1	60.63	71.5
	RT-RAG	83.0	78.24	85.5	18.9	27.68	31.3	37.8	45.77	54.1
Ours	**LoCoRAG**	**84.6**	**79.75**	**86.7**	**52.7**	**52.64**	**63.3**	**73.9**	**76.71**	**87.2**

Table 1: Main results with Qwen3-Next-80B-A3B-Instruct as the base LLM. Performance comparison on three multi-hop QA benchmarks (2WikiMultiHopQA, MuSiQue, and HotpotQA). Bold is best, underlined is second-best.

retrieval agents (Search-R1 (Jin et al., 2025a), R1-Searcher (Song et al., 2025)). Details of datasets, baselines and implementation are in Appendix B.

4.2 Main Results

4.2.1 Results Against Baselines

Table 1 presents the main results on Qwen3-Next. LoCoRAG achieves SOTA performance across all three datasets and all three metrics, showing that Memory-grounded Failure Localization outperform graph-based methods, trained search policies, and RAG with simple replan on multi-hop QA.

Compared with the strongest non-LoCoRAG baseline on each metric. The gains are consistent across benchmarks, but their magnitude tracks how strongly each dataset stresses genuine multi-hop decomposition: on MuSiQue, where some questions require three or more supporting hops, LoCoRAG's 52.7 Cover-EM and 63.3 LLM-Acc substantially outperform both RL-trained search agents and existing agentic RAG, supporting our hypothesis that the bottleneck on long reasoning chains is not one-shot retrieval quality but whether the system identifies the true source of failure and repairs the plan from the correct entry point.

On 2WikiMultiHopQA and HotpotQA, the margins are smaller but remain consistent, and LoCoRAG is the only method ranking first on every metric. Although Youtu-GraphRAG attains high Cover-EM and LLM-Acc, its lower F1 indicates verbose outputs; LoCoRAG, in contrast, produces answers that are both semantically correct and well calibrated to the expected answer form.

Looking across baseline families, the common gap is the absence of memory-grounded failure localization. Graph-based methods improve multi-hop retrieval and single-step execution but cannot correct an executed-then-failed step. IR-CoT and Iter-RetGen, lacking an explicit plan, drop sharply on MuSiQue, where longer chains demand more structured reasoning. Search-R1 and R1-Searcher learn a multi-step search policy via reinforcement learning but expose no explicit recovery mechanism. Among methods that do replan, REAP has the strongest existing mechanism—trajectory replanning with sub-query regeneration—yet its correction is anchored at the failed step and driven by the trajectory rather than corpus evidence, widening the MuSiQue performance gap relative to LoCoRAG; RT-RAG is competitive on 2WikiMultiHopQA but drops on MuSiQue and HotpotQA, showing that tree-structured decomposition and repeated sampling alone are insufficient against inaccurate decomposition and error propagation in deeper multihop question.

Across these categories, the variable that aligns with LoCoRAG's gains is whether the system can diagnose *where* along the plan the failure originates rather than *which* step has failed, which is why the MuSiQue gap is by far the largest in the table.

4.2.2 Generalization across Base LLMs

Table 2 reports results with Qwen3.5-27B, an LLM of a different architecture and scale. LoCoRAG remains the leading method, achieving the highest LLM-Acc and Cover-EM among other methods, with average gains of 4.5% and 3.0% over the strongest baselines. The original LoCoRAG

Category	Method	2WikiMultiHopQA			MuSiQue			HotpotQA		
		cover-EM	F1	LLM-Acc	cover-EM	F1	LLM-Acc	cover-EM	F1	LLM-Acc
Single	Inference-only	32.7	35.42	35.6	7.4	13.04	14.7	27.1	31.34	36.7
	Naive-RAG	53.2	53.85	56.7	22.3	28.39	30.4	65.6	69.39	78.3
Graph-Based	HippoRAG2	78.8	73.82	81.0	43.5	43.89	53.8	67.6	72.87	85.9
	LogicRAG	80.0	76.54	82.6	33.9	41.22	47.9	64.8	71.48	82.8
	Youtu-GraphRAG	85.7	14.87	84.1	56.4	18.20	63.9	75.8	22.21	86.5
Iterative\ Agentic	REAP-prompt	80.5	50.38	79.8	44.2	37.89	49.5	72.8	58.14	82.7
	IR-CoT	74.4	49.70	65.7	32.6	21.57	37.4	65.8	53.40	74.4
	Iter-RetGen	60.4	59.69	64.1	31.8	37.32	40.0	69.8	73.86	84.4
	MA-RAG	85.5	73.73	85.2	45.6	45.67	51.1	66.4	66.33	79.9
	RT-RAG	84.4	80.14	87.6	18.0	25.96	29.1	31.8	38.97	44.6
Ours	LoCoRAG	**88.7**	68.51	90.2	**57.8**	48.07	67.7	**78.1**	67.00	**90.4**
	+only_answer	88.8	**81.51**	**91.2**	56.3	**55.71**	**68.1**	76.4	**73.88**	90.3

Table 2: Main results with Qwen3.5-27B as the base LLM. Performance comparison on three multi-hop QA benchmarks (2WikiMultiHopQA, MuSiQue, and HotpotQA).

slightly lags in F1 on 2WikiMultiHopQA and HotpotQA; case-level analysis attributes this to an evaluation artifact—some questions admit multiple valid answers while the gold lists only one, and LoCoRAG's correct multi-answer outputs are penalized by token-level F1 against a single reference.

Constraining the final-answer prompt to emit exactly one answer (`+only_answer`) removes this artifact and places LoCoRAG first in F1 on all three datasets. Notably, LoCoRAG attains even higher accuracy on Qwen3.5-27B than on Qwen3-Next while preserving a substantial margin over baselines, confirming that the improvement is not tied to a specific base model and that, as a training-free framework, LoCoRAG further benefits from stronger underlying LLMs.

Further re-evaluating LLM-Acc with DeepSeek-V4-Flash (DeepSeek-AI, 2026) under a relaxed multi-answer judge prompt (Appendix C.1) likewise places LoCoRAG first on all three datasets (91.6 / 70.1 / 91.3), with +2.7% / +7.4% / +4.5% margins over the strongest baseline.

4.3 Detailed Analysis

4.3.1 Ablation Study

Table 3 reports the ablation on Qwen3-Next-80B-A3B-Instruct. The variants follow the modules in Section 3: the first block tests the two-stage Thinker–Generator executor, the middle row removes Memory-grounded Failure Localization, and the final block weakens Self-correcting Global Replan by removing either an action or a memory input. The largest drops concentrate on MuSiQue, where longer reasoning chains make both failure diagnosis and correction scope more important.

Two-stage thinker-generator executor. Removing the Thinker's recall-oriented evidence extraction with original passages (*w/o Thinker evidence extraction*) degrades all three benchmarks, with the clearest drop on MuSiQue (-3.7% Cover-EM and -3.5% LLM-Acc relatively over full version). This supports the executor design in Section 3.2: $\mathcal{M}_E$ should store corpus-grounded evidence before the Generator commits a sub-answer and status. Dropping the original question from the Thinker–Generator prompt (*w/o original-question in T–G*) also hurts performance, especially on HotpotQA (-2.7% Cover-EM, -2.7% F1, and -1.9% LLM-Acc) and MuSiQue LLM-Acc (-4.3%). This indicates that executing each requirement in isolation is insufficient; the original question keeps evidence selection and sub-answer commitment aligned with the global answer target.

Localization and global replan. Removing Memory-grounded Failure Localization causes the largest single degradation, especially on MuSiQue (-9.3% Cover-EM, -8.90% F1, and -11.5% LLM-Acc relatively). This confirms that non-direct statuses cannot be repaired reliably by treating the observed failed requirement as the default repair point; the system needs to compare $\mathcal{M}_E$ and $\mathcal{M}_X$ to identify whether the cause is local, structural, or upstream. The two replan ablations further isolate how the diagnosis is used. Without upstream replan, MuSiQue drops by 4.7% Cover-EM and 4.3% LLM-Acc, showing that rolling back to an earlier root cause is necessary under error propagation especially for deeper multi-hop questions. Without evidence memory in replan, MuSiQue and HotpotQA both decline, showing that the replanner

Variant	2WikiMultiHopQA			MuSiQue			HotpotQA		
	Cover-EM	F1	LLM-Acc	Cover-EM	F1	LLM-Acc	Cover-EM	F1	LLM-Acc
LoCoRAG (full)	**84.6**	**79.75**	**86.7**	**52.7**	**52.64**	**63.3**	**73.9**	**76.71**	**87.2**
w/o Thinker evidence extraction	83.8(-0.8)	78.77(-0.98)	85.8(-0.9)	50.6(-2.1)	51.86(-0.78)	61.1(-2.2)	73.7(-0.2)	76.54(-0.17)	86.8(-0.4)
w/o original-question in T–G	83.8(-0.8)	79.18(-0.57)	86.2(-0.5)	50.9(-1.8)	52.00(-0.64)	60.6(-2.7)	71.9(-2)	74.64(-2.07)	85.5(-1.7)
w/o Memory-grounded Failure Localization	83.5(-1.1)	78.68(-1.07)	85.9(-0.8)	47.8(-4.9)	47.94(-4.70)	56.0(-7.3)	72.8(-1.1)	75.78(-0.93)	86.3(-0.9)
w/o upstream replan	84.1(-0.5)	79.53(-0.22)	86.4(-0.3)	50.2(-2.5)	51.09(-1.55)	60.6(-2.7)	73.8(-0.1)	76.39(-0.93)	86.5(-0.7)
w/o evidence memory in replan	84.0(-0.6)	79.00(-0.75)	85.7(-1.0)	50.0(-2.7)	51.27(-1.37)	61.3(-2.0)	72.3(-1.6)	75.61(-1.10)	85.9(-1.3)

Table 3: Ablation study on Qwen3-Next-80B-A3B-Instruct.

