DOCTOR-RAG: A FAILURE-AWARE REPAIR FRAMEWORK FOR AGENTIC RETRIEVAL-AUGMENTED GENERATION

Anonymous authors
Paper under double-blind review

ABSTRACT

Agentic Retrieval-Augmented Generation interleaves retrieval and reasoning for multi-hop QA and complex knowledge tasks. As reasoning trajectories lengthen, failures become more frequent, while existing methods often either stop at diagnosis or rely on coarse replanning and rerun-style recovery, incurring high computational cost. We propose **Doctor-RAG (DR-RAG)**, a diagnose-and-repair framework that corrects failures via explicit error localization and prefix reuse. DR-RAG operates in two stages: (i) trajectory-level failure diagnosis, where a distilled diagnosis model jointly assesses evidence sufficiency, classifies the failure type, and localizes the earliest failure point; and (ii) tool-conditioned local repair that intervenes only at the diagnosed point while reusing conditionally valid prefixes and retrieved evidence. By separating error attribution from correction, DR-RAG avoids blind reruns in a post-hoc repair setting and enables targeted, efficient correction of known failed trajectories. Experiments on three multi-hop QA benchmarks across multiple agentic RAG baselines and backbone models show substantial improvements in answer accuracy. The anonymized code and data are available here.

1 INTRODUCTION

In recent years, Agentic Retrieval-Augmented Generation (Agentic RAG) has emerged as an important paradigm for multi-hop question answering and complex knowledge reasoning (Yao et al., 2023b; Jin et al., 2025; Song et al., 2025). By dynamically generating search queries, invoking retrievers, and performing multi-step reasoning during inference, Agentic RAG enables models to progressively plan and solve complex problems, improving evidence acquisition and multi-step problem-solving over static RAG systems (Shi et al., 2025; Singh et al., 2025; Luo et al., 2025; Zhang et al., 2025).

However, as reasoning chains become longer and interaction rounds increase, failures in Agentic RAG become more frequent and difficult to avoid (Agrawal et al., 2024; Leung et al., 2026; Ru et al., 2024). Since errors in multi-step reasoning may arise at arbitrary stages and accumulate over time, simply improving single-step reasoning or retrieval performance is insufficient to ensure stable system behavior (Dong et al., 2025c; Yan et al., 2024; Asai et al., 2024). Therefore, the challenges faced by Agentic RAG extend beyond whether a correct answer can be generated, and increasingly concern:

> ***How can an Agentic RAG system localize the cause of a failure and repair it with minimal recomputation once the failure occurs?***

Existing studies have analyzed and modeled failures in RAG systems from different perspectives (Ru et al., 2024; Agrawal et al., 2024; Leung et al., 2026; Cao et al., 2025; Guinet et al., 2024). One line of work focuses on systematically categorizing and diagnosing errors in RAG outputs (Ru et al., 2024; Yehudai et al., 2025), but does not further address automatic repair or performance improvement after failures occur. Another line of work uses diagnostic feedback to guide agentic repair (Dong et al., 2025c), but typically follows a replanning–rerun strategy that discards the original trajectory and re-executes the retrieval–reasoning pipeline. These limitations suggest that failure analysis must be coupled with fine-grained recovery mechanisms that can selectively intervene in an existing trajectory

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

rather than regenerate it from scratch. Figure 6 in Appendix D.11 further illustrates this issue: failure types under the ReAct baseline on HotpotQA arise from diverse sources rather than a single dominant cause, making unconditional reruns or one-size-fits-all repair strategies inherently inefficient.

To address this problem, we propose **Doctor-RAG (DR-RAG)**, a unified diagnose-and-repair framework for Agentic RAG, designed to enable efficient failure correction through explicit error attribution and minimal-cost intervention. Unlike existing approaches that directly retry or replan the entire retrieval–reasoning pipeline, DR-RAG explicitly models failure correction as two consecutive stages: error diagnosis and conditional repair.

DR-RAG consists of two tightly coupled components: (1) **Taxonomy-Constrained Error Diagnosis and Localization.** When a reasoning failure occurs, DR-RAG treats the full agentic RAG trajectory as the object of diagnosis. A distilled diagnosis model jointly assesses evidence sufficiency, classifies the failure type under a sufficiency-aware taxonomy, and localizes the earliest failure point. This identifies the first reasoning or retrieval step that deviates from a valid solution path, providing structured signals for minimal intervention. (2) **Tool-Conditioned Local Repair.** Given the diagnosed error type and failure location, DR-RAG selects a repair operator that intervenes only at the failure point, while maximizing reuse of validated reasoning prefixes and retrieved evidence. Repair operations include lightweight answer rewriting for Format Error, evidence-preserving re-reasoning under full coverage, query rewriting and retrieval enhancement for Retriever Error, and planning-based repair for Search Error. By constraining repair to necessary components, DR-RAG reduces the cost of retrieval and reasoning while preserving coherence across repaired trajectories.

Our main contributions are as follows:

- We are the first to formulate failure handling in Agentic RAG at the system level, modeling correction as diagnosis-guided minimal-cost intervention rather than full-pipeline retry.
- We propose a sufficiency-aware error taxonomy with a distilled diagnosis model that jointly performs evidence sufficiency assessment, failure classification, and earliest-step localization.
- We design tool-conditioned repair operators that adapt repair actions to diagnosed error types, enabling localized correction with prefix and evidence reuse.
- Experiments across multiple datasets, agentic RAG baselines, and backbone models show that DR-RAG achieves state-of-the-art repair rates while consistently improving downstream answer accuracy.

2 Related Work

Retrieval-Augmented Generation. LLMs are prone to hallucinations and outdated knowledge; RAG mitigates this by integrating external knowledge at inference time, improving factual accuracy without retraining (Guu et al., 2020; Karpukhin et al., 2020; Jiao et al., 2026; Huang et al., 2025a). The retriever–generator framework (Lewis et al., 2020) has been extended with advanced reasoning and retrieval, including Chain-of-Thought (Yao et al., 2023b), Tree-of-Thought (Yao et al., 2023a; Jiao et al., 2026), and adaptive retrieval (Jeong et al., 2024) for effective multi-step reasoning.

Agentic RAG. Agentic RAG (e.g., ReAct (Yao et al., 2023b), Search-o1 (Li et al., 2025a), Search-R1 (Jin et al., 2025)) unifies reasoning and tool use—retrieval (Dong et al., 2025b), memory (Li et al., 2025b), summarization (Dai et al., 2025)—as sequential decision-making. Policy learning (Dong et al., 2025d;a) enables autonomous action selection; prior work (Wu et al., 2025; Chen et al., 2025; Gao et al., 2025) optimizes policies via outcome-level rewards or preferences. Recent robustness studies include RAGChecker (Ru et al., 2024) (evaluation-only) and RAG-Critic (Dong et al., 2025c) (trajectory replanning), but early error localization and prefix reuse remain underexplored, motivating our structured diagnosis and selective trajectory repair approach.

Step-Level Online Verification and Repair. Step-level online verification methods (Lightman et al., 2024; Xiong et al., 2024; Zhou et al., 2024) score intermediate actions or states during trajectory construction, enabling immediate correction but requiring access to the agent's internal execution loop. This makes them difficult to apply to black-box or closed-source agentic RAG systems, while frequent step-wise intervention may disrupt long-horizon reasoning and incur substantial per-step computation. In contrast, DR-RAG avoids online step-wise intervention by diagnosing completed

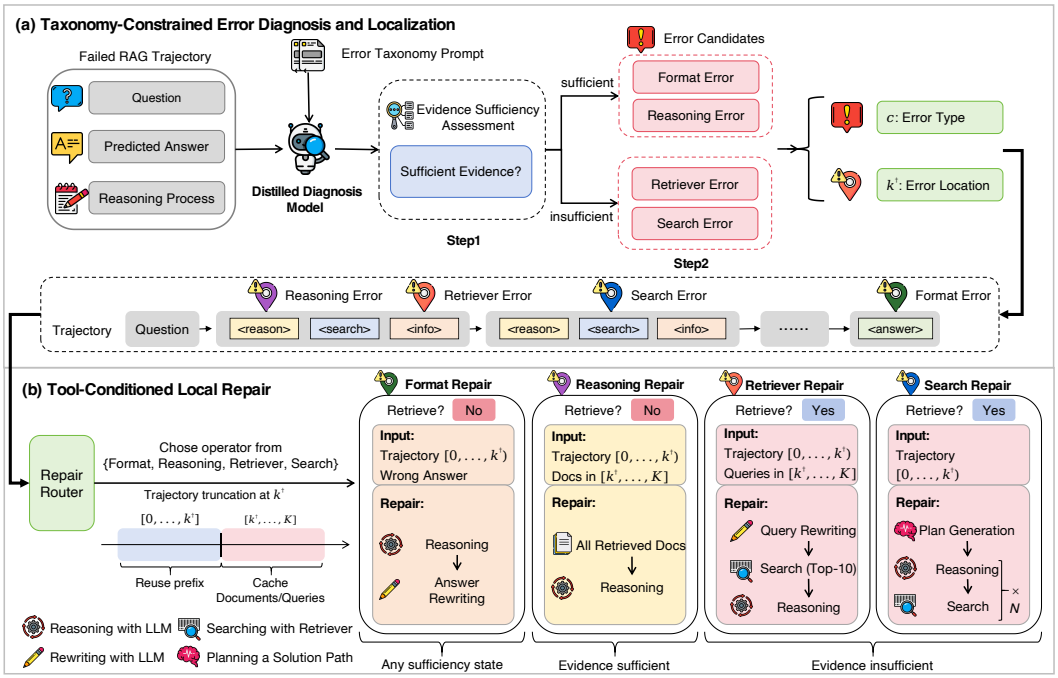


Figure 1: Overview of DR-RAG. Given a failed agentic RAG trajectory, the diagnosis model jointly assesses evidence sufficiency and performs error classification and localization. Finally, tool-conditioned local repair intervenes at the localized failure point while reusing conditionally valid prefixes and retrieved evidence.

failed trajectories and applying localized repair with reuse of valid prefixes and evidence, making it compatible with black-box agentic RAG backbones while reducing repeated verification overhead.

3 Agentic RAG Task Formulation

We consider multi-step question answering under agentic retrieval-augmented generation (RAG), where a language model alternates between **Reason** (r), **Search** (q), and **Information** (d) actions.

Given a question $x \in \mathcal{X}$, the system produces an action trajectory:

$$y = \{\rho^{(1)}, \rho^{(2)}, \ldots, \rho^{(K)}, a_{\text{pred}}\}, \tag{1}$$

where $\rho^{(k)} \in \{r, q, d\}$ and a_{pred} is the final answer (treated as a terminal action).

Actions typically interleave, and the trajectory length and structure may vary depending on intermediate reasoning outcomes, especially after localized repair, as illustrated in Figure 1(a).

4 Methodology

4.1 DR-RAG Overview

Figure 1 illustrates DR-RAG, a diagnose-and-repair framework for agentic RAG. Unlike prior approaches that rerun the full reasoning–retrieval pipeline, DR-RAG treats failed trajectories as reusable objects and corrects them via targeted, localized interventions.

The framework operates in two stages: (1) error diagnosis and localization, and (2) tool-conditioned local repair. During diagnosis, DR-RAG jointly assesses evidence sufficiency and identifies the error type and earliest failure point. In repair, a specialized operator intervenes only at the failure point, reusing all previously validated prefix steps.

Error Type	Format Error	Reasoning Error	Retriever Error	Search Error
Evidence	—	Sufficient	Insufficient	Insufficient
Localized Component	answer	reasoning	information	search

Table 1: Error taxonomy for agentic RAG trajectories with localized failure attribution. Evidence sufficiency is assessed implicitly by the diagnosis model.

4.2 Diagnosis Module

The diagnosis module analyzes failed agentic RAG trajectories and outputs a structured diagnosis to condition downstream repair. It takes as input a failed action-level trajectory y produced by the underlying agentic RAG system, where y contains reasoning steps, search queries, retrieved evidence blocks, observations, and a final predicted answer a_{pred}. The trajectory is considered failed when the prediction is incorrect, i.e., $\text{EM}(a_{\text{pred}}, a^\star) = 0$.

The goal of diagnosis is to determine both *why* the trajectory fails and *where* the failure first occurs. Formally, the module outputs a tuple $(c, k^\dagger)$, where c denotes the predicted error type and $k^\dagger$ denotes the earliest localized failure point. This structured diagnosis serves as the control signal for tool-conditioned local repair.

4.2.1 Error Taxonomy

Table 1 summarizes the error taxonomy used by DR-RAG. Each error type captures a repairable failure mode in the retrieval–reasoning process and restricts the downstream repair operator to a targeted intervention. The taxonomy separates failures according to the earliest component that makes the trajectory unrecoverable without intervention.

Format Error captures cases where the final answer is semantically correct but fails exact match due to surface-form differences. **Reasoning Error** captures cases where sufficient evidence has been retrieved, but the model derives an incorrect answer or intermediate conclusion. **Retriever Error** captures cases where a valid search intent fails to retrieve the required evidence. **Search Error** captures cases where the search itself is misdirected due to flawed reasoning, such as using incorrect entities, unsupported assumptions, wrong search directions, or overly vague queries. Detailed definitions and decision criteria for each error type are provided in Appendix E.5.

4.2.2 Decision Protocol and Localization

The diagnosis follows a hierarchical protocol. First, the model checks for **Format Error**, where the final answer is semantically correct but violates the exact-match format. If the answer is incorrect, the model then assesses whether the retrieved evidence is sufficient to answer the question. When the evidence is sufficient, the failure is classified as **Reasoning Error** and localized to the first reasoning step that deviates from a valid solution path.

When the evidence is insufficient, the model further distinguishes **Retriever Error** from **Search Error** through query-level validity judgment. A search query is considered *valid* if it uses correct entities from the question or prior evidence, targets relevant information, and is sufficiently specific. If such a valid query fails to retrieve the necessary evidence, the error is attributed to a Retriever Error. In contrast, a query is treated as *poisoned* if it is caused by flawed reasoning, such as hallucinated entities, unsupported assumptions, wrong search directions, or excessive vagueness. If missing evidence results from poisoned queries, the failure is classified as a Search Error. If multiple errors occur along the same trajectory, the earliest occurring error determines the localized error type.

Localization is performed at the action level. For Format Error, the failure point is the final answer step. For Reasoning Error, it is the first reasoning step that produces an unsupported or incorrect intermediate conclusion despite sufficient evidence. For Retriever Error, it is the evidence block following a valid query that fails to contain the necessary information. For Search Error, it is the first poisoned search action—i.e., the earliest search query misdirected by flawed upstream reasoning. Actions preceding $k^\dagger$ are treated as conditionally valid and can be reused during repair, while actions after $k^\dagger$ are considered unreliable. Thus, the predicted error type c determines which repair operator

to invoke, and the localized point $k^\dagger$ determines where local repair starts. More detailed steps and the diagnostic prompt template are provided in Appendix E.

4.2.3 Evidence Sufficiency in Diagnosis

Evidence sufficiency is a key intermediate judgment in the above decision protocol, rather than an independent prediction module. Instead of relying on a separate coverage predictor, DR-RAG estimates evidence sufficiency within the reasoning process of the diagnosis model. After excluding format-invalid answers, the diagnostic prompt instructs the model to inspect the trajectory's `[Evidence]` blocks and determine whether they contain the necessary facts to support the correct answer. Sufficient evidence indicates that the failure lies in how the model uses the evidence, leading to Reasoning Error. Insufficient evidence indicates that the trajectory lacks necessary facts, and the diagnosis further determines whether this is caused by a retriever-side miss or by a reasoning-induced bad query. This integrated design allows the diagnosis model to jointly reason about evidence sufficiency, error type, and failure location in a single inference pass, avoiding error propagation from an external coverage estimator.

4.2.4 Distilled Diagnosis Model

We train the diagnosis module via supervised fine-tuning (SFT). A high-capacity teacher model (Qwen3.5-Plus in thinking mode) Team (2026) annotates failed trajectories with structured diagnostic reasoning and compact labels covering all four error types. The student model (Qwen3-8B (Yang et al., 2025), LLaMA-3.1-8B-Instruct (Team, 2024)) is trained to reproduce this reasoning in `<think>` tags and output a JSON object specifying the predicted error type c and earliest failure point $k^\dagger$. The teacher and student use the same prompt structure; the teacher's advantage lies in its stronger reasoning capability rather than additional information access.

