

VRR: Verification-Guided Retrieval Revision for Multi-Step QA

Zhijie Li^{1,2,3} Lei Wang^{1,2,3*} Tianyuan Li^{1,2,3} Bangju Han^{1,2,3}

Ahtamjan Ahmat^{1,2,3} Rui Dong^{1,2,3} Zhen Wang^{1,2,3}

¹Xinjiang Technical Institute of Physics & Chemistry,
Chinese Academy of Sciences, Urumqi, China

²University of Chinese Academy of Sciences, Beijing, China

³Xinjiang Laboratory of Minority Speech and Language Information Processing,
Urumqi, China

{lizhijie24, litianyuan23, hanbangju23}@mailsucas.ac.cn

{wanglei, ahtmj, dongrui, wang_zhenj}@ms.xjb.ac.cn

Abstract

Retrieval-augmented language models improve knowledge-intensive question answering, but they remain brittle on multi-step questions, where an early retrieval error can steer subsequent reasoning toward the wrong entity or relation. Verification can identify unreliable answers or weak evidence states, and multi-step RAG further needs a mechanism that turns such diagnoses into concrete retrieval repairs. We propose Verification-Guided Retrieval Revision (VRR), a framework that makes verification actionable at the retrieval-control level. VRR couples a Generator that proposes search and answers, a Judge that verifies answer correctness and evidential support while emitting localized and typed failure diagnoses, and a Retrieval Context Manager (RCM) that converts these diagnoses into retriever-facing updates such as clue selection, query rewriting, evidence packing, and state-aware ranking. To make this loop learnable, we train VRR in three stages: iterative cold-start training to stabilize search, judge-shaped policy optimization to improve long-horizon reasoning, and diagnosis-conditioned RCM training for retrieval-state adaptation. Across seven open-domain QA benchmarks with a fixed retriever and corpus, training on HotpotQA and evaluating cross-benchmark transfer, VRR surpasses the strongest prior baseline by 4.2 EM on average under budget-aligned evaluation, with larger gains on multi-hop benchmarks (+5.4).

1 Introduction

Large language models (LLMs) answer many knowledge-intensive questions by combining parametric knowledge with contextual reasoning, and retrieval-augmented generation (RAG) grounds this reasoning in external evidence (Mallen et al., 2023; Lewis et al., 2020; Guu et al., 2020; Izacard and

Grave, 2021). Many open-domain questions require multi-step retrieval: the model must identify bridge entities, refine queries, disambiguate candidates, and decide whether current evidence is sufficient. Success therefore depends not only on using retrieved documents, but also on *revising retrieval* after the search trajectory drifts.

This issue is especially severe in multi-step QA. Once early retrieval commits to the wrong entity, relation, or constraint, later reasoning may remain locally coherent while being grounded in irrelevant evidence. Prior work has improved multi-step QA with stronger retrievers (Karpukhin et al., 2020; Izacard and Grave, 2021), interleaved retrieval and reasoning (Xiong et al., 2021; Trivedi et al., 2023; Yao et al., 2023; Asai et al., 2024), search-oriented controllers (Jin et al., 2025; Chang et al., 2025; Huang et al., 2025a), and verification modules (Zhang et al., 2023; Dong et al., 2025; Chu et al., 2025; Chen et al., 2025). VRR advances this line by learning a structured diagnosis-to-retrieval interface: verification localizes the failure, and the RCM uses it to update retrieval.

We focus on verification-to-retrieval action in the retrieval-drift regime. Multi-hop failures often involve wrong bridge entities, unsupported relations, missing bridge evidence, or locally plausible but globally incorrect evidence trajectories. A useful verifier should therefore produce retrieval-facing signals that identify what failed and how to repair the evidence state.

We propose **Verification-Guided Retrieval Revision (VRR)**, a three-module framework with a *Generator* that proposes search actions and answers, a *Judge* that predicts accept/revise and emits typed failure diagnoses, and a *Retrieval Context Manager (RCM)* that maps diagnoses into retriever-facing updates. The Judge labels failures such as entity mismatch, missing bridge evidence, unsupported relation, and answer-type mismatch;

*Corresponding author: wanglei@ms.xjb.ac.cn.