Dataset	Cases	LoCoRAG	Pure-LLM	Oracle
2Wiki	163	55.21	47.85	77.30
HotpotQA	209	75.12	64.59	82.30
MuSiQue	434	41.24	31.34	60.14

Table 4: Final QA accuracy on cases that trigger localization.

needs corpus-grounded evidence to decide which affected region to revise while preserving validated parts of the working plan.

4.3.2 Evaluation of Failure Localization

We directly evaluate the component that implements the paper's central claim. From each benchmark, we randomly sample 100 cases that trigger localization; a case may contain multiple localization events. Given the official supporting-fact paragraphs, human annotators judge whether the predicted cause and analysis are correct, whether the proposed repair location is correct, and whether the revised plan moves in the correct direction. The event-level results are reported in Appendix C.2; here we focus on the end-to-end consequence of replacing structured localization with free-form analysis. *Pure-LLM analysis* removes the structured cause set and evidence checks, while *Oracle* answers with chain-of-thought using only official supporting facts and estimates answerability under ideal retrieval. As shown in Table 4, structured localization improves over free-form analysis by 7.36, 10.53, and 9.90 accuracy points on 2Wiki, HotpotQA, and MuSiQue, respectively.

4.3.3 Multi-Answer Evaluation

Some benchmark questions admit multiple valid answers, while the gold annotation lists only one. To reduce this single-reference bias, we re-evaluate LLM-Acc with **DeepSeek-V4-Flash** (DeepSeek-AI, 2026) under a relaxed prompt that accepts any plausible valid answer. As shown in Table 5, LoCoRAG remains best on all three datasets, reaching 91.6/70.1/91.3 on 2WikiMultiHopQA, MuSiQue, and HotpotQA, respectively. This result complements the single-reference metrics and confirms that the gains are not caused by matching one particular gold string.

Method	2Wiki	MuSiQue	HotpotQA
Inference-only	35.7	14.0	36.5
Naive-RAG	57.1	30.8	79.4
HippoRAG2	81.8	56.7	85.5
LogicRAG	83.8	49.7	83.6
Youtu-GraphRAG	85.3	65.3	87.4
Search-R1-7B	71.7	50.7	81.8
R1-Searcher-7B	78.0	51.4	85.1
REAP-8B	80.5	46.9	81.0
REAP-prompt	81.1	51.2	83.3
IR-CoT	70.5	35.8	76.8
Iter-RetGen	64.9	41.0	84.5
MA-RAG	89.2	58.6	83.0
RT-RAG	88.5	29.4	44.5
LoCoRAG	**91.6**	**70.1**	**91.3**

Table 5: LLM-Acc under the multi-answer evaluation protocol with DeepSeek-V4-Flash as the judge. Best results are highlighted in **bold**.

4.3.4 Efficiency Analysis

Efficiency evaluation is conducted using 100 randomly sampled queries per dataset, where LoCoRAG is configured with a replan budget of 4 iterations and compared against agentic and plan-based baselines. To ensure a fair comparison, all methods are executed under identical serving conditions using Qwen3-Next-80B-A3B-Instruct via the official Alibaba Cloud API. As shown in Figure 3, LoCoRAG achieves the optimal accuracy-efficiency trade-off. It attains the highest accuracy and is slightly less efficient than MA-RAG and REAP. In sharp contrast, high-cost baseline methods fail to translate their substantially larger budgets into better performance. RT-RAG consumes more than 1.4 million input tokens on 2WikiMultiHopQA, over 100 times that of LoCoRAG, while Youtu-GraphRAG requires 3,557 seconds per query on MuSiQue, approximately 100 times

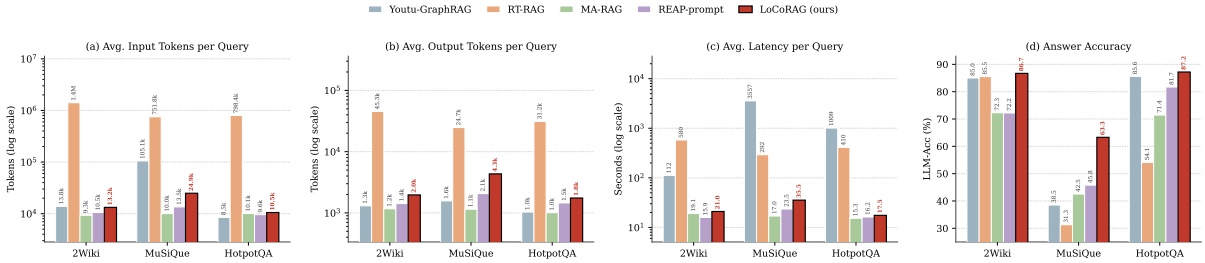

Figure 3: Efficiency comparison of LoCoRAG against agentic baselines. Panels (a)–(c) report the average input tokens, output tokens, and end-to-end latency per query; panel (d) reports the corresponding LLM-Acc.

Method	Cover-EM	F1	LLM-Acc
Inference-only	26.05	27.23	27.82
Naive-RAG	37.32	28.01	30.04
Search-R1-7B	46.90	53.53	58.70
R1-Searcher-7B	51.58	56.67	61.99
REAP-8B	47.74	54.88	60.84
REAP-prompt	58.54	37.27	62.53
IR-CoT	43.75	18.77	41.84
Iter-RetGen	40.00	31.67	33.49
LoCoRAG	**62.30**	**64.77**	**71.42**

Table 6: Results on AgenticRAGTracer with a large-scale Wikipedia corpus (wiki-18).

longer than LoCoRAG. LoCoRAG avoids such explosive growth because its self-correction loop replans and re-executes retrieval only for the part of the plan diagnosed as faulty when needed. As a result, the additional computation is confined to a small incremental overhead, without requiring repeated sampling or graph-based retrieval.

4.3.5 Corpus Expansion

To evaluate corpus scalability, we test on AgenticRAGTracer (You et al., 2026) on Qwen3-Next using the full Wikipedia (wiki-18) corpus following Search-R1 (Jin et al., 2025a). Graph-based methods are excluded as indexing millions of documents is impractical for them. As shown in Table 6, LoCoRAG achieves the best performance across all three metrics, outperforming the strongest baselines by 6.4%, 14.3%, and 14.2%, respectively. Iterative methods without explicit planning (IR-CoT, Iter-RetGen) degrade substantially under the larger corpus, while LoCoRAG maintains robust performance through its memory-grounded failure localization and targeted replanning, which confines re-retrieval to the diagnosed faulty steps rather than requiring exhaustive re-execution.

4.3.6 Case Study

We present three representative cases to show how LoCoRAG uses Memory-grounded Failure Localization and Self-correcting Global Replan to self-correct the plan. The complete traces are provided in Appendix D and Tables 18, 19, and 20.

5 Conclusion

We propose LoCoRAG, a globally self-correcting agentic RAG framework built on the principle of localize-then-correct. LoCoRAG maintains a persistent dual memory through a two-stage pipeline, on which *memory-grounded failure localization* attributes each failure to its true root cause and drives a self-correcting global replan that selects a matched repair. Empirically, LoCoRAG achieves SOTA performance on all datasets and metrics, with a 10.8 points average LLM-Acc gain on 3-hop-or-deeper scenarios, showing that effective self-correction in agentic RAG requires knowing *where* and *why* a plan went wrong.

Acknowledgments

This work is supported by NSFC (No. 62606466, No. 62322606, No. 62441605) and a collaboration funding by FinVolution Group.

Limitations

LoCoRAG is designed as a training-free framework with no learnable parameters, which makes it portable across backbones and corpora but also prevents it from being fitted to a specific dataset. Our evaluation centers on multi-hop QA, where error propagation and inaccurate decomposition are most pronounced and global self-correction has the most to repair. LoCoRAG therefore offers limited benefit and less efficient on single-hop QA, where one retrieval pass typically suffices, and presupposes a base LLM capable of producing usable plans and

failure attributions; on substantially smaller models, decomposition and localization quality degrade together, leaving the corrective loop with less to work with.

Ethics Considerations

As an LLM-based agentic framework, LoCoRAG relies on LLM outputs at multiple stages, whose responses may contain hallucinations or unsupported claims. To mitigate this, LoCoRAG is built around a dual-memory design: the evidence memory $\mathcal{M}_E$ records corpus-grounded passages, while the process memory $\mathcal{M}_X$ records the sub-answers the system commits to. Users can therefore inspect both memories at inference time to verify whether the final response is supported by retrieved evidence rather than parametric guessing. In this work, AI-assisted writing is used solely for refining the manuscript. The authors reviewed and verified all AI-assisted content and are responsible for the final paper.

References

Shulin Cao, Jiajie Zhang, Jiaxin Shi, Xin Lv, Zijun Yao, Qi Tian, Juanzi Li, and Lei Hou. 2023. Probabilistic tree-of-thought reasoning for answering knowledge-intensive complex questions. *arXiv preprint arXiv:2311.13982*.

Jianlv Chen, Shitao Xiao, Peitian Zhang, Kun Luo, Defu Lian, and Zheng Liu. 2025. M3-embedding: Multilinguality, multi-functionality, multi-granularity text embeddings through self-knowledge distillation. *Preprint*, arXiv:2402.03216.