4.3 Repair Module

Given the diagnosis output $(c, k^\dagger)$ from the diagnosis module, the repair stage performs intervention on the failed agentic RAG trajectory. Rather than restarting the reasoning–retrieval process, DR-RAG applies a *tool-conditioned repair operator* selected according to the diagnosed error type. This enables partial reuse of conditionally valid trajectory prefixes and avoids redundant computation.

Formally, we define a set of repair operators: $\mathcal{F} = \{f_1, \ldots, f_L\}$, where each operator f_ℓ is specialized for repairing an error. Operator selection is diagnosis-driven: $f = \mathcal{S}(c)$, where $\mathcal{S} : \mathcal{C} \to \mathcal{F}$ maps an error type c to its repair operator.

4.3.1 Repair under Sufficient Evidence

When sufficient evidence is confirmed, repair corrects reasoning or answer generation without further retrieval.

Format-Invalid Answer Repair. For format errors, the trajectory and retrieved documents are considered valid. Repair is limited to rewriting the final answer to meet task-specific constraints. The model is prompted with the original question, retrieved documents, and formatting instructions, and asked to regenerate the answer. No reasoning or retrieval is changed.

Reasoning Logic Error Repair. For reasoning errors, the failure arises from incorrect reasoning over sufficient evidence. The trajectory is truncated at the localized error index $k^\dagger$, retaining the prefix:

$$y_{\text{prefix}} = \{\rho^{(1)}, \ldots, \rho^{(k^\dagger - 1)}\}, \tag{2}$$

where $\rho^{(k)}$ denotes the k-th action.

Let $\mathcal{D}(y)$ denote document observation actions in y, and aggregate retrieved documents as:

$$D_{\text{all}}(y) = \bigcup_{\rho \in \mathcal{D}(y)} \rho. \tag{3}$$

The model re-performs reasoning from the retained prefix, grounded on $D_{\text{all}}(y)$, discarding previous incorrect steps. This corrects reasoning errors without additional retrieval.

Baseline	Method	Repair Rate (↑)	HotpotQA			2Wiki			MuSiQue		
			ΔEM	ΔF1	ΔR-L	ΔEM	ΔF1	ΔR-L	ΔEM	ΔF1	ΔR-L
Qwen3-8B											
ReAct	Rerun	12.2	17.6	16.3	14.8	15.1	15.3	15.0	3.8	3.8	3.8
	Step-wise	8.1	13.0	13.0	11.3	8.3	12.8	12.5	3.2	1.1	0.8
	RAG-Critic	13.9	18.1	17.3	16.8	15.6	13.1	12.9	6.9	5.7	5.3
	DR-RAG	**21.8**	**31.6**	**27.3**	**26.9**	**26.0**	**24.4**	**24.5**	**7.7**	**6.2**	**6.2**
Search-o1	Rerun	9.8	14.7	11.8	12.2	7.3	5.2	5.1	7.3	3.5	3.5
	Step-wise	7.3	12.7	9.1	6.4	5.6	1.0	0.2	3.5	0.5	-0.2
	RAG-Critic	9.9	13.2	12.0	12.6	5.6	2.0	2.1	10.8	9.8	9.8
	DR-RAG	**22.5**	**29.8**	**22.7**	**21.1**	**19.5**	**15.3**	**14.1**	**18.1**	**15.2**	**14.6**
Search-R1	Rerun	16.6	19.4	20.8	20.6	23.3	23.5	23.9	7.2	8.8	8.7
	Step-wise	7.4	7.6	9.0	8.3	12.9	14.8	15.7	1.7	2.0	2.2
	RAG-Critic	12.2	11.8	12.8	12.7	15.3	13.8	15.0	7.3	8.2	9.2
	DR-RAG	**19.8**	**25.4**	**26.1**	**25.1**	**25.8**	**24.4**	**25.3**	**8.3**	**9.0**	**9.7**
LLaMA-3.1-8B-Instruct											
ReAct	Rerun	9.6	16.4	13.9	12.8	10.1	8.4	8.1	2.2	0.3	-0.1
	Step-wise	8.4	14.3	12.5	10.6	8.3	9.2	8.6	2.5	0.8	1.0
	RAG-Critic	5.3	7.1	5.4	4.4	6.9	3.7	3.1	1.9	-1.7	-1.4
	DR-RAG	**15.5**	**27.8**	**23.4**	**21.3**	**13.3**	**9.9**	**9.5**	**5.4**	**1.5**	**1.5**
Search-o1	Rerun	7.6	12.7	7.8	6.7	5.6	0.8	-0.0	4.5	2.3	2.1
	Step-wise	6.8	9.3	4.0	1.6	6.7	1.7	0.7	4.5	1.4	0.7
	RAG-Critic	5.7	9.8	7.2	7.2	5.6	2.4	2.3	1.7	0.1	0.2
	DR-RAG	**17.8**	**28.4**	**17.8**	**14.8**	**12.3**	**4.0**	**2.6**	**12.5**	**6.3**	**5.4**
Search-R1	Rerun	6.9	8.9	11.6	11.0	7.7	9.6	9.8	4.1	3.1	3.4
	Step-wise	5.4	8.3	9.5	9.0	5.2	8.8	9.1	2.6	2.7	3.3
	RAG-Critic	2.7	3.2	1.2	1.1	3.7	2.0	2.4	1.2	-0.7	-0.8
	DR-RAG	**12.4**	**17.2**	**14.9**	**14.1**	**13.5**	**10.0**	**10.6**	**6.5**	**4.5**	**3.8**

Table 2: Main repair results on three multi-hop QA benchmarks across two backbone LMs. All methods are evaluated on the same failed trajectories (EM=0). Repair Rate is averaged over datasets, while ΔEM, ΔF1, and ΔR-L are reported per dataset. Best results within each baseline block are in **bold**.

4.3.2 Repair under Insufficient Evidence

When retrieved evidence is insufficient, repair must address both retrieval failures and answer-generation issues.

Format-Invalid Answer Repair. Format errors may also occur under partial evidence coverage, since the model can sometimes infer the correct semantic answer from partial evidence or its parametric knowledge, even when not all supporting evidence is retrieved. In this case, repair is limited to rewriting the final answer into the required form, without altering reasoning or retrieval steps.

Retriever Error Repair. If the retriever fails to return relevant documents despite valid queries, the trajectory is truncated at $k^\dagger$ (the failed evidence block) and the prefix y_{prefix} is retained. The queries that produced insufficient evidence (at $k^\dagger$ and any subsequent failed retrievals), denoted $\mathcal{Q}_{\text{failed}}$, are rewritten into multiple alternative formulations to improve recall while preserving intent. The retriever's top-k is increased to obtain more candidate documents, then the final answer is regenerated using the retained prefix and augmented evidence.

Search Error Repair. If failure is caused by incorrect reasoning that misguides retrieval, reasoning, and retrieval beyond $k^\dagger$ are regenerated. After truncation, the model produces a high-level solution

plan conditioned on the question and retained prefix, decomposing the task into reasoning and retrieval steps to generate a new trajectory suffix. Beyond the truncation point, both reasoning and retrieval may diverge from the original path.

4.3.3 Summary of Repair Operators

All repair operators share a common structure: they preserve conditionally valid prefixes, intervene only at $k^{\dagger}$, and modify the minimal components required. This tool-conditioned design aligns with the diagnosis module, enabling efficient, localized repair.

5 Experiments

5.1 Research Questions

We investigate five research questions: **RQ1**, whether DR-RAG improves repair rates and performance gains across agentic RAG baselines; **RQ2**, the contributions of error taxonomy and localization; **RQ3**, inference efficiency including both diagnosis and repair costs; **RQ4**, error classification and failure localization accuracy across datasets and baselines; and **RQ5**, repair rates across error types and the effect of diagnosis correctness.

5.2 Experiment Settings

Datasets and Metrics. Experiments are conducted on three multi-hop QA benchmarks: HotpotQA (Yang et al., 2018), 2Wiki (Ho et al., 2020), and MuSiQue (Trivedi et al., 2022), all providing gold supporting evidence for fine-grained analysis. We evaluate using standard QA metrics: Exact Match (EM), F1, and ROUGE-L (Lin, 2004). To measure repair effectiveness, we report the *Repair Rate*, defined as the proportion of initially incorrect instances (EM=0) corrected to exact match (EM=1). Absolute improvements relative to the unrepaired baseline are denoted as Δ (e.g., ΔEM). Detailed dataset statistics and metric definitions are provided in Appendix B.2.

Baselines. We evaluate DR-RAG on three agentic RAG baselines: **ReAct** (Yao et al., 2023b), **Search-o1** (Li et al., 2025a), and **Search-R1** (Jin et al., 2025), all of which generate multi-step trajectories interleaving reasoning and retrieval. We compare DR-RAG with the following repair strategies applied to the same failed trajectories:

(1) Rerun discards the entire failed trajectory and regenerates it from scratch, treating all prior steps as unreliable and unnecessarily recomputing prefixes that may still be valid.

(2) Step-wise Retry sequentially verifies reasoning steps and reuses the validated prefix for subsequent steps. While enabling partial reuse, it does not classify error types or explicitly diagnose the earliest failure point.

(3) RAG-Critic (Dong et al., 2025c) uses a diagnose–replan–rerun approach, identifying coarse error categories and replanning recovery actions. Diagnosis operates at the trajectory level without precise failure localization, which may lead to re-execution or modification of correct segments.

Implementation Details. Experiments follow prior agentic RAG work (Jiang et al., 2025) using two backbone LMs: Qwen3-8B (Yang et al., 2025), LLaMA-3.1-8B-Instruct (Team, 2024). All baselines and diagnosis-repair strategies share the same retrieval setup, using the full Wikipedia 2018 dump (Karpukhin et al., 2020) indexed with FAISS (Johnson et al., 2021), BGE-large-en-v1.5 (Xiao et al., 2023) as the retriever, and retrieving the top-5 documents for each query. All inference costs incurred by diagnosis and repair are included in the efficiency evaluation. Experiments run on four NVIDIA RTX Pro 6000 GPUs using vLLM (Kwon et al., 2023). Full implementation details are provided in Appendix B.1.

5.3 Main Results (RQ1)

Table 2 reports repair performance across three benchmarks and two backbone LMs. DR-RAG consistently achieves the highest repair rate and metric gains across all settings, substantially outperforming competing repair strategies in most cases (e.g., 21.8% vs. 13.9% on ReAct with Qwen3-8B).

Baseline	HotpotQA		2Wiki		MuSiQue	
	ΔF1	ΔEM	ΔF1	ΔEM	ΔF1	ΔEM
DR-RAG	**25.4**	**28.9**	**21.4**	**23.8**	**10.1**	**11.4**
w/o Taxonomy	16.0	17.5	15.3	16.0	6.3	6.3
w/o Localization	16.1	19.5	15.7	18.3	7.0	8.6

Table 3: Ablation study on error taxonomy and localization. Results are averaged over three baselines. Removing either component degrades both ΔF1 and ΔEM.

Method	ReAct			Search-o1			Search-R1		
	HotpotQA	2Wiki	MuSiQue	HotpotQA	2Wiki	MuSiQue	HotpotQA	2Wiki	MuSiQue
Rerun	713	842	927	633	716	817	652	474	773
Step-wise	1,031	915	990	611	762	1,034	530	702	734
RAG-Critic	5,036	5,003	7,111	4,735	5,595	8,790	5,511	5,193	8,852
DR-RAG	**308**	**396**	**233**	**293**	**403**	**250**	**289**	**276**	**394**

Table 4: Inference time (seconds) for repairing all failed trajectories per dataset. DR-RAG includes both diagnosis and repair time. Lowest values are in **bold**.

The advantage stems from taxonomy-guided localization: by identifying error type and the earliest failure step, DR-RAG applies targeted repair (format rewriting, re-reasoning, or re-retrieval) without regenerating valid prefixes. In contrast, Rerun often reproduces errors, Step-wise lacks error-type awareness, and RAG-Critic may over-repair due to imprecise localization.

DR-RAG also generalizes across backbones: on LLaMA-3.1-8B-Instruct, it achieves 12.4%–17.8% repair rates versus 2.7%–9.6% for competing methods, confirming that the approach is not tied to a specific backbone. Gains on MuSiQue are relatively smaller in several settings because the task is more challenging and leaves less recoverable room for post-hoc repair. Nevertheless, DR-RAG still consistently outperforms all baselines.

5.4 Ablation Study (RQ2)

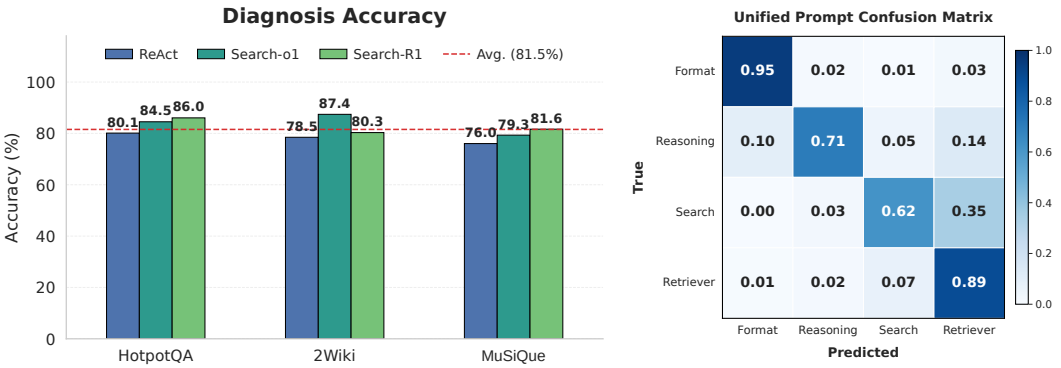
Table 3 isolates the contributions of taxonomy and localization. Removing taxonomy (routing errors through a generic repair) drops ΔEM by 7–11 points, as the model applies mismatched strategies (e.g., re-retrieval for format errors). Removing localization (repairing from the trajectory start rather than the failure point) causes a smaller but consistent drop of 5–9 points, due to unnecessary regeneration of correct prefixes. The two components are complementary: taxonomy selects *what* to fix, while localization determines *where* to intervene.

5.5 Inference Efficiency (RQ3)

Table 4 compares end-to-end inference time across repair strategies. DR-RAG is 2–3× faster than Rerun and Step-wise, and over 10× faster than RAG-Critic, despite including the diagnosis overhead. The efficiency gain comes from localized repair: DR-RAG reuses conditionally valid prefixes and only performs generation for the diagnosed repair step, whereas Rerun regenerates entire trajectories and RAG-Critic incurs additional planning and execution overhead, as its generated repair plan may invoke multiple LLM-based repair operators.

5.6 Error Diagnosis Reliability (RQ4)

We evaluate DR-RAG's diagnosis module against Qwen3.5-Plus-annotated diagnostic labels across all 9 baseline×dataset combinations (Figure 2a). The module achieves 81.3% overall accuracy, with per-configuration accuracy ranging from 76.0% to 87.4%, showing stable diagnosis reliability across datasets and baselines. This suggests that the distilled diagnosis model generalizes consistently across different agentic RAG settings rather than overfitting to a specific dataset or baseline.



(a) Accuracy across datasets and baselines.

(b) Confusion matrix across error types.

Figure 2: Automated error diagnosis evaluated against Qwen3.5-Plus-annotated diagnostic labels across all datasets and baselines.

Diagnosis	Format	Reasoning	Retriever	Search	All
Correct	41.3	26.5	17.7	16.5	22.1
Wrong	11.1	15.8	12.2	7.2	10.5
Overall	39.5	24.1	17.1	13.1	20.1

Table 5: Repair rate (%) by error type and diagnosis correctness. "Correct"/"Wrong" indicate whether the diagnosis matches the Qwen3.5-Plus label. Results are aggregated across baselines and datasets.

Per-type analysis (Figure 2b) shows that most error types are diagnosed reliably, especially Format (94.9%) and Retriever (89.1%) errors, which have clearer surface signals. Reasoning errors reach 70.7% accuracy, while Search errors are lower at 62.0% because they share the insufficient-evidence setting with Retriever errors and require finer causal attribution over query formation. Nevertheless, the impact is limited because Search errors are less frequent, and many misclassified cases are still routed to retrieval-enhanced repair, which can recover missing evidence through additional retrieval and re-reasoning. Additional robustness analysis is provided in Appendix D.2.