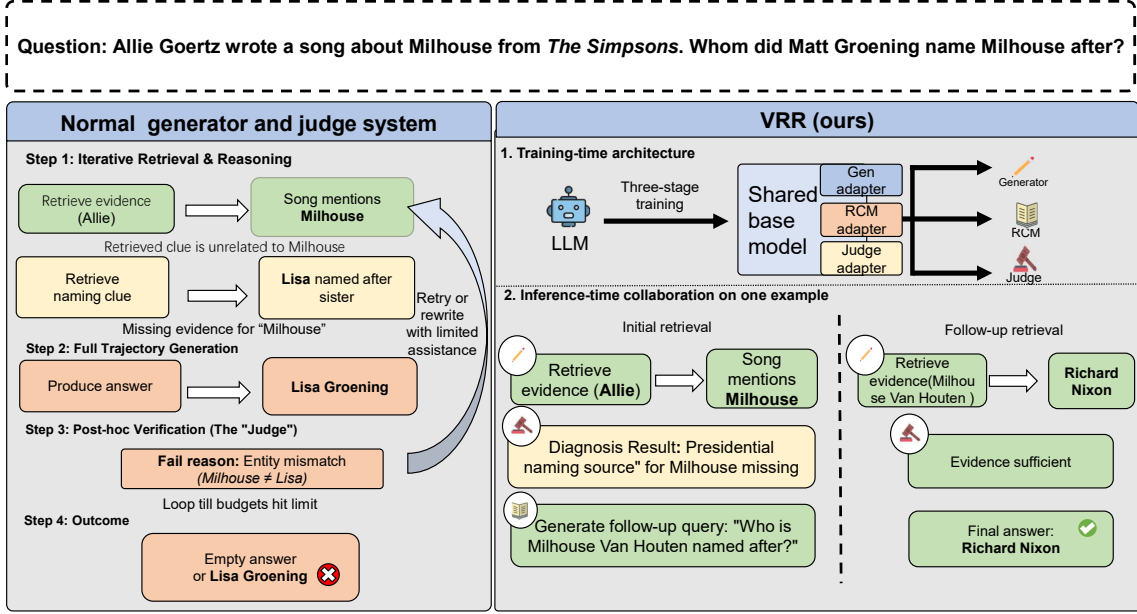


Figure 1: Overview example contrasting a standard generator–judge setup with VRR on a multi-hop question. Left: ReAct follows an entity-mismatched trajectory and ends with an incorrect answer. Right: VRR uses Judge diagnoses to trigger retriever-facing revisions through the RCM and recovers the correct answer. Additional step-by-step examples are provided in Appendix D.1.

the RCM turns them into clue selection, diagnosis-conditioned query rewriting, state-aware ranking, and evidence packing or replacement. VRR makes verification actionable at the retrieval-control level by learning how localized verification failures revise the next retrieval state.

Training this loop is challenging because the Generator, Judge, and RCM co-adapt while search distributions shift and errors compound over long horizons. We therefore train VRR in three stages that stabilize the closed loop: iterative cold-start training bootstraps stable search and reasoning trajectories, Judge-guided policy optimization shapes long-horizon reasoning with verification signals, and retrieval optimization trains the RCM as a diagnosis-conditioned retrieval controller.

We evaluate VRR on seven open-domain QA benchmarks, training only on HotpotQA and testing cross-benchmark transfer. With a fixed retriever and corpus, VRR improves average EM by 4.2 points over the strongest prior baseline, with larger gains on multi-hop benchmarks. Retrieval-call cap analysis shows that VRR uses additional retrieval opportunities more effectively than R1-Searcher++, and ablations show that the final gain comes from diagnosis-conditioned retrieval revision rather than a stronger retriever, extra evidence source, or generic reranking.

Our contributions are threefold:

- We identify the *verification-to-retrieval interface* as a significant control problem for repairing retrieval drift in multi-step QA.
- We propose **VRR**, a Generator–Judge–RCM framework that converts localized verification diagnoses into retrieval-state updates.
- We develop a three-stage training pipeline and show consistent gains across seven QA benchmarks, especially on multi-hop transfer.

2 Related Work

VRR is most related to iterative retrieval, verification-based RAG, and search-control agents. Existing methods improve evidence acquisition, answer checking, or tool-use planning, while VRR focuses on the interface between verification and retrieval: a localized failure diagnosis is converted into the next retrieval-state update.

Iterative retrieval and multi-step reasoning. Open-domain QA has evolved from retrieve-then-read pipelines to interleaved retrieval–reasoning methods for multi-step questions (Karpukhin et al., 2020; Xiong et al., 2021; Trivedi et al., 2023; Yao et al., 2023; Asai et al., 2024). Multi-hop retrieval

also emphasizes bridge-entity transitions and evidence chaining (Das et al., 2019). These methods improve evidence coverage by repeatedly querying, reasoning, or self-refining. VRR instead targets retrieval drift: when a trajectory follows a wrong entity, relation, or constraint, the Judge identifies the failure type and the RCM revises the retrieval state accordingly.

Verification and critique for grounded QA.

Verification and critique modules improve RAG reliability by checking answer support, ranking candidate evidence, or detecting unsupported generations (Asai et al., 2024; Zhang et al., 2025; Chen et al., 2025; Dong et al., 2025; Chu et al., 2025). These approaches show that answer and evidence verification are useful for grounded QA. VRR makes verification actionable for retrieval control. The Judge emits typed diagnoses such as entity mismatch, missing bridge evidence, unsupported relation, and answer-type mismatch; the RCM maps them to query rewriting, evidence expansion, state-aware selection, and context replacement. Appendix C.4 reports comparisons with representative critic-based RAG baselines.

Search control and role decomposition. Recent agentic and modular search systems train models to plan retrieval, use tools, or decompose search into specialized roles, including Search-R1, R1-Searcher, R1-Searcher++, MAIN-RAG, RAG-Gym, ManuSearch, and MetaGPT (Jin et al., 2025; Song et al., 2025a,b; Chang et al., 2025; Xiong et al., 2026; Huang et al., 2025a; Hong et al., 2024). VRR is complementary to this line: it introduces a diagnosis-conditioned control interface in which verification directly changes what is retrieved next. Under a fixed retriever, corpus, and budget, this design isolates the benefit of connecting failure diagnosis to retrieval revision.