Shengyuan Chen, Chuang Zhou, Zheng Yuan, Qinggang Zhang, Zeyang Cui, Hao Chen, Yilin Xiao, Jiannong Cao, and Xiao Huang. 2026. You don't need pre-built graphs for RAG: Retrieval augmented generation with adaptive reasoning structures. In *Proceedings of the AAAI Conference on Artificial Intelligence*.

Zheng Chu, Jingchang Chen, Qianglong Chen, Haotian Wang, Kun Zhu, Xiyuan Du, Weijiang Yu, Ming Liu, and Bing Qin. 2024. Beamaggr: Beam aggregation reasoning over multi-source knowledge for multi-hop question answering. *arXiv preprint arXiv:2406.19820*.

DeepSeek-AI. 2025. Deepseek-v3.2: Pushing the frontier of open large language models. *Preprint*, arXiv:2512.02556.

DeepSeek-AI. 2026. DeepSeek-V4: Towards highly efficient million-token context intelligence. `https://huggingface.co/deepseek-ai/DeepSeek-V4-Flash`. Preview release, April 24, 2026. Model: 284B total / 13B active parameters, 1M-token context window.

Junnan Dong, Siyu An, Yifei Yu, Qian-Wen Zhang, Linhao Luo, Xiao Huang, Yunsheng Wu, Di Yin, and Xing Sun. 2025. Youtu-GraphRAG: Vertically unified agents for graph retrieval-augmented complex reasoning. *arXiv preprint arXiv:2508.19855*.

Darren Edge, Ha Trinh, Newman Cheng, Joshua Bradley, Alex Chao, Apurva Mody, Steven Truitt, Dasha Metropolitansky, Robert Osazuwa Ness, and Jonathan Larson. 2024. From local to global: A graph rag approach to query-focused summarization. *arXiv preprint arXiv:2404.16130*.

Yunfan Gao, Yun Xiong, Xinyu Gao, Kangxiang Jia, Jinliu Pan, Yuxi Bi, Yi Dai, Jiawei Sun, Meng Wang, and Haofen Wang. 2023. Retrieval-augmented generation for large language models: A survey. *arXiv preprint arXiv:2312.10997*.

Bernal Jiménez Gutiérrez, Yiheng Shu, Weijian Qi, Sizhe Zhou, and Yu Su. 2025. From rag to memory: Non-parametric continual learning for large language models. *Preprint*, arXiv:2502.14802.

Xiaoxin He, Yijun Tian, Yifei Sun, Nitesh V. Chawla, Thomas Laurent, Yann LeCun, Xavier Bresson, and Bryan Hooi. 2024. G-retriever: Retrieval-augmented generation for textual graph understanding and question answering. In *Advances in Neural Information Processing Systems*, volume 37, pages 132876–132907. Curran Associates, Inc.

Xanh Ho, Anh-Khoa Duong Nguyen, Saku Sugawara, and Akiko Aizawa. 2020. Constructing a multi-hop qa dataset for comprehensive evaluation of reasoning steps. In *Proceedings of the 28th International Conference on Computational Linguistics*, pages 6609–6625.

Jinhao Jiang, Jiayi Chen, Junyi Li, Ruiyang Ren, Shijie Wang, Wayne Xin Zhao, Yang Song, and Tao Zhang. 2025. RAG-star: Enhancing deliberative reasoning with retrieval augmented verification and refinement. In *Proceedings of the 2025 Conference of the Nations of the Americas Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, pages 7064–7074, Albuquerque, New Mexico. Association for Computational Linguistics.

Zhengbao Jiang, Frank Xu, Luyu Gao, Zhiqing Sun, Qian Liu, Jane Dwivedi-Yu, Yiming Yang, Jamie Callan, and Graham Neubig. 2023. Active retrieval augmented generation. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 7969–7992, Singapore. Association for Computational Linguistics.

Bowen Jin, Hansi Zeng, Zhenrui Yue, Jinsung Yoon, Sercan Arik, Dong Wang, Hamed Zamani, and Jiawei Han. 2025a. Search-r1: Training llms to reason and leverage search engines with reinforcement learning. *arXiv preprint arXiv:2503.09516*.

Jiajie Jin, Yutao Zhu, Zhicheng Dou, Guanting Dong, Xinyu Yang, Chenghao Zhang, Tong Zhao, Zhao Yang, and Ji-Rong Wen. 2025b. FlashRAG: A modular toolkit for efficient retrieval-augmented generation research. In *Companion Proceedings of the ACM on Web Conference 2025*, pages 737–740. ACM.

Tushar Khot, Harsh Trivedi, Matthew Finlayson, Yao Fu, Kyle Richardson, Peter Clark, and Ashish Sabharwal. 2023. Decomposed prompting: A modular approach for solving complex tasks. In *The Eleventh International Conference on Learning Representations (ICLR)*.

Thang Nguyen, Peter Chin, and Yu-Wing Tai. 2025. Ma-rag: Multi-agent retrieval-augmented generation via collaborative chain-of-thought reasoning. *arXiv preprint arXiv:2505.20096*.

Ofir Press, Muru Zhang, Sewon Min, Ludwig Schmidt, Noah Smith, and Mike Lewis. 2023. Measuring and narrowing the compositionality gap in language models. In *Findings of the Association for Computational Linguistics: EMNLP 2023*, pages 5687–5711, Singapore. Association for Computational Linguistics.

Qwen Team. 2026. Qwen3.5: Towards native multimodal agents.

Zhihong Shao, Yeyun Gong, Yelong Shen, Minlie Huang, Nan Duan, and Weizhu Chen. 2023. Enhancing retrieval-augmented large language models with iterative retrieval-generation synergy. In *Findings of the Association for Computational Linguistics: EMNLP 2023*, pages 9248–9274, Singapore. Association for Computational Linguistics.

Yuling Shi, Maolin Sun, Zijun Liu, Mo Yang, Yixiong Fang, Tianran Sun, and Xiaodong Gu. 2026. Reasoning in trees: Improving retrieval-augmented generation for multi-hop question answering. *arXiv preprint arXiv:2601.11255*.

Huatong Song, Jinhao Jiang, Yingqian Min, Jie Chen, Zhipeng Chen, Wayne Xin Zhao, Lei Fang, and Ji-Rong Wen. 2025. R1-searcher: Incentivizing the search capability in llms via reinforcement learning. *arXiv preprint arXiv:2503.05592*.

Yifei Sun, Haoran Deng, Yang Yang, Chunping Wang, Jiarong Xu, Renhong Huang, Linfeng Cao, Yang Wang, and Lei Chen. 2022. Beyond homophily: Structure-aware path aggregation graph neural network. In *IJCAI*, pages 2233–2240.

Yifei Sun, Zemin Liu, Bryan Hooi, Yang Yang, Rizal Fathony, Jia Chen, and Bingsheng He. 2025a. Multi-label node classification with label influence propagation. In *International Conference on Learning Representations*, volume 2025, pages 15837–15861.

Yifei Sun, Yang Yang, Xiao Feng, Zijun Wang, Haoyang Zhong, Chunping Wang, and Lei Chen. 2025b. Handling feature heterogeneity with learnable graph patches. In *Proceedings of the 31st ACM SIGKDD Conference on Knowledge Discovery and Data Mining V. 1*, pages 1313–1324.

Yifei Sun, Qi Zhu, Yang Yang, Chunping Wang, Tianyu Fan, Jiajun Zhu, and Lei Chen. 2024. Fine-tuning graph neural networks by preserving graph generative patterns. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 38, pages 9053–9061.

Qwen Team. 2025. Qwen3 technical report. *Preprint*, arXiv:2505.09388.

Harsh Trivedi, Niranjan Balasubramanian, Tushar Khot, and Ashish Sabharwal. 2022. MuSiQue: Multi-hop questions via single-hop question composition. *Transactions of the Association for Computational Linguistics*.

Harsh Trivedi, Niranjan Balasubramanian, Tushar Khot, and Ashish Sabharwal. 2023. Interleaving retrieval with chain-of-thought reasoning for knowledge-intensive multi-step questions. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 10014–10037.

Prakhar Verma, Sukruta Prakash Midigeshi, Gaurav Sinha, Arno Solin, Nagarajan Natarajan, and Amit Sharma. 2024. Plan*rag: Efficient test-time planning for retrieval augmented generation. *arXiv preprint arXiv:2410.20753*.

Liang Wang, Nan Yang, Xiaolong Huang, Binxing Jiao, Linjun Yang, Daxin Jiang, Rangan Majumder, and Furu Wei. 2022. Text embeddings by weakly-supervised contrastive pre-training. *arXiv preprint arXiv:2212.03533*.

Zhao Xinjie, Fan Gao, Xingyu Song, Yingjian Chen, Rui Yang, Yanran Fu, Yuyang Wang, Yusuke Iwasawa, Yutaka Matsuo, and Irene Li. 2025. ReAgent: Reversible multi-agent reasoning for knowledge-enhanced multi-hop QA. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pages 4067–4089, Suzhou, China. 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.

Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik Narasimhan, and Yuan Cao. 2023. ReAct: Synergizing reasoning and acting in language models. In *International Conference on Learning Representations*.

Qijie You, Wenkai Yu, and Wentao Zhang. 2026. Agenticragtracer: A hop-aware benchmark for diagnosing multi-step retrieval reasoning in agentic rag. *arXiv preprint arXiv:2602.19127*.

Guibin Zhang, Junhao Wang, Junjie Chen, Wangchunshu Zhou, Kun Wang, and Shuicheng Yan. 2026.

Agentracer: Who is inducing failure in the llm agentic systems? In *International Conference on Learning Representations*, volume 2026, pages 11377–11399.