5.7 Repair Performance by Error Type and Diagnostic Accuracy (RQ5)

Table 5 analyzes repair performance by error type and diagnostic correctness. Correct diagnoses consistently lead to higher repair rates than incorrect ones, confirming that accurate failure attribution is important for effective repair. At the same time, misdiagnosed cases still retain a non-trivial repair rate, indicating that the repair module is not brittle to imperfect diagnostic signals.

Across error types, Format and Reasoning errors are more repairable because they mainly require answer rewriting or re-reasoning over sufficient evidence, whereas Retriever and Search errors are harder since they require recovering missing evidence. Figure 7 in Appendix D.11 further visualizes this per-type repair trend across different repair strategies. Thus, RQ5 shows that DR-RAG is both diagnosis-sensitive and repair-stable: better diagnosis improves routing and recovery, while localized repair remains functional under diagnostic noise.

6 Conclusion

We introduce DR-RAG, a diagnose-and-repair framework for Agentic RAG that corrects failures via trajectory-level diagnosis and localized repair, avoiding full-pipeline reruns. Experiments demonstrate consistent improvements in accuracy and efficiency, supported by reliable automated diagnosis. DR-RAG reframes failure handling as a structured repair problem, providing a foundation for more robust and efficient multi-step reasoning systems. Limitations are discussed in Appendix G.

REFERENCES

Garima Agrawal, Tharindu Kumarage, Zeyad Alghamdi, and Huan Liu. Mindful-rag: A study of points of failure in retrieval augmented generation. In *FLLM*, pp. 607–611. IEEE, 2024.

Akari Asai, Zeqiu Wu, Yizhong Wang, Avirup Sil, and Hannaneh Hajishirzi. Self-rag: Learning to retrieve, generate, and critique through self-reflection. In *ICLR*. OpenReview.net, 2024.

Tianyu Cao, Neel Bhandari, Akhila Yerukola, Akari Asai, and Maarten Sap. Out of style: Rag's fragility to linguistic variation. *CoRR*, abs/2504.08231, 2025.

Guoxin Chen, Zile Qiao, Xuanzhong Chen, Donglei Yu, Haotian Xu, Wayne Xin Zhao, Ruihua Song, Wenbiao Yin, Huifeng Yin, Liwen Zhang, Kuan Li, Minpeng Liao, Yong Jiang, Pengjun Xie, Fei Huang, and Jingren Zhou. Iterresearch: Rethinking long-horizon agents via markovian state reconstruction. *CoRR*, abs/2511.07327, 2025.

Yuqin Dai, Guoqing Wang, Yuan Wang, Kairan Dou, Kaichen Zhou, Zhanwei Zhang, Shuo Yang, Fei Tang, Jun Yin, Pengyu Zeng, Zhenzhe Ying, Can Yi, Changhua Meng, Yuchen Zhou, Yongliang Shen, and Shuai Lu. Evinote-rag: Enhancing RAG models via answer-supportive evidence notes. *CoRR*, abs/2509.00877, 2025.

Guanting Dong, Licheng Bao, Zhongyuan Wang, Kangzhi Zhao, Xiaoxi Li, Jiajie Jin, Jinghan Yang, Hangyu Mao, Fuzheng Zhang, Kun Gai, Guorui Zhou, Yutao Zhu, Ji-Rong Wen, and Zhicheng Dou. Agentic entropy-balanced policy optimization. *CoRR*, abs/2510.14545, 2025a.

Guanting Dong, Yifei Chen, Xiaoxi Li, Jiajie Jin, Hongjin Qian, Yutao Zhu, Hangyu Mao, Guorui Zhou, Zhicheng Dou, and Ji-Rong Wen. Tool-star: Empowering llm-brained multi-tool reasoner via reinforcement learning. *CoRR*, abs/2505.16410, 2025b.

Guanting Dong, Jiajie Jin, Xiaoxi Li, Yutao Zhu, Zhicheng Dou, and Ji-Rong Wen. Rag-critic: Leveraging automated critic-guided agentic workflow for retrieval augmented generation. In *ACL (1)*, pp. 3551–3578. Association for Computational Linguistics, 2025c.

Guanting Dong, Hangyu Mao, Kai Ma, Licheng Bao, Yifei Chen, Zhongyuan Wang, Zhongxia Chen, Jiazhen Du, Huiyang Wang, Fuzheng Zhang, Guorui Zhou, Yutao Zhu, Ji-Rong Wen, and Zhicheng Dou. Agentic reinforced policy optimization. *CoRR*, abs/2507.19849, 2025d.

Jiaxuan Gao, Wei Fu, Minyang Xie, Shusheng Xu, Chuyi He, Zhiyu Mei, Banghua Zhu, and Yi Wu. Beyond ten turns: Unlocking long-horizon agentic search with large-scale asynchronous RL. *CoRR*, abs/2508.07976, 2025.

Gauthier Guinet, Behrooz Omidvar-Tehrani, Anoop Deoras, and Laurent Callot. Automated evaluation of retrieval-augmented language models with task-specific exam generation. In *ICML*. OpenReview.net, 2024.

Kelvin Guu, Kenton Lee, Zora Tung, Panupong Pasupat, and Ming-Wei Chang. Retrieval augmented language model pre-training. In *ICML*, volume 119 of *Proceedings of Machine Learning Research*, pp. 3929–3938. PMLR, 2020.

Xanh Ho, Anh-Khoa Duong Nguyen, Saku Sugawara, and Akiko Aizawa. Constructing A multi-hop QA dataset for comprehensive evaluation of reasoning steps. In *Proceedings of the 28th International Conference on Computational Linguistics, COLING 2020, Barcelona, Spain (Online), December 8-13, 2020*, pp. 6609–6625. International Committee on Computational Linguistics, 2020.

Chengkai Huang, Yu Xia, Rui Wang, Kaige Xie, Tong Yu, Julian J. McAuley, and Lina Yao. Embedding-informed adaptive retrieval-augmented generation of large language models. In *Proceedings of the 31st International Conference on Computational Linguistics, COLING 2025, Abu Dhabi, UAE, January 19-24, 2025*, pp. 1403–1412, 2025a.

Chengkai Huang, Xiaodi Chen, Hongtao Huang, Quan Z Sheng, and Lina Yao. Generative chain of behavior for user trajectory prediction. *arXiv preprint arXiv:2601.18213*, 2026.

Hongtao Huang, Chengkai Huang, Junda Wu, Tong Yu, Julian McAuley, and Lina Yao. Listwise preference diffusion optimization for user behavior trajectories prediction. *arXiv preprint arXiv:2511.00530*, 2025b.

Soyeong Jeong, Jinheon Baek, Sukmin Cho, Sung Ju Hwang, and Jong Park. Adaptive-rag: Learning to adapt retrieval-augmented large language models through question complexity. In *NAACL-HLT*, pp. 7036–7050. Association for Computational Linguistics, 2024.

Jinhao Jiang, Jiayi Chen, Junyi Li, Ruiyang Ren, Shijie Wang, Xin Zhao, Yang Song, and Tao Zhang. Rag-star: Enhancing deliberative reasoning with retrieval augmented verification and refinement. In *NAACL (Long Papers)*, pp. 7064–7074. Association for Computational Linguistics, 2025.

Shuguang Jiao, Xinyu Xiao, Yunfan Wei, Shuhan Qi, Chengkai Huang, Quan Z. Michael Sheng, and Lina Yao. Prunerag: Confidence-guided query decomposition trees for efficient retrieval-augmented generation, 2026.

Bowen Jin, Hansi Zeng, Zhenrui Yue, Jinsung Yoon, Sercan Arik, Dong Wang, Hamed Zamani, and Jiawei Han. Search-r1: Training llms to reason and leverage search engines with reinforcement learning, 2025. URL https://arxiv.org/abs/2503.09516.

Jeff Johnson, Matthijs Douze, and Hervé Jégou. Billion-scale similarity search with gpus. *IEEE Trans. Big Data*, 7(3):535–547, 2021.

Vladimir Karpukhin, Barlas Oguz, Sewon Min, Patrick Lewis, Ledell Wu, Sergey Edunov, Danqi Chen, and Wen-tau Yih. Dense passage retrieval for open-domain question answering. In *EMNLP (1)*, pp. 6769–6781. Association for Computational Linguistics, 2020.

Woosuk Kwon, Zhuohan Li, Siyuan Zhuang, Ying Sheng, Lianmin Zheng, Cody Hao Yu, Joseph Gonzalez, Hao Zhang, and Ion Stoica. Efficient memory management for large language model serving with pagedattention. In *SOSP*, pp. 611–626. ACM, 2023.

Kin Kwan Leung, Mouloud Belbahri, Yi Sui, Alex Labach, Xueying Zhang, Stephen Anthony Rose, and Jesse C. Cresswell. Classifying and addressing the diversity of errors in retrieval-augmented generation systems, 2026.

Patrick Lewis, Ethan Perez, Aleksandra Piktus, Fabio Petroni, Vladimir Karpukhin, Naman Goyal, Heinrich Küttler, Mike Lewis, Wen-tau Yih, Tim Rocktäschel, Sebastian Riedel, and Douwe Kiela. Retrieval-augmented generation for knowledge-intensive NLP tasks. In *NeurIPS*, 2020.

Xiaoxi Li, Guanting Dong, Jiajie Jin, Yuyao Zhang, Yujia Zhou, Yutao Zhu, Peitian Zhang, and Zhicheng Dou. Search-o1: Agentic search-enhanced large reasoning models. *CoRR*, abs/2501.05366, 2025a.

Xiaoxi Li, Wenxiang Jiao, Jiarui Jin, Guanting Dong, Jiajie Jin, Yinuo Wang, Hao Wang, Yutao Zhu, Ji-Rong Wen, Yuan Lu, and Zhicheng Dou. Deepagent: A general reasoning agent with scalable toolsets. *CoRR*, abs/2510.21618, 2025b.

Hunter Lightman, Vineet Kosaraju, Yuri Burda, Harrison Edwards, Bowen Baker, Teddy Lee, Jan Leike, John Schulman, Ilya Sutskever, and Karl Cobbe. Let's verify step by step. In *The Twelfth International Conference on Learning Representations, ICLR 2024, Vienna, Austria, May 7-11, 2024*. OpenReview.net, 2024. URL https://openreview.net/forum?id=v8L0pN6EOi.

Chin-Yew Lin. ROUGE: A package for automatic evaluation of summaries. In *Text Summarization Branches Out*, pp. 74–81, Barcelona, Spain, July 2004. Association for Computational Linguistics.

Haowei Lou, Chengkai Huang, Hye-young Paik, Yongquan Hu, Aaron Quigley, Wen Hu, and Lina Yao. Speechagent: An end-to-end mobile infrastructure for speech impairment assistance. *arXiv preprint arXiv:2510.20113*, 2025.

Qi Luo, Xiaonan Li, Yuxin Wang, Tingshuo Fan, Yuan Li, Xinchi Chen, and Xipeng Qiu. MARAG-R1: beyond single retriever via reinforcement-learned multi-tool agentic retrieval. *CoRR*, abs/2510.27569, 2025.

Dongyu Ru, Lin Qiu, Xiangkun Hu, Tianhang Zhang, Peng Shi, Shuaichen Chang, Cheng Jiayang, Cunxiang Wang, Shichao Sun, Huanyu Li, Zizhao Zhang, Binjie Wang, Jiarong Jiang, Tong He, Zhiguo Wang, Pengfei Liu, Yue Zhang, and Zheng Zhang. Ragchecker: A fine-grained framework for diagnosing retrieval-augmented generation. In *NeurIPS*, 2024.

Zhengliang Shi, Yiqun Chen, Haitao Li, Weiwei Sun, Shiyu Ni, Yougang Lyu, Run-Ze Fan, Bowen Jin, Yixuan Weng, Minjun Zhu, Qiujie Xie, Xinyu Guo, Qu Yang, Jiayi Wu, Jujia Zhao, Xiaqiang Tang, Xinbei Ma, Cunxiang Wang, Jiaxin Mao, Qingyao Ai, Jen-Tse Huang, Wenxuan Wang, Yue Zhang, Yiming Yang, Zhaopeng Tu, and Zhaochun Ren. Deep research: A systematic survey, 2025.

Joykirat Singh, Raghav Magazine, Yash Pandya, and Akshay Nambi. Agentic reasoning and tool integration for llms via reinforcement learning. *CoRR*, abs/2505.01441, 2025.

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

Llama Team. The llama 3 herd of models. *CoRR*, abs/2407.21783, 2024.

Qwen Team. Qwen3.5: Accelerating productivity with native multimodal agents, February 2026. URL https://qwen.ai/blog?id=qwen3.5.

Harsh Trivedi, Niranjan Balasubramanian, Tushar Khot, and Ashish Sabharwal. Musique: Multihop questions via single-hop question composition. *Trans. Assoc. Comput. Linguistics*, 10:539–554, 2022.

Peilin Wu, Mian Zhang, Kun Wan, Wentian Zhao, Kaiyu He, Xinya Du, and Zhiyu Chen. Hiprag: Hierarchical process rewards for efficient agentic retrieval augmented generation. *CoRR*, abs/2510.07794, 2025.

Shitao Xiao, Zheng Liu, Peitian Zhang, and Niklas Muennighoff. C-pack: Packaged resources to advance general chinese embedding. *CoRR*, abs/2309.07597, 2023.

Weimin Xiong, Yifan Song, Xiutian Zhao, Wenhao Wu, Xun Wang, Ke Wang, Cheng Li, Wei Peng, and Sujian Li. Watch every step! LLM agent learning via iterative step-level process refinement. In Yaser Al-Onaizan, Mohit Bansal, and Yun-Nung Chen (eds.), *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing, EMNLP 2024, Miami, FL, USA, November 12-16, 2024*, pp. 1556–1572. Association for Computational Linguistics, 2024. doi: 10.18653/V1/2024.EMNLP-MAIN.93. URL https://doi.org/10.18653/v1/2024.emnlp-main.93.

Shi-Qi Yan, Jia-Chen Gu, Yun Zhu, and Zhen-Hua Ling. Corrective retrieval augmented generation. *CoRR*, abs/2401.15884, 2024.

An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Gao, Chengen Huang, Chenxu Lv, Chujie Zheng, Dayiheng Liu, Fan Zhou, Fei Huang, Feng Hu, Hao Ge, Haoran Wei, Huan Lin, Jialong Tang, Jian Yang, Jianhong Tu, Jianwei Zhang, Jianxin Yang, Jiaxi Yang, Jing Zhou, Jingren Zhou, Junyang Lin, Kai Dang, Keqin Bao, Kexin Yang, Le Yu, Lianghao Deng, Mei Li, Mingfeng Xue, Mingze Li, Pei Zhang, Peng Wang, Qin Zhu, Rui Men, Ruize Gao, Shixuan Liu, Shuang Luo, Tianhao Li, Tianyi Tang, Wenbiao Yin, Xingzhang Ren, Xinyu Wang, Xinyu Zhang, Xuancheng Ren, Yang Fan, Yang Su, Yichang Zhang, Yinger Zhang, Yu Wan, Yuqiong Liu, Zekun Wang, Zeyu Cui, Zhenru Zhang, Zhipeng Zhou, and Zihan Qiu. Qwen3 technical report, 2025. URL https://arxiv.org/abs/2505.09388.

Zhilin Yang, Peng Qi, Saizheng Zhang, Yoshua Bengio, William W. Cohen, Ruslan Salakhutdinov, and Christopher D. Manning. Hotpotqa: A dataset for diverse, explainable multi-hop question answering. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, Brussels, Belgium, October 31 - November 4, 2018*, pp. 2369–2380. Association for Computational Linguistics, 2018.

Shunyu Yao, Dian Yu, Jeffrey Zhao, Izhak Shafran, Tom L. Griffiths, Yuan Cao, and Karthik Narasimhan. Tree of thoughts: Deliberate problem solving with large language models. In *Advances in Neural Information Processing Systems 36: Annual Conference on Neural Information Processing Systems 2023, NeurIPS 2023, New Orleans, LA, USA, December 10 - 16, 2023*, 2023a.

Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik Narasimhan, and Yuan Cao. React: Synergizing reasoning and acting in language models. In *The Eleventh International Conference on Learning Representations, ICLR 2023, Kigali, Rwanda, May 1-5, 2023*. OpenReview.net, 2023b.