3 Methodology

We formulate retrieval-augmented question answering as multi-round search over a fixed corpus. At each round, the Generator proposes a query and a candidate answer, the Judge decides whether that answer is sufficiently grounded, and the Retrieval Context Manager (RCM) converts failed verdicts into retriever-facing updates. In the deployed system, the Judge is hybrid: a learned verifier predicts verdicts and typed diagnoses, while deterministic gates block unsupported accept decisions. The

diagnosis schema covers localized failures such as entity mismatch, missing bridge evidence, unsupported relation, and answer-type mismatch. Generator, Judge, and RCM share one backbone LLM with role-specific LoRA adapters so that they can exchange compact textual states while still specializing in search proposal, verification, and retrieval adaptation. Figure 2 summarizes the training pipeline.

3.1 Problem Setup

Let x be a question, y^* its gold answer, and $\mathcal{C}$ a large external corpus. A retriever

$$\text{Ret}(q; \mathcal{C}, K) \rightarrow \{d_1, \dots, d_K\} \quad (1)$$

returns the top- K documents for query q . The controller interacts with the retriever for at most R rounds. At round t , the search state is

$$s_t = (x, h_t, u_{t-1}), \quad (2)$$

where h_t is the history up to round $t - 1$ and u_{t-1} is the latest Judge-derived revision instruction. We write

$$h_t = \{(q_i, E_i, \hat{y}_i, z_i, \rho_i, f_i)\}_{i=1}^{t-1}, \quad (3)$$

where q_i is the executed query, E_i the packed evidence, $\hat{y}_i$ the candidate answer, $z_i \in \{0, 1\}$ the Judge decision, $\rho_i \in \{0, 1, \dots, 10\}$ the severity score, and $f_i = (\mathcal{I}_i, \mathcal{Q}_i)$ the structured feedback with issue types $\mathcal{I}_i$ and suggested repair queries $\mathcal{Q}_i$. Subscripts index search rounds, while superscripts (0) and (1) denote retrieval before and after rewriting within the same round. We use $\mathcal{D}_t^{(0)}$ and $\mathcal{D}_t^{(1)}$ for the candidate sets before and after revision, m_t for the clue subset selected by the RCM, and u_t for the Judge-derived retrieval instruction. K_0 and K_1 are the two retrieval depths, $\alpha_t(d)$ is the RCM ranking score for passage d , TopM keeps the highest-scoring M passages, and Pack_B compresses them into an evidence budget of B tokens. Model parameters θ , ϕ , and η correspond to the Generator, Judge, and RCM. The objective is to maximize answer quality while keeping search grounded and budget-aware.

3.2 VRR: Verification-Guided Retrieval Revision

VRR factorizes control into a propose–verify–repair loop with three tightly coupled roles: a Generator for search and answer proposal, a Judge for

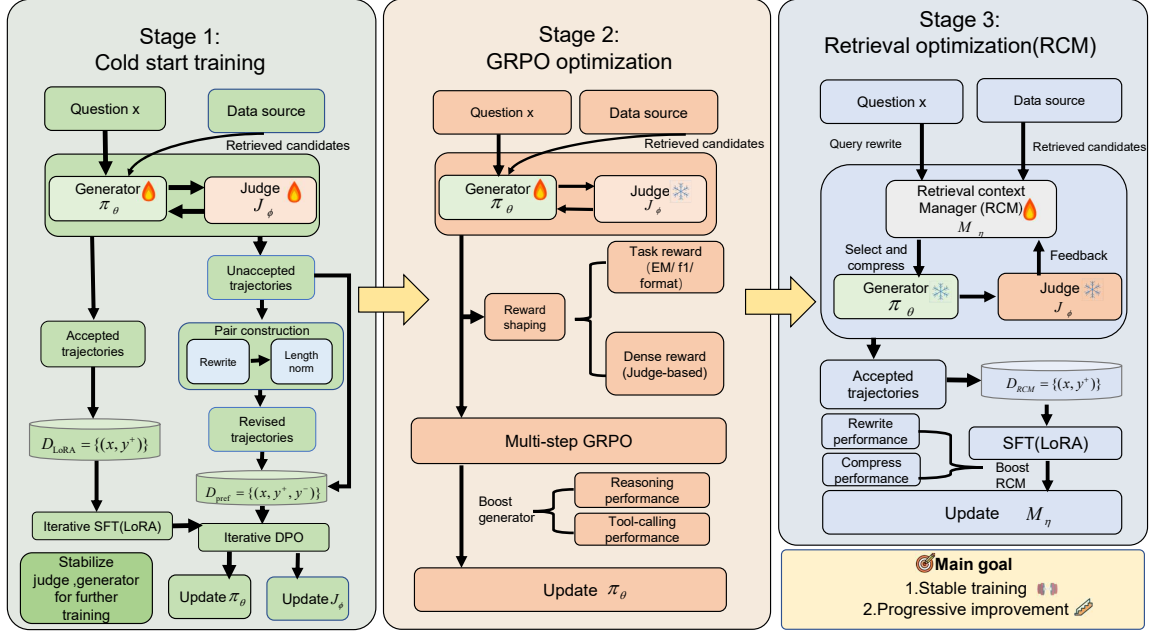


Figure 2: Main training process for our VRR instantiation: Stage 1 cold-start training, Stage 2 GRPO optimization, and Stage 3 retrieval optimization for the feedback-to-instruction framework implemented by the RCM. Symbols: π_θ is the Generator policy, J_ϕ is the Judge, M_η is the retrieval manager, and D_{LoRA} , D_{pref} , D_{RCM} are stage-specific trajectory/pair datasets. The flame icon marks modules updated in the current stage, while the snowflake icon marks frozen modules.