Ningning Zhang, Chi Zhang, Zhizhong Tan, Xingxing Yang, Weiping Deng, and Wenyong Wang. 2025a. Credible plan-driven rag method for multi-hop question answering. *arXiv preprint arXiv:2504.16787*.

Shaokun Zhang, Ming Yin, Jieyu Zhang, Jiale Liu, Zhiguang Han, Jingyang Zhang, Beibin Li, Chi Wang, Huazheng Wang, Yiran Chen, and 1 others. 2025b. Which agent causes task failures and when? on automated failure attribution of llm multi-agent systems. *arXiv preprint arXiv:2505.00212*.

Haoyang Zhong, Yifei Sun, Antong Zhang, Chunping Wang, Lei Chen, and Yang Yang. 2026. A unified framework for context-aware and relation-aware graph retrieval-augmented generation. In *Proceedings of the ACM Web Conference 2026 (WWW '26)*, pages 2477–2488. Association for Computing Machinery.

Yijie Zhu, Haojie Zhou, Wanting Hong, Tailin Liu, and Ning Wang. 2026. REAP: Enhancing RAG with recursive evaluation and adaptive planning for multi-hop question answering. In *Proceedings of the AAAI Conference on Artificial Intelligence*.

A Method

A.1 Notation

Table 7 summarizes the symbols and terminology used throughout the Method section (§3).

B Additional Experimental Setups

Datasets. We evaluate LoCoRAG on four multi-hop question answering benchmarks. **HotpotQA** (Yang et al., 2018) is a foundational Wikipedia-based benchmark that pioneered cross-document evidence aggregation with paragraph-level supporting fact annotation. It contains two major question types: *bridge* questions, which require following an intermediate entity from one document to another, and *comparison* questions, which require extracting attributes of two entities and comparing them to choose the answer. **2WikiMultiHopQA** (Ho et al., 2020) addresses reasoning shortcut vulnerabilities by grounding question construction in explicit Wikidata relations. Its structured evidence triples and predefined reasoning schemas provide a controllable testbed organized around four question types: *comparison*, which compares properties of two entities; *inference*, which combines multiple relations to derive a missing fact; *compositional*, which chains the answer of one sub-question into the next; and *bridge-comparison*, which first resolves bridge entities and then compares their attributes. **MuSiQue** (Trivedi et al., 2022) offers the most rigorous stress-test of multi-hop reasoning by composing 2-to-4-hop questions from verified single-hop chains and introducing unanswerable variants that systematically remove supporting evidence. This design effectively eliminates shortcut exploitation, making it particularly demanding for sequential planning and long-chain reasoning.

AgenticRAGTracer (You et al., 2026) is a hop-aware benchmark designed specifically for diagnosing multi-step retrieval reasoning in Agentic RAG. The benchmark contains 1,305 examples across multiple domains and is constructed to avoid overlap with mainstream multi-hop QA datasets. Its questions cover two reasoning topologies: sequential *inference* questions, where later hops depend on earlier answers, and parallel *comparison* questions, where evidence from multiple branches must be retrieved and compared; both are extended to 2-to-4-hop settings. We use it as a large-corpus stress test with the full Wikipedia (wiki-18) corpus, where the main challenge is not only composing evidence across hops but also maintaining a correct retrieval trajectory under a substantially larger search space.

For HotpotQA, 2WikiMultiHopQA, and MuSiQue, we follow the question sampling and corpus construction protocol used by HippoRAG2 (Gutiérrez et al., 2025) and Youtu-GraphRAG (Dong et al., 2025) so that LoCoRAG and all baselines are evaluated on the same question sets and retrieval corpora. Each dataset has 1000 questions.

Baselines. We compare LoCoRAG against a broad spectrum of retrieval-augmented methods.

Single-turn baselines. Inference-only uses the base LLM without retrieval. **Naive RAG** performs a single dense passage retrieval pass using the original question.

Graph-based retrieval. Building on advances in GNN-based structural modeling (Sun et al., 2022, 2024, 2025b,a), graph-based retrieval methods leverage GNNs or construct explicit knowledge graphs to capture relational structure and improve retrieval (He et al., 2024). **HippoRAG2** (Gutiérrez et al., 2025) augments retrieval with a personalized PageRank-based relation graph. **Log-**

Symbol	Description
Inputs	
q	The original multi-hop question
$\mathcal{C}$	The retrieval corpus
Plan	
$\mathcal{P}$	Working plan; a revisable set of requirements $\{r_1, \ldots, r_n\}$
r_i	The i-th requirement in $\mathcal{P}$, consisting of a sub-question and its dependencies
q_i	Sub-question associated with requirement r_i
Memory	
$\mathcal{M}_E$	Evidence memory: corpus-grounded evidence extracted by the Thinker
$\mathcal{M}_X$	Process memory: structured answers and satisfaction statuses recorded by the Generator Execution
Memory-grounded Failure Localization	
r_f	Failed requirement with PARTIAL_EVIDENCE or FAILED_ANSWER
ℓ	Failure report produced by the localizer; $\ell = (c, \mathcal{B}, a)$
c	Failure cause (one of four types; see below)
$\mathcal{B}$	Supporting evidence used for failure diagnosis
a	Recommended repair action conditioning the subsequent replan
Self-correcting Global Replan	
r_e	Entry point for replanning, induced from $(c, \mathcal{B})$
$A(r_e)$	Affected region: r_e and all unresolved/downstream requirements depending on r_e
Satisfaction statuses (Generator output)	
DIRECT_ANSWER	Sub-question is fully answered by the retrieved evidence
PARTIAL_EVIDENCE	Evidence is retrieved but the answer is incomplete
FAILED_ANSWER	Evidence is insufficient to answer the sub-question
Failure types (Localizer output)	
ENOUGH_TO_UPDATE	Accumulated evidence suffices to continue or slightly refine the current plan
REQUIREMENT_DEFICIENCY	The sub-question is ambiguous, underspecified, or at the wrong granularity
DECOMPOSITION_DEFICIENCY	The sub-question conflates multiple reasoning hops and must be split
UPSTREAM_ISSUE	An earlier answer is unsupported or ambiguous; the downstream failure is a propagated symptom
Recovery operators (Replanner output)	
UPDATE	Absorbs partial evidence to continue or slightly refine sub-question; no structural change
REWRITE	Reformulates the failed sub-question (adds constraints, disambiguates entities)
DECOMPOSE	Replaces the failed sub-question with a smaller dependency-aware subplan
REPLANFROMROOTCAUSE	Rolls back to r_e, invalidates $A(r_e)$, and regenerates the downstream plan

Table 7: Summary of notation used in §3.

icRAG (Chen et al., 2026) dynamically extracts logic-aware/adaptive reasoning structures. **Youtu-GraphRAG** (Dong et al., 2025) integrates graph-structured knowledge for entity-centric retrieval; following the original paper's taxonomy of agent/no-agent and open/reject modes, we adopt the **agent + open** configuration in all experiments.

Iterative retrieval. IR-CoT (Trivedi et al., 2023) interleaves chain-of-thought reasoning with retrieval steps. **Iter-RetGen** (Shao et al., 2023) alternates between retrieval and generation in multiple rounds. We implement both methods using the **FlashRAG** (Jin et al., 2025b) framework and follow its default experimental settings.

Agentic and plan-based RAG. REAP (Zhu et al., 2026) plans retrieval actions via a planner and replans when a sub-question fails. **MA-RAG** (Nguyen et al., 2025) employs a multi-agent architecture for decomposed retrieval. **RT-RAG** (Shi et al., 2026) decomposes questions and uses tree-structured retrieval with reject sampling.

RL-trained retrieval agents. Search-R1 (Jin et al., 2025a) and **R1-Searcher** (Song et al., 2025) are Qwen2.5-7B models trained with reinforcement learning to perform multi-step search.

Implementation Details. We use E5-large-v2 (Wang et al., 2022) as the embedding model for document retrieval with top-k = 5, while BGE-M3 (Chen et al., 2025) is adopted for graph-based methods to compute embeddings for non-document-level texts. Our primary base LLM is **Qwen3-Next-80B-A3B-Instruct** (Team, 2025). To evaluate generalizability across models, we additionally report results with **Qwen3.5-27B** (Qwen Team, 2026) as the base LLM.

Metrics. We report three complementary metrics in a single run. **Cover-EM** is an exact-match-with-

coverage score that marks a prediction as correct when it contains the gold answer, which makes it suitable for systems that may produce short explanatory answers rather than only a bare entity. **F1** measures token-level overlap between the prediction and the gold answer and captures partial lexical correctness. **LLM-Acc** evaluates semantic correctness with an LLM-as-judge, allowing paraphrases and equivalent expressions that string-matching metrics may penalize. Specifically, we use DeepSeek-V3.2 (DeepSeek-AI, 2025) as the judge model and follow the prompting protocol adopted in prior work (Zhu et al., 2026).

C More Results

C.1 Multi-Answer LLM-Acc Evaluation

The full multi-answer comparison is reported in Table 5 in the main text; this appendix entry is retained to preserve the existing cross-reference.

C.2 Direct Evaluation of Failure Localization

The event-level human evaluation underlying the main-text result is provided here for completeness. From each benchmark, we randomly sample 100 cases that trigger localization; a case may contain multiple localization events. Given the official supporting-fact paragraphs, human annotators judge whether (i) the predicted cause and analysis are correct, (ii) the proposed repair location is correct, and (iii) the revised plan moves in the correct direction. Table 8 shows that repair-location accuracy is consistently higher than exact cause accuracy. In particular, *UPSTREAM_ISSUE* is infrequent but obtains the highest sampled replan-success rate.