Asaf Yehudai, Lilach Eden, Yotam Perlitz, Roy Bar-Haim, and Michal Shmueli-Scheuer. CLEAR: error analysis via llm-as-a-judge made easy. *CoRR*, abs/2507.18392, 2025.

Chao Zhang, Yuhao Wang, Derong Xu, Haoxin Zhang, Yuanjie Lyu, Yuhao Chen, Shuochen Liu, Tong Xu, Xiangyu Zhao, Yan Gao, Yao Hu, and Enhong Chen. Tearag: A token-efficient agentic retrieval-augmented generation framework. *CoRR*, abs/2511.05385, 2025.

Yaowei Zheng, Richong Zhang, Junhao Zhang, Yanhan Ye, Zheyan Luo, Zhangchi Feng, and Yongqiang Ma. Llamafactory: Unified efficient fine-tuning of 100+ language models. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 3: System Demonstrations)*, Bangkok, Thailand, 2024. Association for Computational Linguistics. URL http://arxiv.org/abs/2403.13372.

Andy Zhou, Kai Yan, Michal Shlapentokh-Rothman, Haohan Wang, and Yu-Xiong Wang. Language agent tree search unifies reasoning, acting, and planning in language models. In Ruslan Salakhutdinov, Zico Kolter, Katherine A. Heller, Adrian Weller, Nuria Oliver, Jonathan Scarlett, and Felix Berkenkamp (eds.), *Forty-first International Conference on Machine Learning, ICML 2024, Vienna, Austria, July 21-27, 2024*, Proceedings of Machine Learning Research, pp. 62138–62160. PMLR / OpenReview.net, 2024. URL https://proceedings.mlr.press/v235/zhou24r.html.

APPENDIX OUTLINE

A Information About Use of AI Assistants . . . 15
B Implementation Details . . . 15
B.1 Experimental Settings . . . 15
B.2 Datasets and Metrics . . . 15
B.3 Distillation Data Split and Question Overlap . . . 15
C Related Work . . . 16
C.1 Retrieval Augmented Generation . . . 16
C.2 Agentic RAG . . . 16
C.3 Comparison with Active and Step-Level Self-Corrective RAG . . . 16
D Additional Experiments . . . 17
D.1 Supplementary Main Results with Qwen3-4B . . . 17
D.2 Robustness Analysis of the Distilled Diagnosis Model . . . 18
D.3 Case Study . . . 21
D.4 Full-Test-Set Performance and Amortized Cost . . . 21
D.5 Repair Token and LLM-Call Costs . . . 23
D.6 Deployment Simulation without Gold Answers . . . 23
D.7 Retrieval-Budget-Controlled Comparison . . . 24
D.8 Integrated versus Standalone Coverage Prediction . . . 25
D.9 Multi-Seed Stability and Statistical Significance . . . 25
D.10 Sensitivity to the Repair Retriever . . . 26
D.11 Error Distribution and Repair Trends . . . 26
E Diagnosis Algorithm Details . . . 28
E.1 Overall Diagnosis Pipeline . . . 28
E.2 Problem Setup and Trajectory Representation . . . 28
E.3 Distilled Diagnosis Model . . . 29
E.4 Diagnosis Logic . . . 30
E.5 Error Criteria and Query-Level Judgement . . . 31
E.6 Failure Localization and Repair Interface . . . 32
E.7 Teacher and Student Prompts . . . 32
F Repair Module Details . . . 33
F.1 Format-Invalid Answer Repair . . . 33
F.2 Reasoning Logic Error Repair . . . 33
F.3 Retriever Error Repair . . . 33
F.4 Search Error Repair . . . 34
F.5 Repair Algorithm Summary . . . 35
F.6 Repair Prompt Templates . . . 35
G Limitations . . . 39

A Information About Use of AI Assistants

AI assistants were used during the research and writing process. Their use was limited to grammar checking, language polishing, improving manuscript readability, and assisting with part of the data annotation process. All annotations and manuscript revisions involving AI assistance were manually inspected and validated by the authors.

B Implementation Details

B.1 Experimental Settings

Following prior work on agentic RAG systems (Jiang et al., 2025), we use two backbone language models in the main experiments: Qwen3-8B (Yang et al., 2025) and LLaMA-3.1-8B-Instruct (Team, 2024). We additionally include Qwen3-4B (Yang et al., 2025) as a supplementary backbone setting in Appendix D.1. For each backbone setting, the corresponding diagnosis and repair models are instantiated with the same backbone. All baselines and diagnosis-repair strategies are evaluated under identical retrieval and inference settings to ensure fair comparison. Following Karpukhin et al. (2020), we use the full Wikipedia 2018 dump as the retrieval corpus. Our main experiments follow the retrieval setup of Jiang et al. (2025), employing FAISS (Johnson et al., 2021) for indexing, *BGE-large-en-v1.5* (Xiao et al., 2023) as retriever, and retrieving top-5 documents. All methods share the same retrieval settings. All methods are evaluated in a post-hoc manner on failed trajectories produced by the original agentic RAG baselines, without modifying their underlying retrieval or reasoning components. The evidence sufficiency indicator $\mathrm{Cov}(y)$ is estimated within the reasoning process of the diagnosis model, indicating whether the retrieved documents in a trajectory are sufficient to support the answer. This retrieval-sufficiency judgment is used to guide error classification and repair strategy selection. We report inference time as the efficiency metric, including the time consumed by both diagnosis and repair. Experiments are conducted on four NVIDIA RTX Pro 6000 GPUs, using vLLM for inference (Kwon et al., 2023).

B.2 Datasets and Metrics

We conduct experiments on three multi-hop question answering benchmarks: HotpotQA, 2Wiki, and MuSiQue. HotpotQA (Yang et al., 2018) is a large-scale benchmark for multi-hop reasoning over Wikipedia, 2WikiMultihopQA (2Wiki) (Ho et al., 2020) focuses on explicit multi-hop reasoning paths, and MuSiQue (Trivedi et al., 2022) contains 2–4 hop questions constructed from multiple single-hop datasets. All three datasets provide gold supporting evidence, enabling fine-grained analysis of evidence sufficiency and error diagnosis.

For evaluation, we report standard question answering metrics, including Exact Match (EM), F1, and ROUGE-L (Lin, 2004). EM measures whether the predicted answer exactly matches the ground truth. F1 scores evaluate the token-level overlap, while ROUGE-L measures lexical similarity based on the longest common subsequence.

To evaluate error correction effectiveness, we report the *Repair Rate*, defined as the proportion of instances that are initially incorrect (i.e., EM $= 0$) but are successfully corrected to an exact match after repair (i.e., repaired EM $= 1$). For overall performance, we report the absolute improvement, denoted as Δ, relative to the unrepaired baseline. Specifically, ΔEM measures the net increase in the aggregate EM score attributable to the repair process.

B.3 Distillation Data Split and Question Overlap

The distillation (training) data and the evaluation set are drawn from **separately sampled question pools** and split a priori. The evaluation set contains 500 questions per dataset, for **1,500** questions in total:

- **Training (distillation):** We sample **failed (EM=0) trajectories** from the full grid of backbones {Qwen3-4B, Qwen3-8B} $\times$ baselines {ReAct, SearchO1} $\times$ datasets {HotpotQA, 2Wiki, MuSiQue}. This yields **11,323 EM=0 trajectories** covering **8,239 unique questions**; each teacher-annotated trajectory becomes one diagnosis training sample. Since a question fails only in a subset

of backbone×baseline combinations, each question contributes about **1.37** failed trajectories on average.

- **Evaluation:** HotpotQA / 2Wiki / MuSiQue each contain **500** questions, for **1,500** questions total.
- **Leakage check:** We run a question-level set-intersection between training and evaluation questions, normalized by whitespace / case / trailing punctuation. The intersection is empty: **0 overlapping questions**.

C Related Work

C.1 Retrieval Augmented Generation

Large Language Models (LLMs) are prone to hallucinations and lack access to up-to-date knowledge. Retrieval-Augmented Generation (RAG) addresses these limitations by integrating external knowledge sources at inference time, improving factual accuracy without requiring model retraining Guu et al. (2020); Karpukhin et al. (2020). Early work, exemplified by the original RAG framework Lewis et al. (2020), established the retriever–generator architecture that underpins retrieval-enhanced generation. Building on this paradigm, subsequent studies progressively extend RAG with more advanced reasoning and retrieval mechanisms, including Chain-of-Thought Yao et al. (2023b), Tree-of-Thought reasoning Yao et al. (2023a); Jiao et al. (2026), and adaptive retrieval strategies Jeong et al. (2024). These advances enable effective multi-step reasoning in RAG systems.

C.2 Agentic RAG

In recent years, Agentic RAG (e.g., ReAct Yao et al. (2023b), Search-o1 Li et al. (2025a), Search-R1 Jin et al. (2025)) has formulated reasoning and tool invocation (retrieval Dong et al. (2025b), memory Li et al. (2025b), and summarization Dai et al. (2025)) as a unified sequential decision-making process. Through policy learning Dong et al. (2025d;a), these approaches enable models to autonomously select search and information processing actions during reasoning. Existing work Wu et al. (2025); Chen et al. (2025); Gao et al. (2025) typically optimizes reasoning and retrieval policies using outcome-level rewards or preference signals, thereby improving performance on multi-step search and complex reasoning tasks.

More recently, several studies have investigated robustness and failure diagnosis in RAG and agentic RAG systems. RAGChecker Ru et al. (2024) is a claim-level entailment-based diagnostic framework that evaluates RAG system performance but does not address how to repair or improve the system after errors are identified. RAG-Critic Dong et al. (2025c) responds to detected error types by replanning a new reasoning and retrieval trajectory, without localizing the earliest error or reusing validated prefix trajectories. As a result, failed trajectories are still treated as indivisible units, leading to redundant computation and limited error correction efficiency.

In contrast, fine-grained and cost-efficient repair of failed agentic RAG trajectories remains underexplored Huang et al. (2026; 2025b); Lou et al. (2025). Prior work largely lacks explicit error localization within trajectories and does not consider reusing validated reasoning prefixes or retrieved evidence. This gap motivates our work, which treats agentic RAG failures as structured objects that can be diagnosed, localized, and selectively repaired rather than blindly rerun.

C.3 Comparison with Active and Step-Level Self-Corrective RAG

Self-corrective RAG methods improve reliability along two adjacent directions. The first is *active, generation-time correction*, exemplified by Self-RAG (Asai et al., 2024) and CRAG (Yan et al., 2024), where the model dynamically controls retrieval and evaluates evidence during generation using reflection tokens or lightweight evaluators. These methods mainly intervene in the ongoing generation process, deciding when to retrieve, critique, or continue generation. However, they do not target the post-hoc repair setting where a completed failed trajectory must be diagnosed, localized, and selectively corrected.

The second direction is *step-level online verification*, where each action or intermediate state is scored using value functions, reward models, or external critics (Lightman et al., 2024; Xiong et al., 2024; Zhou et al., 2024). Although such methods can provide immediate feedback, they require access to

Baseline	Method	Repair Rate (↑)	HotpotQA			2Wiki			MuSiQue		
			ΔEM	ΔF1	ΔR-L	ΔEM	ΔF1	ΔR-L	ΔEM	ΔF1	ΔR-L
Qwen3-4B											
ReAct	Rerun	9.6	14.3	14.0	12.7	12.8	14.3	13.8	1.6	1.0	0.8
	Step-wise	4.8	8.0	6.2	5.2	3.7	4.4	4.0	2.8	-0.4	-0.5
	RAG-Critic	7.6	8.4	8.1	7.4	11.5	10.3	10.3	2.8	1.3	1.6
	DR-RAG	**18.1**	**28.2**	**24.2**	**22.5**	**18.8**	**16.8**	**17.1**	**7.4**	**6.2**	**6.8**
Search-o1	Rerun	8.2	12.7	10.7	9.4	7.8	5.5	5.5	4.2	1.8	1.5
	Step-wise	3.9	6.4	2.1	0.6	3.4	0.8	0.4	2.1	-1.0	-1.9
	RAG-Critic	7.3	13.7	10.9	12.0	3.9	2.6	2.5	4.2	3.9	4.1
	DR-RAG	**21.6**	**30.4**	**24.1**	**21.3**	**18.4**	**13.6**	**12.9**	**16.1**	**13.7**	**12.6**
Search-R1	Rerun	13.4	13.4	16.3	16.6	22.4	21.8	21.8	4.3	6.0	6.4
	Step-wise	5.5	6.1	7.2	7.0	7.7	12.7	13.3	2.9	3.8	4.1
	RAG-Critic	6.3	6.7	6.9	6.9	10.7	9.4	9.9	1.4	1.1	1.9
	DR-RAG	**17.6**	**20.1**	**22.8**	**22.6**	**26.1**	**26.0**	**26.4**	**6.7**	**8.4**	**8.3**

Table 6: Supplementary main results with Qwen3-4B as backbone. All methods operate on the same failed trajectories. The repair rate is averaged across three datasets. Best results per baseline are in **bold**.

and modification of the agent's internal execution loop, making them difficult to apply to black-box agents or closed-source agentic RAG systems. Moreover, introducing verification at every step may disrupt the continuity of long-horizon reasoning, prematurely steer the model away from exploratory search paths, and impose substantial per-step computational overhead.

These two lines of work focus on improving reliability during trajectory construction, whereas DR-RAG studies how to recover from failures after a trajectory has already been generated. DR-RAG is designed as a post-hoc repair framework that treats the underlying agent as a black box. It only acts on failed trajectories, where a backbone-specific diagnosis model jointly assesses evidence sufficiency, classifies the failure type, and localizes the earliest erroneous step. Based on this structured diagnosis, DR-RAG applies localized repair while reusing valid reasoning prefixes and retrieved evidence. This design provides three key advantages: (i) it does not require modifying the underlying agent workflow, (ii) it reduces inference overhead by intervening only on failed trajectories, and (iii) it enables full-trajectory context for accurate failure localization and maximal prefix reuse.

Furthermore, DR-RAG's diagnosis module is trained using a *distilled SFT* process: a strong teacher LLM annotates failed trajectories with structured reasoning and compact labels, while the student model learns to output the same diagnostic reasoning in a lightweight format. In contrast, online step-level verifiers typically require extensive per-step supervision, reward modeling, or iterative tool feedback during inference, which can be costly and tightly coupled to the agent workflow. By separating trajectory-level diagnosis from generation-time action selection, DR-RAG can be applied across different agentic RAG backbones without modifying or instrumenting the base agent, making it a flexible and efficient complement to existing self-corrective methods.

D Additional Experiments

D.1 Supplementary Main Results with Qwen3-4B

Table 6 reports supplementary results with Qwen3-4B, where both the diagnosis model and repair model are instantiated with Qwen3-4B. These results complement the main evaluation in Table 2, which uses Qwen3-8B and LLaMA-3.1-8B-Instruct.

DR-RAG achieves 18.1%, 21.6%, and 17.6% repair rate on ReAct, Search-o1, and Search-R1, respectively, roughly doubling the best baseline in each case (Rerun: 9.6%, 8.2%, 13.4%). The advantage is particularly large on Search-o1, where DR-RAG yields +30.4 ΔEM on HotpotQA and +18.4 on 2Wiki, compared to +12.7 and +7.8 for Rerun. Step-wise retry performs worst among baselines (3.9%–5.5% RR), as the weaker backbone struggles to self-verify reasoning steps without explicit error-type guidance.

Error Type	Correct	Total	Accuracy
Format Error	357	376	94.9%
Reasoning Error	297	420	70.7%
Retriever Error	1,455	1,633	89.1%
Search Error	435	702	62.0%
Overall	**2,544**	**3,131**	**81.3%**

Table 7: Per-type diagnosis accuracy of the distilled diagnosis model evaluated against Qwen3.5-Plus-annotated diagnostic labels.

Compared to Qwen3-8B (Table 2), DR-RAG with Qwen3-4B shows only moderate degradation (e.g., 18.1% vs. 21.8% on ReAct), while baselines degrade more sharply (Rerun: 9.6% vs. 12.2%; RAG-Critic: 7.6% vs. 13.9%). This indicates that DR-RAG's structured diagnosis-then-repair paradigm is more robust to backbone capacity reduction than unstructured retry strategies, as the taxonomy and localization provide explicit guidance that compensates for weaker generation ability.