evidence-grounded verification, and an RCM for retriever-facing adaptation. The crucial interface is the Judge-to-RCM message: the Judge emits typed diagnoses, and the RCM interprets them as retrieval-state updates rather than free-form critique. This design makes failure analysis actionable at the level of the retriever, not merely descriptive at the level of the Generator.

3.2.1 Generator

At round t , the Generator first proposes a base query

$$q_t^{(0)} \sim \pi_\theta^q(\cdot \mid s_t), \quad (4)$$

and, after receiving evidence E_t , produces a candidate answer

$$\hat{y}_t \sim \pi_\theta^y(\cdot \mid x, h_t, E_t). \quad (5)$$

The Generator uses reasoning to produce an answer and to decide what information should be retrieved next.

3.2.2 Judge

Given $(x, h_t, E_t, \hat{y}_t)$, the Judge outputs

$$(z_t, \rho_t, f_t) = J_\phi(x, h_t, E_t, \hat{y}_t), \quad (6)$$

where $z_t \in \{0, 1\}$ is the accept/revise decision, $\rho_t \in \{0, 1, \dots, 10\}$ is the severity score, and

$f_t = (\mathcal{I}_t, Q_t)$ is structured feedback. We use $\bar{\rho}_t = \rho_t/10$ in reward shaping. An accept decision is allowed only when answer type, entity grounding, and deterministic support all hold:

$$z_t = \mathbf{1} \left[c_{\text{type}}(x, \hat{y}_t) \wedge c_{\text{entity}}(\hat{y}_t, E_t) \wedge c_{\text{support}}(x, \hat{y}_t, E_t) \right]. \quad (7)$$

where c_{type} , c_{entity} , and c_{support} are deterministic checks for answer-type compatibility, entity grounding, and explicit support in E_t . The Judge therefore provides both a stopping signal and a retrieval-revision message. A learned verifier predicts (z_t, ρ_t, f_t) , and the strict gate can rewrite a raw accept to revise. Unlike inter-passage verification, which mainly validates fixed candidate answers across passages (Chen et al., 2025), our Judge emits localized failure diagnoses for search revision. The resulting feedback is converted into the next retrieval instruction

$$u_t = \Gamma(f_t, \rho_t, h_t), \quad (8)$$

which conditions the next round of search. Fuller Judge diagnostics are deferred to Appendix C.9.

3.2.3 Retrieval Context Manager

The RCM is a diagnosis-conditioned retrieval controller. Given a base query $q_t^{(0)}$, state s_t , and revision instruction u_{t-1} , it performs initial retrieval,

clue selection, diagnosis-conditioned query rewriting, state-aware ranking, and evidence packing or context replacement:

$$\begin{aligned}
\mathcal{D}_t^{(0)} &= \text{Ret}(q_t^{(0)}; \mathcal{C}, K_0), \\
m_t &= M_\eta^{\text{sel}}(x, h_t, \mathcal{D}_t^{(0)}, u_{t-1}), \\
q_t^{(1)} &= M_\eta^{\text{rewrite}}(x, h_t, q_t^{(0)}, m_t, u_{t-1}), \\
\mathcal{D}_t^{(1)} &= \text{Ret}(q_t^{(1)}; \mathcal{C}, K_1), \\
\alpha_t(d) &= M_\eta^{\text{rank}}(x, h_t, d, u_{t-1}), \\
E_t &= \text{Pack}_B(\text{TopM}(\mathcal{D}_t^{(0)} \cup \mathcal{D}_t^{(1)}; \alpha_t)).
\end{aligned} \tag{9}$$

The RCM first retrieves with the base query, selects salient clues, rewrites the query under the previous Judge instruction, retrieves again, and then ranks the union of both candidate sets before packing the top- M passages into E_t . The second retrieval is therefore conditioned on Judge feedback rather than on a fixed answer candidate. In practice, M_η denotes the learned retrieval adapter trained in Stage 3; heuristic controls are retained only as ablations and fallbacks under the same RCM interface (Appendix E.4).

3.3 Closed-Loop Inference

At inference time, the controller alternates Generator $\rightarrow$ RCM $\rightarrow$ Judge until an accept decision is issued or the round budget is exhausted. Retrieval revision is triggered only for localized diagnoses such as entity mismatch or missing bridge evidence, which reduces unnecessary query churn and keeps control tied to concrete failure modes. Appendix E.4 provides the corresponding pseudocode.