Finally, Table 9 separates structured diagnosis from both extra context and an extra LLM call. *w/o Localization* receives the same process memory, evidence memory, and failed sub-question and has the same Replanner-call budget, but directly replans without a localization report. *Pure-LLM analysis* retains a diagnosis call but removes its structured cause set and evidence checks. LoCoRAG leads both controls on all nine metrics.

Failure types and replanning frequency. Across the 2,505 localization events in the Qwen3-Next evaluation, *ENOUGH_TO_UPDATE* is the most common outcome (1,486 events; 59.32%), followed by *DECOMPOSITION_DEFICIENCY* (607; 24.23%), *UPSTREAM_ISSUE* (214; 8.54%), and *REQUIREMENT_DEFICIENCY* (198; 7.90%). Table 10 summarizes this full-event distribution, while the independently sampled human evaluation in Table 8 supplies per-type repair-success estimates.

Table 11 gives the number of replans per query. Most questions require no replan; MuSiQue triggers correction more often, with 27.5% of queries requiring at least three replans.

C.3 Comparison with Related Works

Comparison with ReAgent. ReAgent (Xinjie et al., 2025) restores earlier valid states when intermediate outputs conflict, whereas LoCoRAG diagnoses four failure causes and may modify the decomposition itself. Under the same evaluation setting, LoCoRAG outperforms ReAgent on all datasets and metrics (Table 12). This comparison should be read together with the methodological distinction: ReAgent supplies local/global backtracking, while LoCoRAG targets corpus-grounded diagnosis and cause-matched plan modification.

Scale-matched RL comparison. We compare LoCoRAG, REAP, and Search-R1 using identical Qwen2.5-7B-Instruct and Qwen2.5-32B-Instruct base-model scales. Table 13 gives a more nuanced result than the unmatched main comparison. At 7B, RL-trained Search-R1 is strongest, while LoCoRAG remains ahead of prompt-based REAP. At 32B, training-free LoCoRAG is best across all datasets and metrics. The 7B prompt-based methods are also more sensitive to instruction following and structured-output errors.

C.4 Evaluation Stability

Synthesis-prompt robustness. The main experiments follow the open synthesis setting used by several official baseline prompts, allowing parametric knowledge when retrieved evidence is incomplete. We test two alternatives: *normal*, which gives no instruction about insufficient evidence, and *reject*, which requires refusal when evidence is insufficient. Table 14 shows that performance decreases as evidence use becomes stricter, but LoCoRAG remains ahead of the compared baselines in the reject setting. Thus the main conclusion does not depend on forcing a definitive answer, although the absolute scores depend on the synthesis policy.

Repeated runs and human calibration. We repeat LoCoRAG five times under the Qwen3-Next setting. Across all nine dataset/metric combinations, the standard deviation is at most 0.42 (Ta-

Predicted type	Events	Cause correct	Location correct	Replan success
ENOUGH_TO_UPDATE	499	71.54	84.17	46.09
DECOMPOSITION_DEFICIENCY	220	39.09	80.91	35.91
UPSTREAM_ISSUE	53	67.92	84.91	60.38
REQUIREMENT_DEFICIENCY	44	40.91	77.27	38.64

Table 8: Event-level human evaluation of the localizer. All columns except Events are percentages. Replan success means that the revised plan moves in the correct direction under human judgment.

Variant	2Wiki	MuSiQue	HotpotQA
LoCoRAG	**84.6/79.75/86.7**	**52.7/52.64/63.3**	**73.9/76.71/87.2**
Pure-LLM analysis	83.3/78.61/85.6	48.5/49.03/57.9	71.3/74.24/84.8
w/o Localization	83.5/78.68/85.9	47.8/47.94/56.0	72.8/75.78/86.3

Table 9: End-to-end structured-localizer controls, reported as Cover-EM/F1/LLM-Acc.

Failure type	Events	Proportion
ENOUGH_TO_UPDATE	1,486	59.32%
DECOMPOSITION_DEFICIENCY	607	24.23%
UPSTREAM_ISSUE	214	8.54%
REQUIREMENT_DEFICIENCY	198	7.90%

Table 10: Failure-type distribution over all 2,505 localization events in the Qwen3-Next evaluation.

Dataset	0	1	2	3+
2Wiki	83.7	5.5	2.0	8.8
HotpotQA	79.1	11.6	2.7	6.6
MuSiQue	56.6	10.1	5.8	27.5
All	73.13	9.07	3.50	14.30

Table 11: Percentage of queries by number of replans.

ble 15). We also calibrate LLM-Acc against human judgments on single-answer and multi-answer subsets. Table 16 shows small judge–human gaps, and LoCoRAG retains an average 1.6-point human-evaluated lead over the best baseline.

C.5 Full-Dataset Efficiency Validation

The baseline comparison in Figure 3 uses 100 queries because several baselines are prohibitively expensive. We additionally run LoCoRAG on all 1,000 evaluation queries per dataset. Table 17 shows that the full-set per-query averages closely match the sampled estimates.

C.6 Hyperparameters Sensitivity

We further study the effect of retrieval breadth by varying the dense retriever's top-k value in $\{2, 3, 5, 10, 20\}$ while keeping all other settings identical to the main Qwen3-Next setup. Figure 4 shows that LoCoRAG is generally robust to this hyperparameter. Moving from $k = 2$ to $k = 5$ consistently improves performance, indicating that a very narrow retrieval set can miss supporting evidence needed by the Thinker–Generator execution loop. At $k = 5$, LoCoRAG reaches the best results on 2WikiMultiHopQA and remains close to the best setting on MuSiQue and HotpotQA.

Method	2Wiki	MuSiQue	HotpotQA
ReAgent	73.6/21.65/73.3	36.8/12.24/48.7	72.5/22.68/85.9
LoCoRAG	**84.6/79.75/86.7**	**52.7/52.64/63.3**	**73.9/76.71/87.2**

Table 12: ReAgent comparison, reported as Cover-EM/F1/LLM-Acc.

Increasing k beyond 5 does not yield uniform gains. On 2WikiMultiHopQA, all three metrics drop when more candidates are retrieved, suggesting that extra passages can introduce distractors into otherwise well-specified reasoning chains. MuSiQue and HotpotQA benefit slightly from larger k on some metrics, but the improvements are small relative to the additional retrieval and prompting cost. We therefore use top-k = 5 in the main experiments as a balanced setting: it provides enough evidence for memory-grounded failure localization without unnecessarily expanding the context that subsequent replanning must diagnose.

D Detailed Case Study

Case 1: *UPSTREAM_ISSUE*. This case comes from MuSiQue. The key signal is the mismatch between $\mathcal{M}_X$, which propagates *Czechoslovakia*, and $\mathcal{M}_E$, which shows that the invasion and embassy clues require different role assignments and finally point to *Yugoslavia*. LoCoRAG therefore repairs the affected upstream association rather than repeatedly rewriting the failed downstream query, demonstrating the diagnosis-determined correction

Base LLM	Method	2Wiki	MuSiQue	HotpotQA
Qwen2.5-7B	LoCoRAG	31.5/48.40/49.7	19.5/23.57/27.2	46.9/48.73/58.2
	REAP	15.1/17.02/17.6	7.5/10.37/12.0	24.3/27.05/31.6
	Search-R1	**67.3/66.95/71.9**	**37.5/45.97/51.0**	**64.1/73.06/81.0**
Qwen2.5-32B	LoCoRAG	**82.8/77.48/85.5**	**47.3/49.47/58.2**	**71.7/72.98/84.9**
	REAP	75.0/67.90/76.6	29.9/30.32/36.0	64.2/62.25/75.9
	Search-R1	69.0/68.07/74.5	38.7/46.98/52.9	62.6/72.64/81.3

Table 13: Scale-matched results, reported as Cover-EM/F1/LLM-Acc.

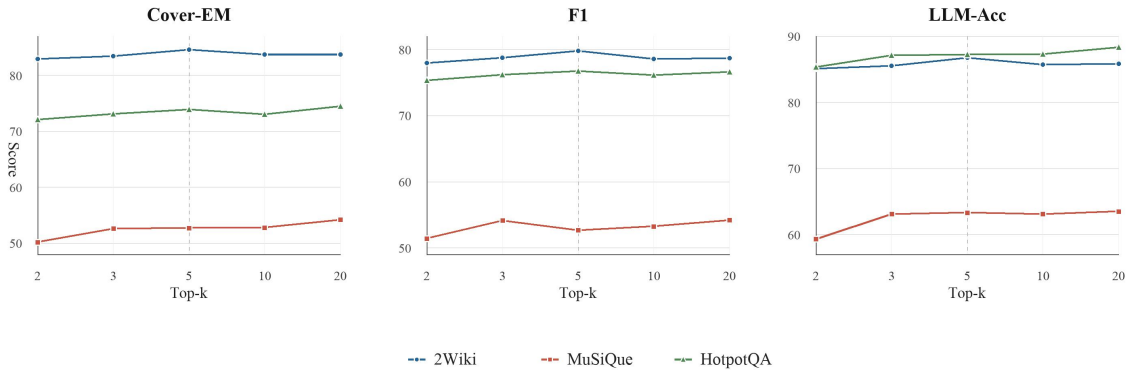

Figure 4: Sensitivity to the retriever top-k value on Qwen3-Next-80B-A3B-Instruct. Each panel reports one evaluation metric across 2WikiMultiHopQA, MuSiQue, and HotpotQA; the dashed vertical line marks the default $k = 5$ used in the main experiments.