D.2 Robustness Analysis of the Distilled Diagnosis Model

A natural question for diagnosis-guided repair systems is how sensitive end-to-end performance is to diagnosis accuracy. We conduct a systematic robustness analysis across three dimensions: per-type diagnosis accuracy, stratified repair rates, and a controlled re-routing experiment.

D.2.1 Human-Annotated Diagnosis Validation

We conduct a human-annotated diagnosis experiment to further verify the reliability of the diagnosis model under independent human labels.

Specifically, we stratified and randomly sampled EM=0 failed trajectories covering **10%** of the HotpotQA test scale according to the overall error-type distribution, and conducted blind human annotation. Annotators only saw the question, ground-truth answer, model answer, and full step-by-step trajectory; they did not see the teacher label or the diagnosis model prediction.

The results show that the diagnosis accuracy under human labels is very close to that under teacher labels. We also report Cohen's κ to measure agreement beyond chance:

Reference label	Diagnosis accuracy	Cohen κ (vs. human)
Teacher labels	**82.0%**	—
Human labels (independent blind annotation)	**80.0%**	0.679

The diagnosis model achieves **80.0%** accuracy against independent human labels, **2.0 points** below its accuracy against teacher labels.

As a cross-check, evaluation with an independent model from a different family than the teacher, **GPT-5.5**, yields a diagnosis accuracy of **80.0%** on the same sample set.

D.2.2 Diagnosis Accuracy Breakdown

Table 7 reports the per-type classification accuracy of the distilled diagnosis model, evaluated against Qwen3.5-Plus-annotated Team (2026) diagnostic labels across 3,131 valid samples (3 baselines $\times$ 3 datasets, excluding samples with invalid model outputs). The model achieves 81.3% overall accuracy, with strong performance on Format Error (94.9%) and Retriever Error (89.1%). The primary source of confusion lies between Retriever Error and Search Error, whose boundary depends on distinguishing semantically appropriate queries that fail to retrieve sufficient evidence from queries flawed by upstream reasoning—a distinction that is inherently ambiguous. Figure 3 presents the full confusion matrix.

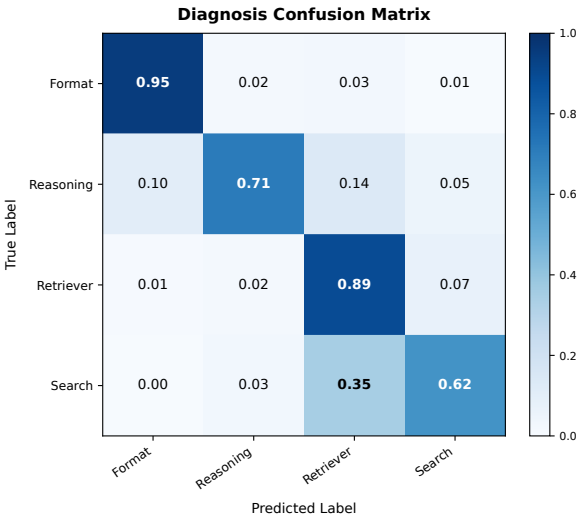


Figure 3: Confusion matrix of the distilled diagnosis model (row-normalized). The primary confusion occurs between Retriever Error and Search Error.

Error Type	Diag. Acc.	Correct	Wrong	Δ
Format	94.9%	41.3%	11.1%	+30.2
Reasoning	70.7%	26.5%	15.8%	+10.7
Retriever	89.1%	17.7%	12.2%	+5.5
Search	62.0%	16.5%	7.2%	+9.3
Overall	**81.3%**	**22.1%**	**10.5%**	**+11.7**

Table 8: Repair rates stratified by diagnosis correctness. "Correct" and "Wrong" denote the repair rates when the diagnosis model correctly or incorrectly classified the error type. Δ shows the gap in percentage points.

D.2.3 Stratified Repair Rates under Correct vs. Incorrect Diagnosis

To understand how diagnosis quality affects downstream repair, we stratify repair outcomes by whether the diagnosis was correct or incorrect (Table 8 and Figure 4). Overall, correctly-diagnosed samples achieve a 22.1% repair rate compared to 10.5% for misdiagnosed samples, a gap of +11.7pp. Importantly, all four error types exhibit positive deltas, confirming that correct diagnosis consistently improves repair effectiveness.

The diagnosis advantage is most pronounced for Format Errors (+30.2pp), where correct identification enables targeted, lightweight answer reformatting. For Retriever Error and Search Error, the gap is smaller (+5.5pp and +9.3pp, respectively), suggesting that repair strategies for these adjacent categories have some functional overlap.

D.2.4 Controlled Re-routing Experiment

We further isolate the causal effect of diagnosis errors through a controlled experiment. For the 587 misdiagnosed samples, we re-run the full repair pipeline with Qwen3.5-Plus-annotated error labels, holding all other variables constant—same repair model, retriever, and evidence context. Table 9 reports the results.

Correcting the routing for misdiagnosed samples yields a modest overall improvement (17.2% → 19.8%, +2.6pp). The improvement is most pronounced for Reasoning Errors (+5.7pp) and Format Errors (+5.3pp), where the correct repair strategy is substantially different from the misrouted one. For Retriever Error and Search Error, the gap is minimal (+2.2pp and +1.1pp), consistent with

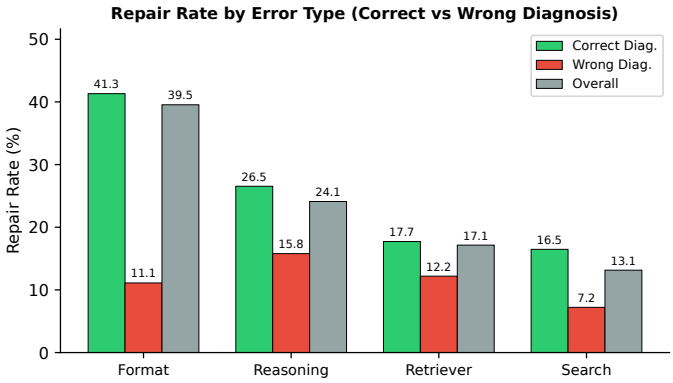


Figure 4: Repair rates by error type, stratified by diagnostic correctness. Correct diagnosis consistently yields higher repair rates across all error categories.

Error Type	Wrong Route	Correct Route	n
Format	10.5%	15.8%	19
Reasoning	33.3%	39.0%	123
Retriever	16.9%	19.1%	178
Search	10.5%	11.6%	267
Overall	**17.2%**	**19.8%**	**587**

Table 9: Controlled re-routing experiment on 587 misdiagnosed samples. "Wrong Route" shows repair rates under the original (incorrect) diagnosis; "Correct Route" shows rates after re-running repair with Qwen3.5-Plus-annotated labels.

their shared query-rewriting repair paradigm. Notably, 62% (364/587) of misroutes occur between Retriever and Search categories, explaining why the overall gain is modest despite correct routing being consistently beneficial.

D.2.5 Discussion

Our analysis reveals several important insights into the behavior and robustness of the DR-RAG framework.

First, correct diagnosis provides a substantial and consistent benefit: across all error types, correctly-diagnosed samples achieve higher repair rates (+11.7pp overall, Table 8). This validates the core premise of diagnosis-guided repair—accurate error classification enables targeted repair strategies that outperform generic approaches.

Second, the controlled re-routing experiment (Table 9) confirms that correct routing improves outcomes, but the gain is modest (+2.6pp overall). This demonstrates *graceful degradation*—the system maintains meaningful repair even under imperfect diagnosis. The modest gap can be explained by three factors:

- **Redundancy in repair strategies:** Retriever Error and Search Error share a common evidence-augmented repair paradigm (query rewriting + re-retrieval). 62% of misroutes between these categories still execute functionally similar repair operations.
- **Cross-effectiveness of repair strategies:** Some repair strategies are partially effective for multiple error types. For example, the retrieval-augmented repair can sometimes fix reasoning issues by providing better evidence.
- **Targeted benefit for distinct categories:** The routing correction benefit is concentrated in Reasoning (+5.7pp) and Format (+5.3pp) errors, where the repair strategies are fundamentally different from the misrouted ones.

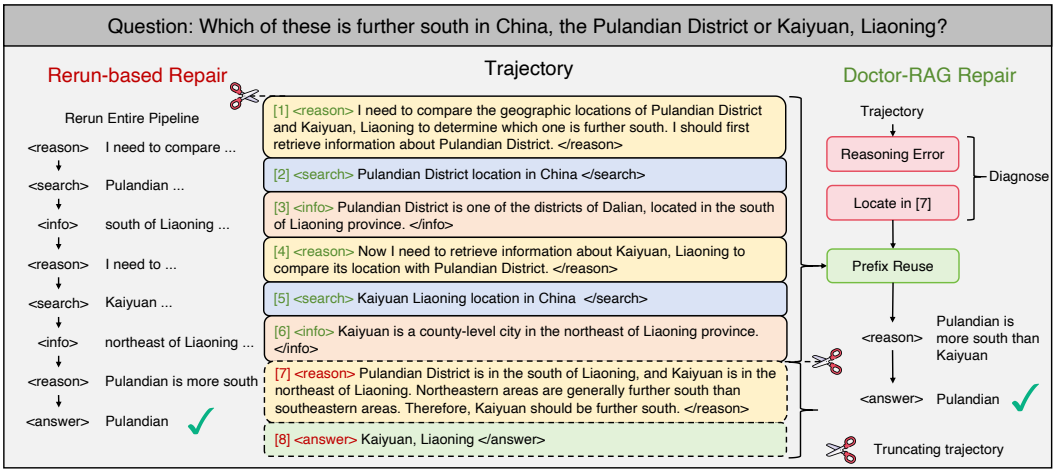


Figure 5: Case study of a reasoning logic error under full evidence coverage. DR-RAG localizes the earliest faulty reasoning step and repairs the error by reusing retrieved evidence, avoiding unnecessary retrieval and full trajectory regeneration.

These factors collectively demonstrate that DR-RAG achieves robust performance even when the finer-grained distinction between errors is incorrect. At the same time, precise diagnosis remains valuable—particularly for Format Errors, where correct classification increases the repair rate by +30.2pp (Table 8), and for Reasoning Errors, where correct routing yields +5.7pp improvement (Table 9). This highlights the architectural design choice of grouping functionally related error types under shared repair paradigms, ensuring resilience to diagnostic errors while maximizing the benefit of accurate classification in critical scenarios.

D.3 Case Study

We present a representative example from HotpotQA to illustrate how DR-RAG repairs reasoning errors under full evidence coverage. The question asks: *"Which of these is further south in China, the Pulandian District or Kaiyuan, Liaoning?"* The agent retrieves the correct documents for both locations, indicating full evidence coverage. However, it incorrectly answers *"Kaiyuan, Liaoning"*, despite the retrieved evidence stating that Pulandian District is located in the south of Liaoning province, while Kaiyuan is in the northeast.

According to our taxonomy, this failure is classified as a reasoning logic error under full coverage, where the error originates from incorrect reasoning rather than missing evidence. Restart-based baselines respond to this failure by regenerating the entire retrieval–reasoning trajectory, even though the retrieved documents contain sufficient information to answer the question.

In contrast, DR-RAG diagnoses the earliest reasoning failure and performs localized repair by reusing the retrieved evidence and recomputing only the affected reasoning steps. This targeted intervention corrects the error without additional retrieval and avoids unnecessary recomputation, demonstrating the effectiveness of diagnosis-guided localized repair.

D.4 Full-Test-Set Performance and Amortized Cost

We report **full-test-set end-to-end EM/F1** and **amortized per-query repair compute overhead**. These results are computed from the **same repaired outputs** as the main table; the only differences are the reporting convention (full test set vs. failed subset) and the added cost columns.

The main results table reports per-method Δ metrics **on the failed (EM=0) subset** to isolate the repair effect, whereas the tables below report **absolute end-to-end EM/F1 over the full test set** (N=500), combining the untouched originally-correct (EM=1) portion with the repaired outcomes of the EM=0 portion. Because the failed subset is only a fraction of the full test set (285–435 of 500 per

cell), the full-test-set gains are the subset improvements diluted by this fraction; the two tables thus present the same underlying results under different denominators.

D.4.1 EVALUATION METRICS

Per baseline×dataset, each table reports the **end-to-end EM/F1 over the full test set** (N=500) together with two efficiency columns:

- **Avg ΔEM** — end-to-end EM gain over `Base`, averaged over datasets.
- **Tok/q (K)** — **amortized** repair tokens per query: total repair-stage tokens (input+output) divided by the full test set (N=500), in thousands. Lower is better. (Diagnosis uses a distilled *no-think* model, ~28 tok/sample, negligible vs. repair; cost is therefore reported on the repair stage.)
- **ΔEM/1K tok** — cost-effectiveness: end-to-end ΔEM per 1K amortized tokens. Higher is better.

D.4.2 QWEN3-8B

Baseline	Method	Avg ΔEM ↑	Tok/q (K) ↓	ΔEM/1K tok ↑	HotpotQA EM	HotpotQA F1	2Wiki EM	2Wiki F1	MuSiQue EM	MuSiQue F1
ReAct	Base	–	–	–	27.2	34.7	29.4	34.4	13.2	19.9
ReAct	Rerun	11.7	6.23	1.98	44.6	53.7	41.0	49.3	19.2	27.3
ReAct	Step-wise	8.5	7.66	1.19	39.6	49.1	38.8	48.4	16.8	23.7
ReAct	RAG-Critic	13.3	12.04	1.18	45.2	53.3	42.8	48.9	21.8	28.0
ReAct	**DR-RAG**	**19.9**	**3.97**	**5.39**	**55.6**	**62.6**	**49.6**	**56.3**	**24.2**	**30.7**
Search-o1	Base	–	–	–	43.0	51.0	33.2	40.2	14.4	21.5
Search-o1	Rerun	4.9	4.79	1.09	49.6	57.9	36.6	46.1	19.0	25.0
Search-o1	Step-wise	5.0	5.19	1.12	49.6	57.8	38.6	47.8	17.4	24.8
Search-o1	RAG-Critic	6.7	13.92	0.50	49.2	57.5	39.4	46.2	22.0	30.0
Search-o1	**DR-RAG**	**16.5**	**3.65**	**4.84**	**60.4**	**66.2**	**50.8**	**57.6**	**29.0**	**37.1**
Search-R1	Base	–	–	–	37.0	45.4	32.4	38.4	13.0	19.9
Search-R1	Rerun	11.5	4.82	2.54	49.2	58.5	48.0	54.3	19.6	28.1
Search-R1	Step-wise	4.9	4.78	1.09	41.8	51.1	41.0	48.7	14.4	21.9
Search-R1	RAG-Critic	8.5	11.76	0.75	44.4	53.5	42.6	47.7	20.8	28.6
Search-R1	**DR-RAG**	**13.0**	**3.66**	**3.98**	**52.2**	**61.1**	**48.8**	**54.0**	**20.4**	**28.0**

D.4.3 LLAMA-3.1-8B

Baseline	Method	Avg ΔEM ↑	Tok/q (K) ↓	ΔEM/1K tok ↑	HotpotQA EM	HotpotQA F1	2Wiki EM	2Wiki F1	MuSiQue EM	MuSiQue F1
ReAct	Base	–	–	–	27.2	34.7	29.4	34.4	13.2	19.9
ReAct	Rerun	10.3	11.58	1.08	44.6	52.8	38.8	47.0	17.4	24.4
ReAct	Step-wise	9.6	16.44	0.63	43.4	52.3	37.4	47.1	17.8	24.6
ReAct	RAG-Critic	7.9	6.25	1.36	39.6	46.9	36.8	42.7	17.2	22.1
ReAct	**DR-RAG**	**14.5**	**3.92**	**3.99**	**51.8**	**58.8**	**40.4**	**45.6**	**21.0**	**27.0**
Search-o1	Base	–	–	–	43.0	51.0	33.2	40.2	14.4	21.5
Search-o1	Rerun	3.6	8.42	0.51	49.0	56.0	35.2	43.8	17.2	25.0
Search-o1	Step-wise	4.2	11.55	0.38	48.4	55.4	36.4	45.1	18.4	25.9
Search-o1	RAG-Critic	2.9	6.68	0.50	47.2	54.5	36.0	43.1	16.2	23.1
Search-o1	**DR-RAG**	**10.4**	**3.40**	**3.50**	**57.6**	**62.6**	**40.4**	**45.1**	**23.8**	**28.9**
Search-R1	Base	–	–	–	37.0	45.4	32.4	38.4	13.0	19.9
Search-R1	Rerun	4.7	6.91	0.71	42.6	52.7	37.4	44.7	16.4	24.5
Search-R1	Step-wise	3.7	9.51	0.40	42.2	51.4	35.8	44.2	15.4	22.7
Search-R1	RAG-Critic	1.8	5.23	0.37	39.0	46.2	34.8	39.9	14.0	19.4
Search-R1	**DR-RAG**	**8.5**	**3.46**	**2.76**	**47.8**	**54.9**	**41.6**	**45.3**	**18.4**	**23.7**

D.4.4 TAKEAWAYS

1. **Full-test-set effectiveness.** DR-RAG achieves the largest dataset-averaged end-to-end EM gain in every backbone×baseline setting (Qwen3-8B: +13.0 to +19.9; LLaMA-3.1-8B: +8.5 to +14.5 Avg ΔEM).
2. **Repair cost and cost-effectiveness.** On both backbones, DR-RAG uses the fewest amortized repair tokens per query among the compared methods (Qwen 3.65–3.97K; LLaMA 3.40–3.92K) and achieves the highest reported cost-effectiveness (Qwen 3.98–5.39; LLaMA 2.76–3.99 ΔEM per 1K tokens).