3.4 Three-Stage Training

Jointly training the Generator, Judge, and RCM from scratch is unstable, so we optimize them sequentially. The three-stage training pipeline stabilizes the closed loop among Generator, Judge, and RCM. A key design choice is to rebuild trajectory data from the model’s own multi-step rollouts rather than rely on a fixed offline trace set. This keeps supervision close to the policy’s visited search distribution. Below, τ denotes a full trajectory, $\mathcal{D}_{\text{LoRA}}$, $\mathcal{D}_{\text{judge}}$, and $\mathcal{D}_{\text{pref}}$ denote the current supervision sets, and β in Eq. (12) is the preference temperature.

3.4.1 Stage 1: Iterative Cold-Start Training

Stage 1 bootstraps stable search and reasoning behavior before reinforcement learning. It provides

usable initial trajectories for closed-loop training and reduces the difficulty of starting from random or unstable interactions. Each bootstrap iteration rolls out the current Generator, Judge, and RCM on the training questions and rebuilds three datasets: accepted trajectories for Generator SFT, Judge-labeled states for verification learning, and preference pairs formed from sibling rollouts on the same question. Because all three sets are reconstructed from the current policy, the supervision remains close to the search distribution actually visited during training.

The Generator is first trained on accepted trajectories with supervised fine-tuning:

$$\mathcal{L}_{\text{gen-sft}} = - \sum_{(x, \tau^+) \in \mathcal{D}_{\text{LoRA}}} \log p_\theta(\tau^+ | x), \tag{10}$$

where τ^+ denotes a successful multi-step search trajectory.

The Judge is trained to predict acceptance, severity, and feedback:

$$\begin{aligned}
\mathcal{L}_{\text{judge}} = - \sum_{(x, \hat{y}, E, z, \rho, f) \in \mathcal{D}_{\text{judge}}} & \\
\log p_\phi(z, \rho, f | x, \hat{y}, E), & \tag{11}
\end{aligned}$$

where z is the binary acceptance label, $\rho \in \{0, 1, \dots, 10\}$ is the integer severity label, and $f = (\mathcal{I}, \mathcal{Q})$ is the structured feedback label.

We then refine the Generator with preference optimization on model-generated trajectory pairs $\mathcal{D}_{\text{pref}} = \{(x, \tau^+, \tau^-)\}$:

$$\begin{aligned}
\mathcal{L}_{\text{pref}} = - \sum_{(x, \tau^+, \tau^-) \in \mathcal{D}_{\text{pref}}} & \log \sigma \Big(\\
\beta \Big[\log \pi_\theta(\tau^+ | x) - \log \pi_\theta(\tau^- | x) \Big] \Big). & \tag{12}
\end{aligned}$$

Here, τ^+ and τ^- are selected from alternative rollouts of the *current* model for the same question, rather than from a fixed external preference set. Stage 1 optimizes a weighted sum of $\mathcal{L}_{\text{gen-sft}}$, $\mathcal{L}_{\text{judge}}$, and $\mathcal{L}_{\text{pref}}$, and repeats rollout collection and updates until the controller reaches a stable pre-RL regime.

3.4.2 Stage 2: Judge-Shaped GRPO

After cold start, Stage 2 optimizes the Generator with Judge-guided GRPO. The objective uses Judge feedback to shape long-horizon reasoning, so the Generator learns to use verification signals across multi-step retrieval and answering rather

than only optimizing final answer generation. For each question, we sample a group of trajectories from the current policy and assign each trajectory a scalar reward

$$r(\tau_i) = \lambda_{\text{em}} r_{\text{em}}(\tau_i) + \lambda_{\text{fl}} r_{\text{fl}}(\tau_i) + \lambda_{\text{fmt}} r_{\text{fmt}}(\tau_i) + \lambda_{\text{acc}} r_{\text{acc}}(\tau_i) - \lambda_{\text{sev}} r_{\text{sev}}(\tau_i). \quad (13)$$

Here, r_{em} and r_{fl} reward final answer quality, r_{fmt} output validity, r_{acc} terminal Judge acceptance, and $r_{\text{sev}} = \rho/10$ penalizes more severe failures. Acceptance provides a coarse grounding signal, while severity ranks failed trajectories within the same group.

Following prior GRPO practice, we further suppress uninformative updates when all sampled trajectories in a group of size G are nearly indistinguishable:

$$g(x) = \mathbf{1}[\text{Var}(\{r(\tau_i)\}_{i=1}^G) > \epsilon_r \vee \text{Var}(\{z_i\}_{i=1}^G) > \epsilon_z], \quad (14)$$

where z_i is the terminal Judge acceptance label of trajectory τ_i . We then optimize $\tilde{\mathcal{L}}_{\text{GRPO}} = g(x) \mathcal{L}_{\text{GRPO}}$. Judge shaping and the Stage 1 initialization together stabilize long-horizon search optimization; Appendix C.12 reports the corresponding saturation and update diagnostics.

3.4.3 Stage 3: Retrieval Optimization for the RCM

Finally, we freeze the Generator and Judge and optimize only the RCM. Stage 3 trains the RCM to map the Judge’s localized diagnoses into retrieval-state updates. It teaches clue selection, diagnosis-conditioned query rewriting, state-aware evidence ranking, and evidence packing or context replacement under the fixed answer policy learned in Stages 1–2.

First, a rewrite objective supervises refined query generation:

$$\mathcal{L}_{\text{rewrite}} = - \sum \log p_{\eta}(q^{(1)\star} | x, h, q^{(0)}, u, \mathcal{D}^{(0)}), \quad (15)$$

where $q^{(1)\star}$ is the refined query associated with a successful trajectory.