Setting	Method	2Wiki	MuSiQue	HotpotQA
Open	REAP	76.7/48.95/76.7	34.7/32.73/43.6	66.0/56.80/79.3
	Best baseline	83.0/78.24/85.5	34.7/36.91/42.6	73.6/21.36/85.6
	LoCoRAG	**84.6/79.75/86.7**	**52.7/52.64/63.3**	**73.9/76.71/87.2**
Normal	REAP	72.7/52.96/72.2	38.7/33.85/45.8	70.4/59.27/81.7
	Best baseline	77.0/75.26/81.3	35.9/40.71/48.8	67.2/72.27/83.9
	LoCoRAG	**82.7/77.86/84.4**	**49.6/49.63/58.8**	**73.2/75.87/85.4**
Reject	REAP	66.1/53.10/67.5	27.2/25.65/31.5	59.7/54.49/70.9
	LoCoRAG	**80.3/75.07/82.0**	**47.3/45.17/54.4**	**69.9/71.45/82.5**

Table 14: Synthesis-policy robustness, reported as Cover-EM/F1/LLM-Acc. "Best baseline" is selected separately for each dataset under that policy.

	2Wiki	MuSiQue	HotpotQA
Mean	84.60/79.71/86.62	52.60/53.15/62.96	74.02/76.60/87.10
Std.	0.31/0.19/0.164	0.34/0.42/0.27	0.11/0.28/0.32

Table 15: Mean and standard deviation over five runs, reported as Cover-EM/F1/LLM-Acc.

entry point described in Section 3.

Case 2: *ENOUGH_TO_UPDATE*. This case comes from HotpotQA. The plan is sound and one branch is already grounded: $\mathcal{M}_X$ records P. G. Wodehouse as from *England*. The failed Barry Unsworth branch lacks explicit biographical evidence but does not contradict the plan, so LoCoRAG keeps the validated branch and

Setting	LoCoRAG LLM/human	Baseline LLM/human
2Wiki-single	86.7/88.1	85.5/86.4
2Wiki-multi	89.2/89.1	88.0/87.7
Hotpot-single	87.2/86.3	85.6/85.5
Hotpot-multi	88.9/88.3	87.0/85.8

Table 16: LLM-judge and human accuracy for LoCoRAG and the strongest baseline.

only slightly sharpens the current evidence-seeking query. This is the minimal correction scope in LoCoRAG: update the current action while preserving completed work.

Case 3: *DECOMPOSITION_DEFICIENCY*. This case comes from HotpotQA. The original branch expands the founder answer into direct questions about which Wanxiang subsidiary each founder headed. These queries fail because the corpus does not provide a single person–role–parent-company statement: the Spider9 passage gives each founder's prior organization, while a separate passage states that A123 Systems is a wholly owned subsidiary of Wanxiang Group. LoCoRAG identifies this as under-decomposition, preserves the founder evidence, and replaces the failed direct queries with separate affiliation and subsidiary-verification checks.

Dataset	100 queries				1,000 queries			
	Input	Output	Calls	Seconds	Input	Output	Calls	Seconds
2Wiki	13,246	1,966	10.78	20.97	13,025	2,044	11.36	21.35
HotpotQA	10,529	1,750	9.75	17.52	10,219	1,875	10.02	17.98
MuSiQue	24,886	4,337	16.70	35.52	24,243	4,052	16.40	34.68

Table 17: Average input/output tokens, LLM calls, and wall-clock seconds per query for LoCoRAG.

Table 18 shows an *UPSTREAM_ISSUE* case in a completed plan. The process memory records the validated answer *Czechoslovakia*, but the downstream requirements place it in the wrong semantic role and later attach it to the embassy clue. Because the evidence memory contains the counter-signal that the embassy evidence points to *Yugoslavia*, LoCoRAG rolls back the affected upstream association instead of rewriting only the failed query. This self-correcting replan recovers the correct bridge and yields *Paraguay*.

Table 19 shows *ENOUGH_TO_UPDATE* case to continue to search. The plan structure is already correct and $\mathcal{M}_X$ contains a supported Wodehouse answer, while the unresolved Barry Unsworth branch has only retrieved indirect literary context. LoCoRAG therefore keeps the completed branch and slightly refines the current retrieval action toward nationality and birthplace evidence. The recovered passage supports *England*, allowing the comparison to be answered directly.

Table 20 shows execution original plan facing *DECOMPOSITION_DEFICIENCY* case in one sub-query. The failed branch asks which Wanxiang subsidiary each Spider9 founder headed, but the corpus does not state this person–role–parent-company relation directly. Instead, $\mathcal{M}_E$ preserves a passage stating that A123 Systems is a wholly owned subsidiary of Wanxiang Group. LoCoRAG therefore decomposes the branch into affiliation extraction and subsidiary verification, ultimately identifying *A123 Systems* as the only answer supported by the evidence.

E Prompt Templates Used by LoCoRAG

Our core prompts include Thinker, Generator, Memory-grounded Failure Localization, Self-correcting Global Replan, and Synthesise, as illustrated in Figures 5–9.

Thinker

Role
You are an advanced Analytical Researcher. Your expertise lies in identifying non-obvious connections, contextual nuances, and latent patterns within complex datasets.

Objective
Analyze the provided ‘[Context]‘ and extract every fragment of information that could potentially be relevant to the ‘[Current Sub-question]‘ or the ‘[Original Query]‘.

Strategy: "Maximal Recall"
Your goal is to ensure NO potentially useful evidence is missed. Keep evidence that helps with the current step and evidence that may be needed for downstream hops to answer the original query.

Inclusion Criteria
Extract any text segments that fall into these categories:
1. **Direct Evidence**: Explicitly mentions or answers components of the query.
2. **Downstream Bridge Evidence**: Identifies bridge entities, candidate sets, or constraints that may be needed in later hops.
3. **Contextual Anchors**: Provides background, environmental factors, or timeframes relevant to the query’s subject.
4. **Causal/Logical Links**: Information that explains the "why" or "how" behind the query’s subject, even if the query doesn’t ask for it directly.
5. **Entity Attributes**: Descriptions of status, properties, or relationships of entities mentioned in the query.
6. **Contradictory Signals**: Data that might challenge the assumptions within the query.

Constraints
- **Dual-Query Grounding**: Consider both the current sub-question and the original query when deciding what to keep.
- **Keep Downstream-Useful Evidence**: Retain evidence that may be needed for later hops, not only evidence that directly answers the current step.
- **Keep Coordinated Facts Intact**: If the evidence contains coordinated candidates or facts, keep ALL of them before filtering. Do not keep only the first matching candidate.
- **Preserve Explicit Qualifiers**: Keep ordinals, counts, time anchors, comparison markers, and scope markers in the exact quoted span. Quote a slightly longer span when needed to preserve them.
- **Zero Paraphrasing**: Quote the text exactly as it appears in the source.
- **Granularity**: Keep fragments concise but self-contained enough to preserve the critical qualifiers.
- **Coreference-Safe Quoting**: When the relevant span begins with a pronoun or definite reference ("the company", "it", "he", "this plant"), expand the quoted span backward (within the same document) until the antecedent - a proper-noun subject or the document title - is included. Never assume a pronoun refers to an entity introduced in a different document.

Output Format
Return the results in a JSON array:
[
{
"evidence_quote": "Exact string from text",
"relevance_type": "Direct / Indirect / Contextual / Causal"
}
]
MOST Crucial Rule: Your entire output must be a single, valid JSON array. Do NOT add ANY text outside of the JSON structure.
Current Sub-question: {current_sub_query}
Original Query: {original_query}
Context: {retrieved_documents}

Final Response (JSON)

Figure 5: Thinker module prompt.

Stage	Execution trace
Original input	**Question**: "A country's military branch, the equivalent of which in the US helped the Old West's local police, was unprepared for the invasion of the country that developed the ZH-29. The unprepared country was the only communist country to have an embassy where?" **Final answer:** ***Paraguay***.
Initial plan	**Initial decomposition:** req1: "What country developed the ZH-29?"; deps: none. req2: "Which country was invaded by <answer of req1>?"; deps: req1. req3: "What is the military branch of <answer of req2> that is equivalent to a US military branch that helped the Old West's local police?"; deps: req2. req4: "Which communist country is <answer of req2>?"; deps: req2. req5: "Where did <answer of req4> have an embassy?"; deps: req4.
Successful execution before failure 1	The Thinker retrieves evidence that the ZH-29 was developed in Czechoslovakia. The Generator stores req1 $\rightarrow$ *Czechoslovakia* with DIRECT_ANSWER. This validated fact is preserved.
Failure 1	After substitution, req2 becomes "Which country was invaded by Czechoslovakia?" Yet $\mathcal{M}_E$ says the Warsaw Pact invaded Czechoslovakia. Czechoslovakia is the invaded country, not the invader, so req2 is marked FAILED_ANSWER.
Self-correction 1	The localizer assigns *UPSTREAM_ISSUE*: the correct req1 answer was propagated with the wrong role. Question decomposition and update for req2 is wrong. LoCoRAG keeps req1 and replans the affected region: req1: completed, answer *Czechoslovakia*. req2_new: pending, "Which country was invaded by the Warsaw Pact forces in 1968?"; deps: req1. req3_new: pending, "What is the military branch of the invaded country (Czechoslovakia) that is equivalent to a US military branch that helped the Old West's local police?"; deps: req2_new. req4_new: pending, "Was Czechoslovakia a communist country?"; deps: req2_new. req5_new: pending, "Where did Czechoslovakia, as the only communist country with an embassy in that location, have an embassy?"; deps: req4_new.
Successful execution after correction 1	**Executed corrected plan:** req2_new: DIRECT_ANSWER, answer *Czechoslovakia*. req4_new: DIRECT_ANSWER, answer "Yes, Czechoslovakia was a communist country from 1948 until 1989." req3_new: DIRECT_ANSWER, answer *Czechoslovak Army*. Remaining unresolved branch: req5_new, which now depends on the locally true but downstream-sensitive entity association in req4_new.
Failure 2	req5_new asks for Czechoslovakia's embassy location. Retrieval instead says Yugoslavia was the only communist country allowed to have an embassy in Alfredo Stroessner's Paraguay. $\mathcal{M}_X$ points to Czechoslovakia, while $\mathcal{M}_E$ points to Yugoslavia; req5_new is marked FAILED_ANSWER.
Self-correction 2	The localizer again assigns *UPSTREAM_ISSUE*. req4_new is true in isolation but selects the wrong entity for the embassy relation. LoCoRAG replans from that upstream association: req1: completed, answer *Czechoslovakia*. req2: invalidated, original role-misaligned invasion requirement. req2_new: completed, answer *Czechoslovakia*. req3_new: completed, answer *Czechoslovak Army*. req4_new: invalidated, "Was Czechoslovakia a communist country?" req5_new: invalidated, "Where did Czechoslovakia, as the only communist country with an embassy in that location, have an embassy?" req4_new_corrected: pending, "Which communist country had an embassy in Paraguay during the Cold War?"; deps: none.
Successful execution after correction 2	**Final corrected plan state:** req1: completed, answer *Czechoslovakia*. req2: invalidated. req2_new: completed, answer *Czechoslovakia*. req3_new: completed, answer *Czechoslovak Army*. req4_new: invalidated. req5_new: invalidated. req4_new_corrected: DIRECT_ANSWER, answer *Yugoslavia*.
Final synthesis	The ZH-29 was developed in Czechoslovakia. Czechoslovakia was a communist country from 1948 to 1989. The US Army, equivalent to Czechoslovakia's military branch, helped in Old West policing. Czechoslovakia was invaded by the Warsaw Pact in 1968. However, the only communist country to have an embassy in Paraguay during the Cold War was Yugoslavia, not Czechoslovakia. Since the question asks for the country that the only communist country with an embassy in a specific location, and since Yugoslavia is the only communist country confirmed to have had an embassy in Paraguay, the unprepared country must be Yugoslavia. Final answer: **Paraguay**.