D.5 REPAIR TOKEN AND LLM-CALL COSTS

We report token counts and LLM-call counts to provide a more fine-grained analysis of repair efficiency from the perspective of model-side computation. The numbers correspond to the **repair operation** performed by each method.

Averaged per repaired query, over all 9 (baseline × dataset) settings:

Method	Tokens/query (Qwen3-8B)	Tokens/query (Llama-3.1-8B)	LLM calls/query
Rerun	7.2 K	12.3 K	2.2
Step-wise	8.0 K	17.1 K	4.5
RAG-Critic	17.2 K	8.3 K	5.3
DR-RAG (ours)	**5.4 K**	**4.9 K**	**1.9**

Takeaways.

- The token and LLM-call results are consistent with the wall-clock results and further support the efficiency advantage of DR-RAG. On both repair models, DR-RAG uses the **fewest tokens and the fewest LLM calls**. For example, on Qwen3-8B, RAG-Critic uses about **3.2**× the tokens and **2.8**× the LLM calls of DR-RAG, showing that its additional planning also corresponds to real model-side computation cost.
- DR-RAG's repair process is **targeted and bounded** (typically 1–2 LLM calls; only the retrieval-planning branch is multi-turn and hard-capped at 4). This is because DR-RAG repairs from the localized failure point rather than re-processing the entire trajectory, while still achieving higher repair rates (Table 2).

D.6 DEPLOYMENT SIMULATION WITHOUT GOLD ANSWERS

We evaluate a deployment simulation that removes the gold answer entirely at inference. We use a heterogeneous API **verifier** (`deepseek-v4-pro`), which differs from the teacher `qwen3.5-plus` used for distillation. The verifier sees **only** `question + retrieved documents + predicted answer` — **never** the gold answer or gold evidence — and outputs two signals:

- a **correctness judgment** on the agent's answer → replacing the gold-based `EM=0` gate as the **failure trigger**;
- its **own answer** from the retrieved evidence → serving as a **pseudo-GT** that replaces the gold answer when conditioning diagnosis.

Gold labels are used **only for post-hoc evaluation**, never as method input. The experiment covers 3 agentic baselines (ReAct / Search-o1 / Search-R1) × 3 datasets (HotpotQA / 2Wiki / MuSiQue), with 500 samples per setting (**4,500 total**). The diagnosis model is the same distilled model as in the main paper; the only change is that the inference-time answer signal is produced by the verifier rather than provided by the gold answer.

(a) Failure detection. Over 4,500 samples, the verifier's failure-detection quality against true EM=0 is:

Precision	Recall	F1	Accuracy
0.867	0.761	**0.811**	0.741

False negatives are concentrated in evidence-insufficient, harder MuSiQue cases.

(b) End-to-end performance on the full test set. We run the full pipeline `verifier trigger -> pseudo-GT diagnosis -> repair` and score every 500-sample cell in full, so both verifier misses and false positives are counted in the final result:

	Baseline	DR-RAG (verifier-driven)	Δ
EM	27.0	**34.0**	**+7.1**
F1	33.9	**42.3**	**+8.4**

On the triggered subset (2,130 samples), repair-rate is **24.8%**, ΔEM is **+14.9**, and ΔF1 is **+17.6**. Breaking this down: among 1,920 true failures, 419 are fixed (21.8%); among 210 originally correct answers that were falsely triggered, 101 are broken. Even after this false-positive cost, the net gain is **+318 EM = +7.1 points**. This shows that verifier-driven DR-RAG remains effective in a deployment simulation without gold answers.

D.7 Retrieval-Budget-Controlled Comparison

Among DR-RAG's four repair operators, **only the retriever-failure operator increases retrieval** — it is triggered exactly on samples diagnosed as "the retriever missed the evidence," where fetching more is the intended fix. The format- and reasoning-repair operators perform **no** retrieval, and the search operator uses the **same** top-k and turn limit as the baselines.

To evaluate performance under matched retrieval limits, we **aligned the retrieval limits** across all methods. Concretely, for every method we fix top-k = 5 and max turns = 4, and we cap the retriever-failure operator to a **single rewritten query** — so one repair intervention retrieves exactly as many documents as a single baseline search (1 query $\times$ 5 docs). We measure budget as documents retrieved per sample:

$$\text{docs/sample} = \sum_i (k_i \times n_i)\,,$$

where k_i is the top-k value for query group i, and n_i is the number of queries in that group.

Comparison with matched retrieval limits on ReAct trajectories (Qwen3-4B backbone, repaired by Qwen3-8B) over HotpotQA / 2WikiMultiHopQA / MuSiQue. Per-sample values are averaged over the three datasets; RepairRate / ΔEM / ΔF1 are computed on the EM=0 failure subset.

Method	retrieval calls	docs / sample	RepairRate	ΔEM	ΔF1
DR-RAG (ours)	**0.86**	**4.3**	**23.78%**	**+23.78%**	**+0.240**
Rerun (self-diagnosed)	1.09	5.5	15.73%	+15.73%	+0.190
Step-wise	0.89	4.4	13.47%	+13.47%	+0.165
RAG-Critic	1.05	5.3	13.12%	+13.12%	+0.135

Two conclusions follow:

1. **DR-RAG achieves the highest repair rate and EM gain under matched retrieval limits.** It outperforms the strongest baseline by **+8.1 points EM** (+23.8% vs. +15.7%). These results support the effectiveness of diagnosis-guided repair under the specified retrieval limits.
2. **DR-RAG has the lowest average retrieval cost among the compared methods**: 0.86 calls and 4.3 documents per sample. The format- and reasoning-repair operators retrieve no documents, while the retriever-failure operator uses a single rewritten query under this setting.

D.8 Integrated versus Standalone Coverage Prediction

We conduct an ablation against a standalone coverage predictor. Both conditions use the same distilled diagnosis model and the same inputs, varying only the decision structure:

- **Integrated (ours):** a single call jointly judges evidence sufficiency and classifies the error over all four types.
- **Standalone predictor:** a separate module first predicts coverage (sufficient / insufficient), then routes the sample to a restricted diagnosis space (sufficient -> {Format, Reasoning}; insufficient -> {Format, Retriever, Search}).

We evaluate on qwen3-4b trajectories across 3 baselines $\times$ 3 datasets. Repair is reported as improvement over the original failed trajectory (ΔEM / ΔF1 / ΔR-L), consistent with the main results table. The **LLM-call and token columns count only the diagnosis stage**; the downstream repair operators are identical in both conditions and are therefore excluded from this comparison.

Design	Diag. Acc.	Repair Rate	ΔEM	ΔF1	ΔR-L	Diagnosis LLM calls/sample	Diagnosis tokens/sample
Integrated (ours)	**81.3**	**20.1**	**20.1**	**18.1**	**17.9**	**1**	**6.6k**
Standalone coverage predictor	74.8	19.8	19.8	17.9	17.6	2 (coverage + diagnosis)	8.8k

The two designs deliver **nearly identical downstream repair quality** (Repair Rate 20.1 vs. 19.8; ΔEM/ΔF1/ΔR-L within 0.3 points), but the integrated design uses **half the diagnosis-stage LLM calls** (1 vs. 2), **~25% fewer diagnosis tokens**, and achieves higher diagnosis accuracy (81.3 vs. 74.8). Thus, decoupling sufficiency into a standalone predictor does not improve repair quality and instead increases diagnosis cost.

The gap stems from routing errors: with 82.2% coverage accuracy, the standalone predictor misroutes about 18% of samples and excludes the correct error type for 11.3%. The integrated design avoids this premature restriction by jointly predicting sufficiency and error type.

D.9 Multi-Seed Stability and Statistical Significance

We conduct a multi-seed analysis in the main setting: we run the repair stage with **3 random seeds**, report **mean** $\pm$ **std** for the main metrics (Repair Rate, ΔEM, ΔF1), and perform paired bootstrap significance tests between DR-RAG and all baselines (Rerun, Step-wise, and RAG-Critic) on shared instances.

The results show that DR-RAG has low seed-to-seed variance: across the 9 framework$\times$dataset settings, the standard deviation of Repair Rate is only $\pm$**0.2–2.3 points**, much smaller than the performance gaps to the main baselines.

D.9.1 DR-RAG — Repair Rate, ΔEM and ΔF1 (mean $\pm$ std over 3 seeds)

Framework	Dataset	Repair Rate (mean $\pm$ std)	ΔEM (mean $\pm$ std)	ΔF1 (mean $\pm$ std)
ReAct	HotpotQA	0.308 ± 0.011	0.308 ± 0.011	0.278 ± 0.010
ReAct	2Wiki	0.235 ± 0.013	0.235 ± 0.013	0.228 ± 0.015
ReAct	MuSiQue	0.107 ± 0.012	0.107 ± 0.012	0.088 ± 0.007
Search-o1	HotpotQA	0.301 ± 0.021	0.301 ± 0.021	0.226 ± 0.016
Search-o1	2Wiki	0.201 ± 0.006	0.201 ± 0.006	0.154 ± 0.009
Search-o1	MuSiQue	0.156 ± 0.023	0.156 ± 0.023	0.129 ± 0.028
Search-R1	HotpotQA	0.224 ± 0.012	0.224 ± 0.012	0.242 ± 0.009
Search-R1	2Wiki	0.273 ± 0.016	0.273 ± 0.016	0.260 ± 0.017
Search-R1	MuSiQue	0.089 ± 0.002	0.089 ± 0.002	0.098 ± 0.011

D.9.2 PAIRED SIGNIFICANCE — DR-RAG MINUS BASELINE (ΔREPAIR RATE; SIG = # OF 3 SEEDS WITH P<0.05)

Framework	Dataset	vs. Rerun: ΔRR	sig	vs. Step-wise: ΔRR	sig	vs. RAG-Critic: ΔRR	sig
ReAct	HotpotQA	+0.173	3/3	+0.199	3/3	+0.219	3/3
ReAct	2Wiki	+0.108	3/3	+0.118	3/3	+0.140	3/3
ReAct	MuSiQue	+0.073	3/3	+0.081	3/3	+0.077	3/3
Search-o1	HotpotQA	+0.173	3/3	+0.184	3/3	+0.160	3/3
Search-o1	2Wiki	+0.126	3/3	+0.126	3/3	+0.161	3/3
Search-o1	MuSiQue	+0.106	3/3	+0.129	3/3	+0.117	3/3
Search-R1	HotpotQA	+0.083	3/3	+0.154	3/3	+0.154	3/3
Search-R1	2Wiki	+0.049	1/3	+0.140	3/3	+0.170	3/3
Search-R1	MuSiQue	+0.029	1/3	+0.065	3/3	+0.074	3/3

The significance tests support the main conclusion: DR-RAG significantly outperforms Step-wise and RAG-Critic in all 9 settings, and significantly outperforms Rerun in 7/9 settings.

D.10 SENSITIVITY TO THE REPAIR RETRIEVER

We replace the dense retriever used during repair (`bge-base-en-v1.5`) with a weaker sparse BM25 retriever while keeping the diagnosis model, repair model, failure set, and evaluation protocol unchanged.

Repair Rate (%) on Qwen3-4B error trajectories is:

Baseline	Retriever	HotpotQA	2Wiki	MuSiQue	**Avg**
ReAct	Dense (bge)	31.6	26.0	7.7	**21.8**
ReAct	BM25	31.1	25.0	7.4	21.2
Search-o1	Dense (bge)	29.8	19.5	18.1	**22.5**
Search-o1	BM25	28.3	20.7	17.3	22.1
Search-R1	Dense (bge)	25.4	25.8	8.3	**19.8**
Search-R1	BM25	24.1	25.5	9.0	19.5
Overall	Dense (bge)	—	—	—	**21.4**
Overall	BM25	—	—	—	20.9

Although BM25 is weaker than the dense retriever, DR-RAG's overall Repair Rate changes only from **21.4%** to **20.9%** (-0.5 pp), and the per-setting differences stay within **[$-$1.5, +1.2] pp**. This indicates that DR-RAG's diagnosis-guided localized repair does not depend on a specific dense retriever and remains stable under weaker retrieval.

D.11 ERROR DISTRIBUTION AND REPAIR TRENDS

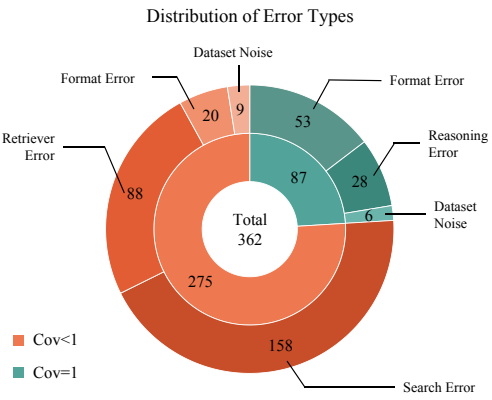


Figure 6: Statistics of different error types in ReAct-based Agentic RAG on HotpotQA (500 samples).

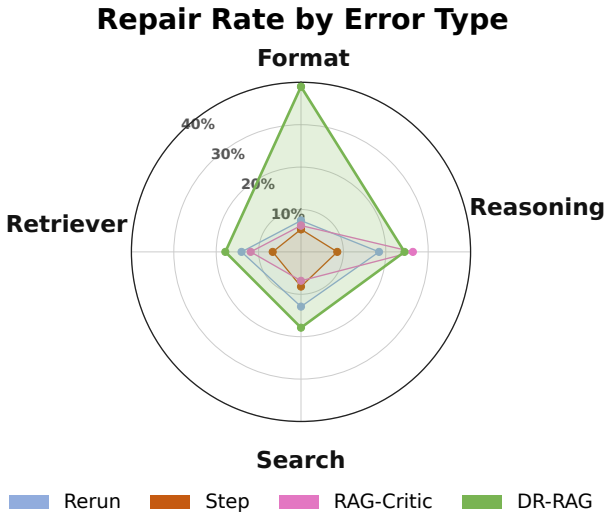


Figure 7: Comparison of average repair rates across error types, aggregating results from three baselines under repair strategies.

E DIAGNOSIS ALGORITHM DETAILS

This appendix provides the detailed implementation of the DR-RAG diagnosis module. DR-RAG uses a backbone-specific diagnosis model that jointly performs evidence sufficiency assessment, error classification, and failure localization. Rather than relying on a separate coverage predictor to gate the candidate error space, DR-RAG integrates evidence sufficiency reasoning directly into the diagnostic prompt, allowing the model to implicitly determine whether retrieved evidence is sufficient as part of its classification process.

For each backbone setting, the diagnosis model is fine-tuned via SFT to output a structured diagnosis including both the error type and the localized failure point. The prompt template defines the diagnostic protocol—including the evidence sufficiency check as a branching point—and the output format, while the final classification and localization are produced by the fine-tuned student model.