Second, a ranking objective supervises evidence selection:

$$\mathcal{L}_{\text{rank}} = - \sum \log \sigma(\alpha_{\eta}(d^+ | x, h, u) - \alpha_{\eta}(d^- | x, h, u)). \quad (16)$$

Here, d^+ is a positive passage appearing in accepted evidence and d^- is a negative passage outside it. Stage 3 optimizes a weighted sum of $\mathcal{L}_{\text{rewrite}}$ and $\mathcal{L}_{\text{rank}}$. Its training examples come from trajectories where Judge feedback exposes a retrieval error and a later revision resolves it, so positives and negatives are defined under the same retrieval state.

4 Experiments

We evaluate VRR on seven benchmarks and use exact match (EM) as the primary metric. The main paper answers four questions: (1) Does VRR improve over strong prompting and learned-search baselines? (2) Do the gains survive strict cost matching? (3) Which training stages and controller modules matter? (4) Is the Judge reliable enough to control both stopping and retrieval revision? Broader transfer, significance, and mechanism analyses are deferred to the appendix so that the main paper stays centered on the core empirical claim.

4.1 Benchmarks and setup

We evaluate on Natural Questions, TriviaQA, PopQA, HotpotQA FullWiki, 2WikiMultiHopQA, MuSiQue, and Bamboogle (Kwiatkowski et al., 2019; Joshi et al., 2017; Mallen et al., 2023; Yang et al., 2018; Ho et al., 2020; Trivedi et al., 2022; Press et al., 2023). VRR is trained only on a HotpotQA FullWiki subset; all remaining benchmarks are cross-benchmark transfer tests. This asymmetric setup aims to emphasize reusable search control rather than benchmark-specific tuning or memorization. Unless noted, we use Qwen2.5-7B (Qwen Team, 2024) with LoRA adapters (Hu et al., 2022), a fixed dense retriever, and the same Wikipedia snapshot across all methods.

We compare against representative prompting baselines (Direct inference, CoT, RAG, IRCot, Search-o1) and learned search baselines (R1-style RL, Search-R1, R1-Searcher++, and RAG-Gym) (Wei et al., 2022; Lewis et al., 2020; Trivedi et al., 2023; Li et al., 2025; Guo et al., 2025; Jin et al., 2025; Song et al., 2025b; Xiong et al., 2026). We report a *budget-aligned* main comparison using the same retrieval environment and round limits for all methods, followed by a retrieval-call cap sensitivity analysis for VRR and R1-Searcher++ under caps of 2, 4, 6, 8, and 10. Unless noted otherwise, methods share the same wrapper, retriever index, decoding setup, and round limits, so score differences primarily reflect controller quality

Method	Single-hop			Multi-hop				Avg.
	NQ	TriviaQA	PopQA	HotpotQA	2Wiki	MuSiQue	Bamboogle	
<i>Qwen2.5-7B-Instruct</i>								
Direct Inference	13.4±0.2	40.8±0.1	14.0±0.1	18.3±0.2	25.0±0.3	3.1±0.1	12.0±0.2	18.1±0.2
CoT (Wei et al., 2022)	4.8±0.1	18.5±0.2	5.4±0.1	9.2±0.2	11.1±0.1	2.2±0.1	23.2±0.3	10.6±0.2
IRCoT (Trivedi et al., 2023)	22.4±0.2	47.8±0.3	30.1±0.2	13.3±0.2	14.9±0.2	7.2±0.1	22.4±0.2	22.6±0.2
Search-o1 (Li et al., 2025)	15.1±0.2	44.3±0.2	13.1±0.1	18.7±0.2	17.6±0.2	5.8±0.3	29.6±0.3	20.6±0.2
RAG (Lewis et al., 2020)	34.9±0.1	58.5±0.2	39.2±0.2	29.9±0.3	23.5±0.2	5.8±0.1	20.8±0.2	30.4±0.2
R1-style RL (Guo et al., 2025)	27.0±0.2	53.7±0.3	19.9±0.2	23.7±0.2	29.2±0.3	7.2±0.1	29.3±0.2	27.1±0.2
Rejection Sampling	36.0±0.2	59.2±0.2	38.0±0.1	33.1±0.2	29.6±0.2	12.3±0.2	35.5±0.3	34.8±0.2
Search-R1 (Jin et al., 2025)	39.3±0.2	61.0±0.1	39.7±0.2	37.0±0.2	41.4±0.3	14.6±0.2	36.8±0.2	38.5±0.1
R1-Searcher++ (Song et al., 2025b)	40.4±0.2	62.3±0.2	40.4±0.2	43.5±0.2	42.3±0.1	17.5±0.1	41.4±0.2	41.0±0.2
VRR (ours)	43.7±0.1	62.5±0.2	44.4±0.1	48.5±0.1	48.0±0.1	25.0±0.1	44.9±0.2	45.2±0.1

Table 1: Main comparison on Qwen2.5-7B under the budget-aligned evaluation protocol. All values are EM.

rather than a stronger backend. Strict matched-cost evaluation and implementation details are reported in Appendix C.3, Appendix B.1, Appendix C.2, Appendix B.2, and Appendix B.7. Additional base-model, critic-based RAG, and larger-generator results are deferred to Appendix C.4–C.17.