Table 18: Complete corrective trace for the LoCoRAG case study. The table preserves the full sequence of consequential pipeline states while omitting retrieval scores and duplicate log records that do not affect the correction decision.

Stage	Execution trace
Original input	**Question**: "Are both Barry Unsworth and P. G. Wodehouse from the same country?" **Final answer:** ***Yes, both Barry Unsworth and P. G. Wodehouse are from England***.
Initial plan	**Initial decomposition:** req1: "What country is Barry Unsworth from?"; deps: none. req2: "What country is P. G. Wodehouse from?"; deps: none. req3: "Are <answer of req1> and <answer of req2> the same country?"; deps: req1, req2.
Successful execution before failure	req2 succeeds with direct biographical evidence: Wodehouse is described as an English author and as born in Guildford. LoCoRAG stores req2 $\rightarrow$ *England* with DIRECT_ANSWER.
Failure	req1 retrieves documents about Barry Unsworth's novels, publishers, and prizes, but none explicitly states his nationality, country of origin, or birthplace. The Generator marks req1 as FAILED_ANSWER and explicitly avoids using parameter knowledge about Unsworth's nationality.
Self-correction	The localizer assigns *ENOUGH_TO_UPDATE*: the decomposition is sound and the Wodehouse branch is complete, but the Barry Unsworth query is too broad. LoCoRAG keeps the solved branch and updates only the unresolved action: req1: pending, "Barry Unsworth nationality country of origin birthplace." req2: completed, answer *England*. req3: pending, "Are <answer of req1> and <answer of req2> the same country?"
Successful execution after correction	The refined search retrieves the Barry Unsworth biography: "Unsworth was born on 10 August 1930 in Wingate, a mining village in County Durham, England." The Generator updates req1 $\rightarrow$ *England* with DIRECT_ANSWER.
Final synthesis	With req1=*England* and req2=*England*, req3 becomes "Are England and England the same country?" LoCoRAG synthesizes the answer: **Yes, both Barry Unsworth and P. G. Wodehouse are from England**.

Table 19: Corrective trace for the HotpotQA *ENOUGH_TO_UPDATE* case. LoCoRAG preserves the completed branch and refines only the failed retrieval action.

Stage	Execution trace
Original input	**Question**: "Spider9 was founded in 2011 by the head of which subsidiary of Wanxiang Group?" **Final answer:** ***A123 Systems***.
Initial plan	**Initial decomposition:** req1: "Who founded Spider9 in 2011?"; deps: none. req2: "Which subsidiary of Wanxiang Group was <answer of req1> the head of?"; deps: req1.
Successful execution before failure	req1 succeeds. The Spider9 document states that the company was founded in 2011 by Glynne Townsend (A123 Systems), Dave Park (former Vice President of Production at Wave Crest Energy Systems), Dave Smith (former chairman of USABC), and Bill Beckman (former Vice President of Finance at Johnson Controls). The planner then expands req2 into four direct candidate queries: req2a: "Which subsidiary of Wanxiang Group was Glynne Townsend the head of?" req2b: "Which subsidiary of Wanxiang Group was Dave Park the head of?" req2c: "Which subsidiary of Wanxiang Group was Dave Smith the head of?" req2d: "Which subsidiary of Wanxiang Group was Bill Beckman the head of?"
Failure	The direct candidate queries fail because the retrieved evidence does not contain a single sentence linking a founder to the phrase "head of a Wanxiang Group subsidiary." Instead, $\mathcal{M}_E$ separately contains founder affiliations from the Spider9 document and a different document stating that A123 Systems is a wholly owned subsidiary of Wanxiang Group.
Self-correction	The localizer assigns *DECOMPOSITION_DEFICIENCY*: the failed requirements skipped the intermediate bridge from founder affiliation to subsidiary verification. LoCoRAG keeps req1, invalidates the direct queries, and decomposes the bridge: req1: completed, founders and affiliations identified. req2a_old–req2d_old: invalidated direct "which subsidiary" queries. req2a_part1: pending, "Was A123 Systems a subsidiary of Wanxiang Group at the time Glynne Townsend was associated with it?" req2b_part1: pending, "Was Wave Crest Energy Systems a subsidiary of Wanxiang Group at the time Dave Park was associated with it?" req2c_part1: pending, "Was USABC a subsidiary of Wanxiang Group at the time Dave Smith was associated with it?" req2d_part1: pending, "Was Johnson Controls a subsidiary of Wanxiang Group at the time Bill Beckman was associated with it?"
Successful execution after correction	The decomposed plan verifies the positive bridge for A123 Systems: the retrieved evidence states that A123 Systems is a wholly owned subsidiary of Wanxiang Group, and the Spider9 evidence links Glynne Townsend to A123 Systems. The other candidate organizations—Wave Crest Energy Systems, USABC, and Johnson Controls—do not receive supporting evidence as Wanxiang subsidiaries. LoCoRAG later normalizes the plan into an explicit two-step form: req_new_1: completed, extract each founder's affiliated organization: Glynne Townsend $\rightarrow$ A123 Systems; Dave Park $\rightarrow$ Wave Crest Energy Systems; Dave Smith $\rightarrow$ USABC; Bill Beckman $\rightarrow$ Johnson Controls. req_new_2a: DIRECT_ANSWER, A123 Systems is a subsidiary of Wanxiang Group. req_new_2b: no supporting evidence that Wave Crest Energy Systems is a Wanxiang subsidiary. req_new_2c: no supporting evidence that USABC is a Wanxiang subsidiary. req_new_2d: no supporting evidence that Johnson Controls is a Wanxiang subsidiary.
Final synthesis	LoCoRAG combines the verified positive path, Glynne Townsend $\rightarrow$ A123 Systems and A123 Systems $\rightarrow$ Wanxiang Group subsidiary, while the other founder-affiliation branches lack evidence for the subsidiary relation. It returns **A123 Systems**.

Table 20: Corrective trace for the HotpotQA *DECOMPOSITION_DEFICIENCY* case. LoCoRAG repairs an under-decomposed relation by separating founder-affiliation extraction from subsidiary verification.

Generator

You extract evidence-grounded facts that answer a sub-query in a multi-hop QA pipeline.
Inputs
- original_query: the overall multi-hop question; use it to preserve the final answer slot, constraints, and cardinality
- current_sub_query: the question you must answer now
- extracted_evidence: candidate quotes from retrieved documents
- requirement_id: identifier to attach to each output fact

Rules
1. Ground every conclusion in the evidence first. ‘direct_evidence‘ must contain exact substrings of ‘extracted_evidence‘ only - never paraphrases, never internal knowledge.
2. No internal-knowledge DIRECT_ANSWER: internal knowledge may be mentioned only as a missing-evidence warning or search hint. It must NOT be used to produce a DIRECT_ANSWER or to introduce facts absent from the evidence.
3. Compare the current sub-query answer slot with the original query’s final answer slot. If answering the current sub-query would produce a type that cannot be the final answer, treat it as a bridge fact and do not overstate it as the final answer.
4. When evidence offers multiple granularities, choose the value that best matches the sub-query’s answer slot. Do not always choose the most specific value if the question asks for a broader type/category.
5. When sources conflict, prefer the one whose subject and relation match the sub-query entity exactly; explain the choice in ‘reasoning‘.
6. For relation-heavy evidence, explicitly track subject, relation, object, and the queried side.
7. For candidate-list problems, a FAILED_ANSWER for one candidate does not negate another candidate with DIRECT_ANSWER. Preserve directly supported candidates for final synthesis.
8. Answer only the current sub-query. If multiple distinct evidence-grounded facts answer it, emit one entry per fact. If the original question is singular and the evidence contains multiple weak candidates, mark unsupported candidates as PARTIAL_EVIDENCE, not DIRECT_ANSWER.
9. Set ‘fulfillment_level‘:
- DIRECT_ANSWER - evidence states the answer explicitly.
- PARTIAL_EVIDENCE - evidence is on-topic but does not fully resolve the answer or requires internal knowledge/inference to finish.
- FAILED_ANSWER - evidence does not anchor the target, or internal knowledge is uncertain.