We first present the complete diagnosis pipeline in Algorithm 1, which connects teacher annotation, student distillation, inference-time diagnosis, and repair routing. We then describe the trajectory representation, the diagnosis logic with its four error types, admissible behavior for each component, error-type criteria, query-level judgement, and the failure localization interface. The key innovation is that the model itself determines evidence sufficiency: if retrieved evidence is sufficient but reasoning fails, the error is *Reasoning Error*; if evidence is insufficient due to retriever failure or flawed query generation, the error is *Retriever Error* or *Search Error*, respectively.

E.1 OVERALL DIAGNOSIS PIPELINE

Algorithm 1 summarizes the complete diagnosis pipeline of DR-RAG. The pipeline consists of two stages. The first stage constructs SFT supervision using a teacher model that receives the ground-truth answer but not golden evidence. The second stage deploys the distilled student model with the same prompt structure (including the ground-truth answer) to predict the error type and localized failure point. A key design choice is that the teacher and student prompts are fully aligned: both receive the question, trajectory, and ground-truth answer, and follow the same diagnostic logic. The teacher's advantage lies only in its stronger reasoning capability (Qwen3.5-Plus in thinking mode), not in additional oracle information.

E.2 PROBLEM SETUP AND TRAJECTORY REPRESENTATION

Given a question q, an agentic RAG system produces a trajectory y that interleaves reasoning steps, search actions, retrieved evidence, and the final answer. DR-RAG performs diagnosis on failed trajectories where the original answer does not match the ground-truth answer under exact match evaluation, i.e., $\mathrm{EM} = 0$.

To support fine-grained fault localization, the raw trajectory is parsed into indexed components:

- `[Step` k`] (Reasoning)`: an intermediate reasoning or planning step;
- `[Action` j`] (Search)`: a retrieval query issued by the agent;
- `[Evidence` j`] (Information)`: the retrieved content returned by the retriever (paired with Action j);
- `[Step` k`] (Answer)`: the final answer generation step.

Reasoning and answer steps share the index k, while search actions use a separate action index j. This representation allows the diagnosis model to return a culprit step label such as `culprit_step_label = "[Step 3] (Reasoning)"` or `culprit_step_label = "[Action 2] (Search)"`. The indexed trajectory also provides the repair module with a concrete intervention point, enabling reuse of the valid prefix before the localized failure.

E.2.1 GLOBAL ACTION INDEX AND DISPLAY LABELS

Our formulation has only one unified action-level index, and the localized failure point $k^\dagger$ is defined on this index. As in Eq. (1), a trajectory is a temporally ordered, interleaved action stream:

$$y = \{\rho^{(1)}, \rho^{(2)}, \ldots, \rho^{(K)}, a_{\mathrm{pred}}\}, \qquad \rho^{(k)} \in \{r, q, d\},$$

Algorithm 1: Diagnosis Pipeline of DR-RAG

Input: Failed trajectory set $\mathcal{D}$ for training; a failed test trajectory (q, y) for inference.
Output: Diagnosis result $(c, k^\dagger)$, where c is the error type and $k^\dagger$ is the localized failure point.

Stage I: Teacher Annotation and Distillation;
Initialize SFT dataset $\mathcal{S} \leftarrow \emptyset$;
foreach *failed training instance* $d = (q, y, a^*) \in \mathcal{D}$ **do**
  Parse trajectory y into indexed reasoning steps, search actions, evidence blocks, and final answer;
  Construct the diagnosis prompt with (q, y, a^*);
  Run the teacher model to obtain a diagnostic rationale r and structured label j;
  Construct one SFT example: input = diagnosis prompt with (q, y, a^*); output = $\langle$think$\rangle r \langle$/think$\rangle j$;
  Add the example to $\mathcal{S}$;
Fine-tune the student diagnosis model on $\mathcal{S}$;

Stage II: Student Diagnosis at Inference Time;
Parse the failed test trajectory y into indexed components;
Construct the diagnosis prompt with (q, y, a^*);
The student performs a diagnosis following the prompt's hierarchical logic:
Phase 1: Check if final answer is semantically equivalent to GT $\Rightarrow$ *Format Error*;
Phase 2: Assess evidence sufficiency from the trajectory's [Evidence] blocks:;
if *evidence is sufficient to derive the GT answer* **then**
  Phase 3: Locate the earliest divergent reasoning step $\Rightarrow$ *Reasoning Error*;
else
  Phase 4: Perform entity provenance check on each search query;
  if $\exists$ *valid query with failed retrieval* **then**
    Set c to *Retriever Error* and $k^\dagger$ to the failed evidence block;
  else
    Set c to *Search Error* and $k^\dagger$ to the first poisoned search action;
if c *is Format Error* **then**
  Set $k^\dagger \leftarrow$ `null`;
Return $(c, k^\dagger)$ as the control signal for local repair;

where reasoning (r), search-query (q), and information/evidence (d) actions all share **one** running index $k = 1, \ldots, K$. The diagnosis output $(c, k^\dagger)$ and the prefix used for repair, $y_{\text{prefix}} = \{\rho^{(1)}, \ldots, \rho^{(k^\dagger - 1)}\}$, are both defined on this unified index. Localization is therefore performed at the **action level**, consistent with Eq. (1).

For readability, the trajectory display uses type-local display indices: `[Step k] (Reasoning)` denotes the k-th reasoning step; `[Action j] (Search)` denotes the j-th retrieval action; `[Evidence j] (Information)` denotes the evidence returned by that retrieval and therefore shares `j` with `[Action j]`; `[Step k] (Answer)` denotes the final answer generation step, and because the answer appears only once at the end of the trajectory, the display label reuses the final reasoning-step index `k`. These are display aliases, not the global action index in Eq. (1). For Reasoning Error, localization selects the first erroneous reasoning action; for Retriever Error, it selects the evidence block d returned by a valid query that fails to retrieve the required evidence; for Search Error, it selects the first poisoned search action q. Format Error does not require failure localization: the localization field is `null`, and repair targets the final answer.

E.3 Distilled Diagnosis Model

The diagnosis module is implemented as a distilled diagnosis model. Specifically, for each backbone setting, we fine-tune the corresponding student diagnosis model via supervised fine-tuning (SFT) to perform error classification and failure localization. The training process follows a teacher–student distillation pipeline.

First, a stronger teacher model, Qwen3.5-Plus in thinking mode, annotates failed trajectories using the same diagnostic prompt as the student. During annotation, the teacher receives the question, parsed trajectory, and ground-truth answer—the same input as the student. The teacher does *not* receive golden evidence or golden context. Its advantage over the student lies solely in its stronger reasoning

Diagnosis Output Schema

Output (all error types):

```
{"assigned_error_type": "...",
 "culprit_step_label": "..."}
```

`assigned_error_type` ∈ {Format-Invalid Answer Error, Reasoning Logic Error, Retriever Failure, Reasoning-Induced Missing Retrieval}.
`culprit_step_label`: the label of the first erroneous step (e.g., "[Step 2] (Reasoning)", "[Action 1] (Search)", or "[Evidence 2] (Information)"). Set to `null` for Format Error.

Figure 8: Structured output schema of the diagnosis model. A single format covers all four error types.

capability, which produces higher-quality diagnostic rationales. This prompt-aligned design ensures that the distilled reasoning traces are directly compatible with the student's inference-time input, avoiding information leakage.

Second, the teacher output is converted into SFT data. Each SFT example uses the diagnostic prompt (with question, trajectory, and ground-truth answer) as input. The output contains the teacher's diagnostic rationale (extracted from thinking mode) followed by a compact JSON label. Figure 8 illustrates the structured output schema.

Third, the student model is trained to reproduce the teacher's diagnostic behavior. At inference time, the student receives the question q, trajectory y, and ground-truth answer a^*, and outputs the predicted error type and failure location. This design separates training-time distillation (where the teacher's stronger reasoning produces reliable labels) from inference-time diagnosis (where the student applies the same prompt independently).

We train the student model with full-parameter SFT using LLaMA-Factory (Zheng et al., 2024). The learning rate is 5×10^{-6} with a cosine scheduler and 10% warmup. We train for 3 epochs with batch size 1, gradient accumulation 16, cutoff length 16384, DeepSpeed ZeRO-2, bf16 precision, and `enable_thinking=true`.

E.4 Diagnosis Logic

The diagnosis model covers all four error types within a single prompt. Rather than relying on an external coverage signal to gate the candidate error space, the model itself assesses evidence sufficiency as an intermediate reasoning step. This evidence sufficiency check serves as the key branching point in the diagnostic logic.

The diagnostic process follows a strict priority order:

Phase 1: Format Error. The model first checks whether the final answer is semantically equivalent to the ground-truth answer but fails exact match due to surface-form mismatch (e.g., "27" vs. "27 years old", "Steve Jobs" vs. "Steven Paul Jobs"). If so, the error is classified as *Format Error*.

Phase 2: Evidence Sufficiency Check. If the answer is factually wrong, the model assesses whether the trajectory's `[Evidence]` blocks contain sufficient information to derive the ground-truth answer. This is the key branching point:

- **Evidence sufficient**: the model had the right information but failed to use it correctly ⇒ proceed to Phase 3.
- **Evidence insufficient**: the model was missing critical information ⇒ proceed to Phase 4.

Phase 3: Reasoning Error. When evidence is sufficient, the failure must lie in the reasoning process—specifically, the model reasons incorrectly when *consuming* already-sufficient evidence (e.g., misinterpreting a retrieved fact or ignoring a key passage). The model locates the earliest

Error Type	Classification Criteria	What Constitutes an "Error"
Format Error	Final answer is *semantically equivalent* to GT but fails EM due to surface form, e.g., "27" vs. "27 years old", "Steve Jobs" vs. "Steven Paul Jobs".	Answer is factually correct but string-mismatched.
Reasoning Error	Evidence is sufficient, but the trajectory contains a step where reasoning diverges: hallucination, contradiction, wrong entity, misinterpretation, or ignored evidence.	Sufficient evidence exists, but the model uses it incorrectly.
Retriever Error	Evidence is insufficient. At least one valid query (correct entities from the question or prior evidence) is issued, but the retriever returns irrelevant or incomplete evidence.	The query targets the right information, but the retrieval fails.
Search Error	Evidence is insufficient. Search queries are wrong due to flawed reasoning: hallucinated entities, wrong assumptions, misdirected targets, or excessive vagueness.	The missing evidence is caused by invalid query generation rooted in reasoning errors rather than retriever failure.

Table 10: Error type definitions and classification criteria. The diagnosis model follows a strict priority order with evidence sufficiency as the key branching point.

step where reasoning diverges: hallucination, contradiction, wrong entity/date, misinterpretation, or ignored evidence.

Phase 4: Retrieval Error Classification. When evidence is insufficient, the model must determine *why* the needed information was not retrieved. It performs entity provenance checking on each search query to classify queries as *Valid* or *Poisoned*:

- **Retriever Error**: at least one valid query (correct entities from question or prior evidence) was issued, but the retriever returned irrelevant results.
- **Search Error**: all queries were *poisoned* by flawed reasoning (hallucinated entities, wrong assumptions, misdirected targets), so the retriever was never given a fair chance. Unlike Reasoning Error where flawed reasoning misuses sufficient evidence, here the flawed reasoning manifests in *generating* invalid queries, causing the needed evidence to never be retrieved.

This integrated approach allows the diagnosis model to jointly reason about evidence sufficiency and error type, rather than relying on a separate, potentially noisy coverage predictor.

Shorthand notation. For brevity, we use the following short names throughout the paper: *Format Error*, *Reasoning Error*, *Retriever Error*, and *Search Error*. Their full formal names (Format-Invalid Answer Error, Reasoning Logic Error, Retriever Failure, and Reasoning-Induced Missing Retrieval, respectively) are used only in the error-type definitions and the prompt templates where exact label strings are required by the model output schema. The short form "Search Error" emphasizes that the failure manifests as ineffective search behavior, while the full name clarifies that the root cause lies in upstream reasoning rather than the retriever itself.

E.5 Error Criteria and Query-Level Judgement

We define an action as erroneous only when it violates the admissible behavior of its corresponding component. For reasoning steps, admissible behavior means that each inference is entailed by the question, prior reasoning, or retrieved evidence, and does not introduce unsupported entities or contradictions. For search actions, admissible behavior means that the query uses correct entities from the question or previously retrieved evidence, targets a necessary missing relation or attribute, and is specific enough to give the retriever a fair chance. For retriever behavior, admissible behavior means that a valid query should return evidence relevant to the queried entity or relation; if it fails to do so, the error is attributed to the retriever.

Table 10 summarizes the criteria for each error type. The diagnosis follows a strict priority order: Format Error is checked first, followed by the evidence sufficiency assessment, and then the appropriate sub-classification.

For all cases, the model first checks whether the final answer is semantically equivalent to the ground truth. If so, the error is classified as *Format Error*. Otherwise, the model assesses evidence sufficiency. If evidence is sufficient, the remaining failure is classified as *Reasoning Error*, and the model localizes the earliest divergent reasoning step. If evidence is insufficient, the model performs query-level judgement to distinguish *Retriever Error* from *Search Error*.

Each query is classified as either *Valid* or *Poisoned*. A query is **Valid** if all entities in the query come from the user question or previously retrieved evidence, the query targets information relevant to answering the question, and is specific enough to retrieve the missing evidence. If such a valid query fails to retrieve the needed evidence, the failure is attributed to the retriever.

A query is **Poisoned** if it is generated from flawed reasoning. Typical poisoned queries include queries with hallucinated entities (names, dates, or facts absent from the question and all prior evidence), wrong assumptions (assuming a fact never established in the trajectory), wrong directions (targeting the wrong entity/relationship due to a prior reasoning error), or excessive vagueness. If all search queries are poisoned, the failure is classified as *Search Error*.

An important edge case is that a query may look well-formed but still target the wrong entity because of an earlier incorrect assumption. In this case, the query is treated as poisoned, and the error is classified as *Search Error* rather than *Retriever Error*. The diagnosis requires an *entity provenance check*: for each key entity in a search query, the model verifies whether it originates from the user question or previously retrieved evidence (valid source) or was invented by the model in a reasoning step without evidence support (invalid source).

E.6 Failure Localization and Repair Interface

Beyond assigning an error type, DR-RAG also localizes the earliest failure point. This localization is essential because the repair module reuses the valid prefix before the failure point and only regenerates the affected part of the trajectory.

The diagnosis model returns `culprit_step_label` as the localization field for all error types. If the error is *Reasoning Error*, the culprit is the earliest reasoning step where the trajectory diverges from a valid solution path (e.g., `"[Step 3] (Reasoning)"`). If the error is *Retriever Error*, the culprit is the evidence block returned by a valid search action that fails to contain the necessary information (e.g., `"[Evidence 2] (Information)"`). If the error is *Search Error*, the culprit is the first poisoned search action—i.e., the earliest search query that is misdirected due to flawed upstream reasoning (e.g., `"[Action 2] (Search)"`). For *Format Error*, no localization is needed, and the field is set to `null`.

The final diagnosis output is denoted as $(c, k^\dagger)$, where c is the assigned error type and $k^\dagger$ is the localized failure point. This pair serves as the interface between diagnosis and repair. The error type c selects the repair operator, while the failure location $k^\dagger$ specifies the point from which local repair should be performed.

E.7 Teacher and Student Prompts

Figure 9 presents the diagnosis prompt used for both teacher annotation and student inference. The teacher and student use the same prompt: both receive the question, trajectory, and ground-truth answer, and follow the same diagnostic logic. The teacher's advantage lies only in its stronger reasoning capability (Qwen3.5-Plus in thinking mode), which produces more reliable diagnostic rationales for distillation.

The prompt encodes the complete diagnostic protocol as a four-phase hierarchy: (1) format checking, (2) evidence sufficiency assessment, (3) reasoning error localization (if evidence sufficient), (4) retrieval error classification via entity provenance checking (if evidence is insufficient). The model implicitly determines evidence sufficiency during its reasoning process, eliminating the need for an external coverage signal.

A key design choice is that the ground-truth answer is provided with an explicit *Assumption* statement ("The Ground Truth Answer is CORRECT. Do not question or second-guess it."). This eliminates the need for a separate Dataset Noise category by constraining the model to accept the ground truth as given, focusing the diagnosis entirely on identifying why the model's trajectory failed to produce the correct answer.

F Repair Module Details

This appendix provides the detailed implementation of the DR-RAG repair module, including the algorithmic procedure for each repair operator and the prompt templates used at inference time. Given a diagnosis output $(c, k^\dagger)$—where c is the error type and $k^\dagger$ is the localized failure point—the repair module selects and applies a specialized operator to correct the trajectory with minimal recomputation.