4.2 Main results

Table 1 shows that VRR outperforms both prompting and learned-search baselines under the budget-aligned protocol. Relative to R1-Searcher++, VRR gains +4.2 EM on average (45.2 vs. 41.0), with a much larger improvement on multi-hop datasets (+5.4) than on single-hop datasets (+2.5). This pattern supports the narrower retrieval-drift claim targeted by VRR: when search must recover from an early bridge-entity, disambiguation, or support-selection error, verifier feedback can redirect retrieval instead of repeating the same flawed trajectory.

The dataset-level pattern matches this interpretation. VRR improves six of the seven benchmarks and yields the largest gains on MuSiQue (+7.5), 2Wiki (+5.7), HotpotQA (+5.0), and PopQA (+4.0). The only near-tie is TriviaQA (+0.2), where many questions are effectively answerable once the relevant document is found and therefore benefit less from multi-round retrieval repair. Paired tests in Appendix C.7 show that all datasets except TriviaQA are statistically significant, and the overall and multi-hop averages are significant as well. These results show that VRR improves accuracy by using the available budget more effectively.

4.3 Ablations and controlled analyses

4.3.1 Retrieval-call cap sensitivity

The budget-aligned comparison controls the retrieval environment and round limits; we further

Retrieval-call cap	R1-Searcher++ Avg. EM	VRR Avg. EM	Improvement
2	38.4	44.0	+5.6
4	40.1	44.2	+4.1
6	40.2	44.3	+4.1
8	40.3	44.5	+4.2
10	40.3	44.6	+4.3

Table 2: Retrieval-call cap sensitivity. All values are average EM under the same retriever and corpus. The cap is an upper bound over retriever calls, including revised retrieval triggered by the RCM.

test whether VRR uses a fixed retrieval budget more effectively. Table 2 varies the retrieval-call cap from 2 to 10 for VRR and R1-Searcher++ and reports average EM over the same evaluation suite. The cap is an upper bound on retriever invocations, including both initial retrieval and RCM-triggered revised retrieval, rather than a claim that every question requires that many reasoning hops.

Under a tight cap of two calls, the two methods are close because there is limited room for retrieval revision. As the cap increases, VRR uses the additional retrieval opportunities more effectively: R1-Searcher++ saturates around 40.1–40.3 from four to ten calls, while VRR improves from 44.0 to 44.6 over the same range. The default cap of eight is a near-saturation operating point, since increasing the cap from eight to ten only improves VRR from 44.5 to 44.6 EM. The pattern supports the main mechanism claim: diagnosis-conditioned retrieval revision converts extra calls into higher-yield evidence states rather than merely extending the trajectory. A stricter matched-cost rerun that also controls candidate passages, evidence tokens, and total LLM calls is reported in Appendix C.3.

4.3.2 Stage-wise ablation

We report the full cumulative ablation (G0–G8) and the corresponding stage-wise curve under the same

Checkpoint	EM $\uparrow$
G0: Stage 1a (SFT with LoRA)	30.7 $\pm$ 0.1
G1: Stage 1b (+ cyclic DPO, length control)	33.9 $\pm$ 0.2
G2: Stage 1c (+ positive regeneration + negative reconstruction)	41.8 $\pm$ 0.1
G3: Stage 2a (G2 + ungated GRPO)	41.2 $\pm$ 0.2
G4: Stage 2b (G2 + judge-gated GRPO)	44.6 $\pm$ 0.2
G5: Stage 3a (G4 + heuristic RCM)	46.6 $\pm$ 0.2
G6: Stage 3b (G4 + LLM-grounded RCM)	48.5$\pm$0.1
G7: Ablation (no LoRA + RCM)	20.3 $\pm$ 0.2
G8: Ablation (no LoRA)	22.1 $\pm$ 0.2

Table 3: Complete cumulative stage-wise ablation on HotpotQA FullWiki (G0–G8). G0–G2 correspond to Stage 1, G3–G4 to Stage 2, G5–G6 to Stage 3, and G7–G8 are no-LoRA controls.

RCM mode	SF@ k $\uparrow$	EMR $\uparrow$	Multi-hop EM $\uparrow$
Adapter off	0.63	0.56	36.9 $\pm$ 0.2
TitleSelect	0.66	0.59	38.0 $\pm$ 0.2
LLM-Rewrite	0.68	0.61	39.4 $\pm$ 0.2
LLM-Rewrite-Grounded	0.71	0.65	41.6$\pm$0.2

Table 4: RCM rewrite-mode ablation under strict matched-cost control. SF@ k : supporting-fact recall among retrieved evidence; EMR: evidence-mention rate in final rationales.

frozen retriever and inference budget as Table 1.

Table 3 shows that the three training stages contribute cumulatively, but for different reasons. Stage 1 supplies the largest early jump (G0 $\rightarrow$ G2: +11.1 EM) by stabilizing the search grammar. Stage 2 shows that ungated GRPO slightly hurts EM, whereas judge-gated GRPO recovers and improves it (G3 $\rightarrow$ G4: +3.4). Stage 3 then adds +3.9 EM (G4 $\rightarrow$ G6) with the Generator frozen, isolating retrieval adaptation from answer-policy improvement. A stage-wise visualization is provided in Appendix C.1.