Output
Return one valid JSON object only - no prose, no markdown fences.
‘‘‘json
{{
"reasoned_facts": [
{{
"reasoning": "How the evidence supports the conclusion; include conflict resolution, answer-slot checks, and relation-role checks when relevant.",
"direct_evidence": ["exact quote 1", "exact quote 2"],
"statement": "Concise factual answer to the sub-query.",
"fulfills_requirement_id": "{requirement_id}",
"fulfillment_level": "DIRECT_ANSWER | PARTIAL_EVIDENCE | FAILED_ANSWER"
}}
]
}}
‘‘‘
Original Query: {original_query}
Current Sub-query: {current_sub_query}
Requirement ID: {requirement_id}
Extracted Evidence: {extracted_evidence}

Final Response (JSON)

Figure 6: Generator module prompt.

Memory-grounded Failure Localization

Multi-hop Query Pipeline Problem Classification
You are an expert analyst tasked with diagnosing failures in multi-hop query pipelines. Given an original query, completed requirements with their answers/evidence, and failed requirements, determine the root cause of the failure.

Decision Rules
Choose **EXACTLY ONE** category:
Analysis Type	When to Choose	Key Indicators
ENOUGH_TO_UPDATE	A completed requirement with PARTIAL_EVIDENCE can provide sufficient evidence to continue the original plan. The failure is superficial and can be resolved by updating the search strategy or rephrasing the query.	
- Completed requirements contain relevant information - The search direction is correct - Minor adjustments would likely succeed - Evidence exists but wasn't properly utilized		
UPSTREAM_ISSUE	A completed requirement produces output that is incorrect or misaligned in entity, level, or scope, causing the downstream query to target the wrong object or interpretation.	- Earlier requirement returned wrong entity/concept - Mismatch in granularity (e.g., city vs. country) - Scope drift from original intent - There're multiple correct answer to past question
REQUIREMENT_DEFICIENCY	The failed requirement itself is ill-defined, conflates distinct concepts, or omits constraints required by the original query, making it impossible to determine a correct answer even with correct upstream inputs.	- Ambiguous or vague question formulation - Multiple concepts merged inappropriately - Missing critical constraints from original query - Unclear what constitutes a valid answer
DECOMPOSITION_DEFICIENCY	The failed requirement is overly complex and implicitly combines multiple retrieval or reasoning steps into a single question, requiring further decomposition before it can be correctly answered.	-Question contains multiple implicit sub-steps - Requires sequential reasoning not expressible in one query -Combines retrieval + computation + filtering - Too broad to answer atomically

Analysis Process
1. **Trace the dependency chain**: Examine how the failed requirement depends on completed requirements
2. **Evaluate upstream quality**: Check if completed requirements provided correct and relevant information
3. **Assess requirement clarity**: Determine if the failed requirement is well-defined and answerable
4. **Check complexity**: Evaluate if the failed requirement needs further breakdown

Output Format
Return **ONLY** valid JSON in this exact structure:
```json
{{
"analysis_type": "<one of: ENOUGH_TO_UPDATE | UPSTREAM_ISSUE | REQUIREMENT_DEFICIENCY | DECOMPOSITION_DEFICIENCY>",
"reasoning": "<2-3 sentences explaining why this classification was chosen>",
"evidence": {{
"upstream_analysis": "<brief assessment of completed requirements quality>",
"failed_requirement_analysis": "<brief assessment of the failed requirement itself>",
"dependency_impact": "<how upstream affects the failed requirement>"
}},
"recommendation": "<specific action to resolve the issue>"
}}
```
MOST Crucial Rule: Your entire output must be a single, valid JSON object. Do NOT add ANY text outside of the JSON structure.

Original Multi-hop Query
{original_query}

Completed Requirements (with answers/evidence)
{completed_query}

Failed Requirement(s)
{failed_requirement_query}
Now analyze the provided input and return ONLY the JSON output.

Figure 7: Memory-grounded failure localization prompt.

Self-correcting Global Replan

Multi-hop Query Recovery and Replanning
You are an expert query planner responsible for recovering from failures in multi-hop query pipelines. Based on the problem analysis, you will generate a new execution plan using one of three recovery strategies.

Recovery Strategies
Based on the ‘analysis_type‘ from the problem analysis, choose one of three recovery strategies:
Strategy 1: REWRITE
When to use: ‘REQUIREMENT_DEFICIENCY‘
Action: Modify the failed query itself to fix ambiguities, add missing constraints, or refine the search approach.

Strategy 2: DECOMPOSE
When to use: ‘DECOMPOSITION_DEFICIENCY‘
Action: Break down the failed query into 2 sequential sub-questions that together answer the original requirement.

Strategy 3: REPLAN_FROM_ROOT_CAUSE
When to use: ‘UPSTREAM_ISSUE'
Action: Identify the earliest incorrect upstream requirement, invalidate all requirements that depend on it, and rebuild the plan from that point.

Output Format
Return **ONLY** valid JSON in this exact structure:
“‘json
{
"analysis": {"problem_diagnosis": "" "recovery_strategy": "", "strategy_rationale": "" },
"decision": { "next_step": "", "updated_plan": [{ "requirement_id": "", “question": "", "depends_on": [""], "status": "", "answer": ""], "next_actions": [{ "requirement_id": "",
"question": "", "reasoning": "" }] }
}
“‘
Field Specifications
analysis section:
- **problem_diagnosis**: 2-3 sentences explaining the root cause and its impact
- **recovery_strategy**: One of: ‘REWRITE‘, ‘DECOMPOSE‘, ‘REPLAN_FROM_ROOT_CAUSE‘
- **strategy_rationale**: Brief explanation of why this strategy fits the problem
  decision section:
- **next_step**:
- ‘SYNTHESIZE_ANSWER‘ if all requirements are now satisfied
- ‘CONTINUE_SEARCH‘ if more queries are needed
- **updated_plan**: Complete plan including:
- Previously completed requirements (with status="completed" and their answers)
- New/modified pending requirements (with status="pending")
- Invalidated requirements if using REPLAN_FROM_ROOT_CAUSE (with status="invalidated")

Strategy Selection Examples
Example 1 - REWRITE:
- Query was too vague: "find information about the company" → "find the founding year of Tesla Inc."
- Missing constraints: "search for reviews" → "search for customer reviews of iPhone 15 Pro from 2024"
  Example 2 - DECOMPOSE:
- Too complex: "Compare the GDP growth and inflation rates of top 5 economies"
- Split into: req_3a: "Find GDP growth rates of top 5 economies", req_3b: "Find inflation rates for the countries identified in req_3a"
  Example 3 - REPLAN_FROM_ROOT_CAUSE:
- req_1 incorrectly identified "Apple Inc." when user meant "Apple Records"
- req_2, req_3 all based on wrong company → invalidate all
- Create new req_1b with correct search, rebuild from there

Original Multi-hop Query
{original_query}

Original Plan
{original_plan}

Failed Requirement(s)
{failed_requirement}

Problem Analysis
{problem_analysis}
Now process the provided input and generate the recovery plan as JSON output.

Figure 8: Self-correcting global replanning prompt.

Synthesise

You are an "Answer Synthesizer" AI. Your sole purpose is to reason over the available facts and generate a definitive answer to the user's original question.
Your Task:
1. **Anchor on the Goal:** Read the 'original_user_question' and hold it as your single most important objective.
2. **Filter the Facts:** Scrutinize the 'reasoned_facts'. **Strictly use only facts that are directly relevant to answering the 'original_user_question'.**
3. **Reason First:** Produce a concise reasoning process that explains how the relevant facts lead to the answer.
4. **Synthesize the Answer:** Using the filtered facts, construct a single, coherent answer. **You MUST provide an answer based on available information - uncertainty is not an option.**
Crucial Rules:
- **BE DECISIVE:** You must always reach a conclusion. When direct evidence is incomplete, draw on your own parametric knowledge, domain expertise, and logical reasoning to fill the gaps. Make the most reasonable determination based on all available information - both provided context and what you already know.
- **BE CONCISE:** Answer should be concise. The answer should be concise while must preserve the original phrasing from the retrieved facts - do not over-compress.
- **BE DIRECT:** The reasoning should focus only on the logic needed to reach the answer. The answer should directly state the conclusion.
- **NO IRRELEVANT DETAILS:** Exclude any information that doesn't directly contribute to answering the specific question.
- **COMMIT TO AN ANSWER:** Even with limited information, provide the most likely answer rather than stating "I don't know " or "It's unclear". Use your best judgment based on what you have.
- **JSON ONLY:** Return valid JSON only. Do not add markdown, code fences, or any extra text.
Output Format:
```json
{
"reasoning": "<concise reasoning process grounded in the collected facts>",
"answer": "<final answer>"
}
```
MOST Crucial Rule: The output must match the JSON schema above exactly. Always deliver a definitive answer, and ensure the answer type matches the original question's answer type. If the final answer has multiple items, list all of them inside the 'answer' field.
Original User Question:
{query}
Collected Facts:
{facts}

Figure 9: Synthesis stage prompt.