All repair operators share a common design principle: they preserve the conditionally valid prefix $y_{\text{prefix}} = \{\rho^{(1)}, \ldots, \rho^{(k^\dagger-1)}\}$ and intervene only at or after the failure point. The repair model uses the same backbone LLM as the original agentic RAG system under each experimental setting, with prompt-guided generation using vLLM. Token budgets are enforced via truncation with a safety buffer of 50 tokens, and the maximum prompt length is capped at 30,000 tokens.

F.1 Format-Invalid Answer Repair

Format errors occur when the model's answer is semantically correct but fails an exact match due to surface-form differences. Since the trajectory and retrieved evidence are valid, repair is limited to rewriting the final answer.

The repair prompt (Figure 10) instructs the model to:

1. Analyze the dialogue history (including all retrieved documents and reasoning steps) to identify the correct factual answer.
2. Determine whether the exact answer string can be found verbatim in the retrieved documents (Priority 1: document extraction) or must be derived from reasoning (Priority 2: format enumeration).
3. Generate a comprehensive JSON list of all valid exact-match variants (at least 10), covering: alternate surface forms, number formats, article variations, common shortenings, and symbol differences.

The output is a JSON list of candidate strings. During evaluation, if any candidate matches the ground truth under an exact match, the repair is considered successful.

F.2 Reasoning Logic Error Repair

Reasoning errors occur when evidence is sufficient, but the model's reasoning diverges from the correct solution path. The trajectory is truncated at the localized error index $k^\dagger$, retaining the conditionally valid prefix $y_{\text{prefix}} = \{\rho^{(1)}, \ldots, \rho^{(k^\dagger-1)}\}$. All retrieved documents $D(y)$ from the trajectory are aggregated, and the repair model re-performs reasoning from the retained prefix, grounded on $D(y)$, to generate a corrected answer (Figure 11). The model is instructed to base its reasoning strictly on the provided evidence and output the final answer in `<answer>` tags.

F.3 Retriever Error Repair

Retriever errors occur when the model issues valid search queries, but the retriever fails to return relevant documents. The trajectory is truncated at $k^\dagger$, retaining the valid prefix y_{prefix}. The repair procedure has three stages: query rewriting, re-retrieval, and answer generation.

Stage 1: Query Rewriting. The failed search queries (at $k^\dagger$ and any subsequent failed retrievals) are rewritten into multiple alternative formulations (Figure 12). For each original query, the model generates at least 3 alternative versions using different wording, synonyms, or broader/narrower expressions, while preserving the original information needed.

Algorithm 2: Tool-Conditioned Local Repair

Input: Failed trajectory (q, y, a_{pred}); diagnosis result $(c, k^\dagger)$; retrieval service $\mathcal{R}$; max search turns T.
Output: Repaired answer a_{repaired}.

switch c **do**
- **case** *Format Error* **do**
  - Prompt the model with (q, y) to generate EM-valid answer variants;
  - Return the variant list as a_{repaired};
- **case** *Reasoning Error* **do**
  - Extract valid prefix $y_{\text{prefix}} \leftarrow y[1 : k^\dagger - 1]$;
  - Aggregate all retrieved documents $D_{all}(y)$ from the trajectory;
  - Prompt the model with $(q, y_{\text{prefix}}, D_{all}(y))$ to re-reason and generate answer;
  - Return a_{repaired};
- **case** *Retriever Error* **do**
  - Extract valid prefix $y_{\text{prefix}} \leftarrow y[1 : k^\dagger - 1]$;
  - Extract queries whose retrieval results failed at and after $k^\dagger$: $\mathcal{Q}_{\text{failed}}$;
  - Rewrite each $q_i \in \mathcal{Q}_{\text{failed}}$ into alternative formulations $\{q_i^{(1)}, q_i^{(2)}, q_i^{(3)}\}$;
  - Re-retrieve: $D_{\text{new}} \leftarrow \bigcup_j \mathcal{R}(q_i^{(j)}, \text{top-}k)$;
  - Prompt the model with $(q, y_{\text{prefix}}, D_{\text{new}})$ to generate answer;
  - Return a_{repaired};
- **case** *Search Error* **do**
  - Extract valid prefix $y_{\text{prefix}} \leftarrow y[1 : k^\dagger - 1]$;
  - **Turn 0**: Generate search plan from (q, y_{prefix});
  - Initialize new history $h \leftarrow$ plan;
  - **for** $t = 1$ **to** T **do**
    - Prompt the model with $(q, y_{\text{prefix}}, h, t, T)$;
    - Parse output: `<reason>` + action;
    - **if** *action is* `<search>query</search>` **then**
      - $d \leftarrow \mathcal{R}(\text{query}, \text{top-}k)$;
      - Append reason, query, and d to h;
    - **else if** *action is* `<answer>result</answer>` **then**
      - Return result as a_{repaired};
  - Force answer generation on final turn;
  - Return a_{repaired};

Stage 2: Re-Retrieval. Each rewritten query is submitted to the retrieval service with an increased top-k to maximize recall. The retrieved documents from all rewritten queries are aggregated.

Stage 3: Answer Generation. The model generates a final answer using the original question, the retained valid prefix, and the newly retrieved evidence (Figure 13). The model is instructed to use only the provided evidence and output the answer in `<answer>` tags.

F.4 Search Error Repair

Search errors occur when flawed reasoning produces invalid search queries, preventing the retriever from ever receiving a fair query. Since the root cause lies in the reasoning that generated the queries, repair requires regenerating both reasoning and retrieval from the failure point. This is the most complex repair operator, implemented as a multi-turn planning and execution loop.

Stage 1: Plan Generation (Turn 0). The model receives the valid trajectory prefix (up to $k^\dagger$) and generates a structured search plan (Figure 14). The plan decomposes the question into atomic facts, labels each as SUPPORTED (present in history) or MISSING, and specifies concrete search intents for each missing fact. This prevents redundant searches for already-known facts.

Stage 2: Iterative Search and Reasoning (Turns 1–N). After planning, the model iteratively executes the plan (Figure 15). Each turn produces:

Diagnosis Prompt (Teacher Annotation & Student Inference)

You are an expert Fault Localization Judge for an Agentic RAG system. The model's answer failed Exact Match (EM=0). Your goal is to identify the root cause of the failure. **Assumption**: The Ground Truth Answer is CORRECT. Do not question or second-guess it. Your job is to find WHY the model failed, not whether the Ground Truth is valid.

Diagnosis Logic (Execute strictly in this order):
Phase 1: Format-Invalid Answer Error (Priority 1 — Check this FIRST) — Check: Extract the model's final answer. Compare it to the Ground Truth Answer. **IS a Format Error** (semantically same, string different): Same entity with different surface form ("Barry Bonds" vs "Barry Lamar Bonds"); same value with different expression ("27" vs "27 years old"); synonym or equivalent category ("film" vs "movie"); model answered correctly via parametric knowledge despite potentially incomplete evidence. **NOT a Format Error** (different entity or factually wrong): Completely different entities; different people; different facts; model answered the wrong aspect of the question. **Decision**: Output **"Format-Invalid Answer Error"** ONLY if the answer is the same fact in a different string form.
Phase 2: Evidence Sufficiency Check (Key Branching Point) — After ruling out Format Error, determine whether the retrieved evidence is sufficient to answer correctly: **Evidence IS Sufficient**: [Evidence] blocks contain the necessary facts to derive the GT Answer. The model HAD the right information but failed to use it ⇒ Go to Phase 3. **Evidence is NOT Sufficient**: [Evidence] blocks do NOT contain the key facts needed ⇒ Go to Phase 4.
Phase 3: Reasoning Logic Error (Evidence Sufficient) — Condition: Retrieved evidence is sufficient, but the model still failed. **Criteria**: Hallucination, Contradiction, Wrong Entity/Date, Misinterpretation, Ignored Evidence. **Decision**: ⇒ **"Reasoning Logic Error"** + identify the culprit reasoning step.
Phase 4: Retrieval Error Classification (Evidence NOT Sufficient) — Entity Provenance Check: For each search query, verify entity sources: Entity from User Question or previous Evidence ⇒ Valid source. Entity invented/hallucinated by model ⇒ Invalid source.
Phase 4a — Retriever Failure: Valid query (correct entities, specific) but retriever returned irrelevant results ⇒ **"Retriever Failure"** (culprit = the failed evidence block).
Phase 4b — Reasoning-Induced Missing Retrieval: Query is wrong due to hallucinated entities, wrong assumption, or misdirected targets ⇒ **"Reasoning-Induced Missing Retrieval"** (culprit = the first poisoned search action).
Edge case: Well-formed query targeting wrong entity (hallucinated) ⇒ Phase 4b, not 4a.

Input: `User Question:` {*question*} `Ground Truth Answer:` {*gt_answer*} `Model Trajectory` (Indexed Steps): {*formatted_trajectory*}
Output (strictly JSON): `{"assigned_error_type": "...", "culprit_step_label": "..."}`

Figure 9: Diagnosis prompt used for both teacher annotation and student inference. The prompt covers all four error types with evidence sufficiency as an implicit branching point, eliminating the need for an external coverage signal.

1. A `<reason>` block analyzing the current state and deciding the next action.
2. Exactly one action: either `<search>query</search>` to retrieve missing information, or `<answer>result</answer>` when sufficient information is gathered.

Search results are appended to the context for subsequent turns. On the final turn, the model is forced to output an answer regardless of information completeness.

F.5 Repair Algorithm Summary

Algorithm 2 summarizes the complete repair procedure. The key properties are: (1) prefix reuse—all operators preserve the valid prefix before $k^{\dagger}$; (2) minimal intervention—each operator modifies only the components required by its error type; (3) bounded computation—the search error repair has a fixed turn budget to prevent unbounded generation.

F.6 Repair Prompt Templates

Figures 10–15 present the complete prompt templates for all repair operators.

Format-Invalid Answer Repair Prompt

You are a Senior Text Rewriting and Format Correction Specialist. Your task is to analyze the entire dialogue history, particularly the retrieval results and reasoning, to generate a comprehensive JSON list of all possible Exact Match (EM) answers for the original question.

I. Determine the Golden Answer Format and Source
The model's original answer is factually correct but suffers from a Format Error. Determine the correct EM answer based on:
Priority 1: Document Extraction — Strictly review content inside all `<information>` tags. If the final answer can be found verbatim in retrieved documents, use that document's original expression as the primary answer.
Priority 2: Format Enumeration — If not explicitly found, derive the core fact and generate all valid EM variations.

II. Generate All EM Variants
Generate at least 10 valid variations including: alternate symbol/connector forms, number formats, common shortenings, article variations, and casing permutations.
Requirements: Accuracy (factually correct), Minimality (Minimal EM standard), Exhaustiveness (all common variations).

III. Output — Strict JSON list of strings. No reasoning or markdown outside the JSON.
Example: `["Original Document Phrase", "Variant 2", "Variant 3"]`

Input: Dialogue History (reasoning + tool calls), Original Question, Model's Incorrect Final Answer.

Figure 10: Format repair prompt. The model generates multiple exact-match candidate strings from the existing trajectory without modifying reasoning or retrieval.

Reasoning Logic Error Repair Prompt

Question: {*question*}
Valid Prefix (reasoning history before the error): {*valid_prefix*}
Retrieved Evidence: {*aggregated_documents*}
Instruction:
- Carefully analyze the Question using the valid prefix and the retrieved evidence.
- You may perform step-by-step reasoning to reach the answer.
- Base your reasoning strictly on the provided evidence.

Final Output Requirement:
- Provide the final answer wrapped strictly in `<answer>...</answer>`.
- If the answer can be found verbatim in the documents, use that expression.
- The answer must be a single entity, name, or short phrase. Do NOT write complete sentences.

Figure 11: Reasoning repair prompt. Given the valid prefix and aggregated retrieved documents, the model re-reasons to produce a corrected answer.

Query Rewrite Prompt (Retriever Error Repair — Stage 1)

User Question: {*question*}
The agent previously issued the following failed or ineffective search queries:
{*failed_queries*}
Your Task: For EACH original query:
1. Analyze its underlying information need.
2. Rewrite it into AT LEAST 3 alternative search queries.
3. The rewritten queries should: preserve the original intent; use different wording, synonyms, or broader/narrower expressions; be suitable for dense or hybrid retrieval systems.

Important Rules:
- Do NOT merge different original queries. Handle each independently.
- Each original query must have its own rewritten versions.

Output Format: For each original query, output rewritten versions in `<search>` tags.

```
Original Query: xxx
<search>rewrite 1</search>
<search>rewrite 2</search>
<search>rewrite 3</search>
```

Figure 12: Query rewrite prompt. Each failed query is independently rewritten into multiple alternative formulations to improve retrieval recall.

Answer Generation Prompt (Retriever Error Repair — Stage 3)

Question: {*question*}
— Retrieved Evidence —
The following documents are retrieved from the knowledge base. They may contain overlapping or complementary information.
{*docs_block*}
— Instruction —
Using ONLY the information in the retrieved evidence above, answer the question as accurately as possible. Wrap your final answer strictly inside `<answer>` and `</answer>`. If the answer can be found verbatim in the documents, use that expression. The answer must be a single entity, name, or short phrase. Do NOT write complete sentences.

Figure 13: Answer generation prompt after re-retrieval. The model synthesizes the newly retrieved evidence to produce a corrected answer.

Search Plan Prompt (Search Error Repair — Turn 0)

— TASK —
You are repairing a failed reasoning trajectory. Your task is to RE-PLAN a correct solution path.
[History Start]
{*old_history (valid prefix)*}
[History End]
Question: {*question*}
— DEFINITIONS —
- The history may already contain VALID facts from previous searches.
- Any information present in the history CAN be reused directly.
- You MUST NOT plan new searches for facts already supported by the history.

— INSTRUCTIONS —
Based strictly on the history and question, construct a SEARCH-DRIVEN execution plan. The goal is to obtain ONLY the missing knowledge needed to answer the question.
Your plan MUST be enclosed in `<plan>` and `</plan>` tags. In the plan:
1. List all atomic facts REQUIRED to answer the question.
2. For EACH fact, label as SUPPORTED or MISSING.
3. For EACH MISSING fact, specify ONE concrete `<search>` intent.
4. Define the search order.

Constraints: Do NOT include searches for SUPPORTED facts. Do NOT perform reasoning, searching, or answering. ONLY output the `<plan>`.

Figure 14: Search plan prompt (Turn 0). The model generates a structured plan identifying which facts are missing and what searches are needed.

Search Execution Prompt (Search Error Repair — Turns 1–N)

System Role
You are a Correction Agent. Your goal is to fix a reasoning path and guide it to the correct answer.
Output Format Rules
1. `<reason>`: Analyze the next step. Strictly plain text.
2. **Action**: Immediately follow reasoning with EXACTLY ONE block:
   - `<search>query</search>`: Retrieve missing external info.
   - `<answer>result</answer>`: Provide the final answer (single entity/short phrase).

Input Data
Question: {*question*}
Context History: {*old_history*}
Current Corrective Reasoning: {*new_history*}
Instructions
1. Reason: Output reasoning for the correction in a `<reason>` tag.
2. Act: Take the first correct action (`<search>` or `<answer>`).

{*turn_constraint*}

turn_constraint = "This is the LAST TURN. You MUST output `<answer>` immediately." on the final turn; otherwise "You may use `<search>` if needed, or `<answer>` if sufficient information is gathered."

Figure 15: Search execution prompt (Turns 1–N). The model iteratively searches for missing facts and reasons toward the final answer, with a forced-answer constraint on the last turn.

G LIMITATIONS

While DR-RAG demonstrates strong trajectory-level diagnosis and localized repair for text-based Agentic RAG tasks, several natural extensions remain for future work. First, the framework can be extended to multi-modal settings involving images, tables, or structured documents. Second, DR-RAG can be combined with more diverse retrieval strategies and external knowledge sources. Third, the repaired trajectories produced by DR-RAG may serve as improved supervision signals for trajectory distillation, enabling future systems to learn from corrected reasoning and retrieval behaviors. Finally, applying DR-RAG to longer or more hierarchical reasoning trajectories may further broaden its applicability. These directions do not affect the effectiveness of the proposed framework on the standard text-based tasks studied in this paper, but they provide useful opportunities for future exploration.