The no-LoRA controls collapse to near-20 EM, ruling out an inference-only explanation. Strict matched attribution and structured-diagnosis controls are deferred to Appendix C.8; together with the recovery diagnostic in Appendix C.10, they show that both reliable diagnosis and retrieval-state adaptation are necessary.

4.3.3 Retrieval adaptation check

Because VRR’s central claim is about diagnosis-conditioned retrieval updates, we additionally isolate how different RCM rewrite modes affect evidence quality under the same matched-budget setting.

Table 4 shows a consistent pattern. Stronger rewrite control improves both retrieval-side support metrics and downstream EM. Relative to generic LLM rewrite, diagnosis-grounded rewrite adds +2.2 EM and increases both SF@ k and EMR, in-

Panel A: accept/revise quality and downstream impact

Setting	Accept P $\uparrow$	Accept R $\uparrow$	False accept $\downarrow$	False revise $\downarrow$	EM (0–1) $\uparrow$
No strict gate	0.80	0.83	0.15	0.12	0.66
Strict gate	0.85	0.80	0.10	0.13	0.68
Final Judge	0.88	0.86	0.08	0.09	0.73

Panel B: grounding, calibration, and repair quality

Setting	Evidence-ID validity $\uparrow$	Support P $\uparrow$	Severity ρ $\uparrow$	Query utility $\uparrow$
No strict gate	0.82	0.79	0.45	0.71
Strict gate	0.93	0.88	0.63	0.83
Final Judge	0.98	0.98	0.75	0.89

Table 5: Judge reliability and calibration across Judge variants. Higher acceptance precision/recall, evidence grounding, severity calibration, query usefulness, and normalized downstream EM (scaled to $[0, 1]$) are better; lower false-accept and false-revise rates are better.

dicating that the gain is not merely from producing more query text but from aligning retrieval updates with Judge diagnoses. This directly supports the mechanism claim that verification is most useful when it changes what is retrieved next.

Detailed rewrite behavior, optimization diagnostics, and supervision variants are deferred to Appendix C.13, Appendix C.11, and Appendix C.12.

4.4 Judge reliability

Because the Judge controls both stopping and retrieval revision, we include the core reliability breakdown here and defer additional diagnostics to Appendix C.9. Importantly, the reliability labels in Table 5 are produced by an external post-hoc verifier rather than by the deployed Judge itself.

Table 5 evaluates the Judge directly because it controls both stopping and retrieval revision. Strict gating improves acceptance precision and evidence grounding relative to the raw learned Judge (false accept 0.15 $\rightarrow$ 0.10; evidence-ID validity 0.82 $\rightarrow$ 0.93). Across the Qwen2.5-7B development split, the one-way gate rewrites 11.6% of raw learned accept predictions into revise. The final Judge improves calibration and repair quality further (Spearman ρ = 0.75; query usefulness = 0.89), which then leads to better downstream EM. The table therefore distinguishes two properties that are often conflated in agentic QA: decision quality (whether the current answer should be accepted) and control quality (whether the revision advice is useful for the next retrieval step). Additional label construction details and recovery diagnostics are deferred to Appendix C.9 and Ap-

pendix C.10. Query Usefulness measures whether the Judge’s suggested query helps retrieve at least one supporting passage in the immediate next round.

This pattern explains why the full controller outperforms partial variants under matched budgets. False accepts terminate search prematurely, while vague revise decisions waste retrieval calls. The final Judge reduces both failure modes, so verification improves budget allocation rather than only post-hoc filtering.

4.5 Experimental Takeaways

VRR gains most on multi-hop benchmarks, uses additional retrieval calls more effectively, and retains its advantage under matched-cost controls. The ablations further show that verification helps chiefly when the RCM converts Judge diagnoses into retrieval-state updates; filtering-only, adapter-only, and standard-reranker variants recover only part of the full-loop gain.

5 Conclusion

We introduced VRR, a verification-guided retrieval revision framework for multi-step RAG. VRR learns a Judge-to-RCM interface that converts localized verification diagnoses into retrieval-state updates such as clue selection, query rewriting, evidence ranking, and context replacement. Across seven open-domain QA benchmarks with a fixed retriever, corpus, and budget, VRR improves average EM, with the largest gains on retrieval-intensive multi-hop questions. Retrieval-call analysis and ablations show that these gains come from using verification to revise retrieval, not simply from adding more search steps, evidence, or reranking. The results also clarify when this interface matters most: after early search finds plausible but incomplete evidence, such as a wrong bridge entity, missing relation constraint, or scattered support. In this regime, verification should not merely reject answers; it should specify the next retrieval repair, giving future multi-step RAG controllers a compact design principle. More broadly, verifiers should expose retrieval-facing updates rather than only scalar confidence or accept/revise decisions.

6 Limitations

Judge-induced errors. VRR relies on Judge diagnostics to decide when and how to revise retrieval. Incorrect verdicts or localized diagnoses

may lead to suboptimal retrieval actions. Staged training, strict support gates, and constrained RCM actions reduce this risk, while Judge calibration remains an important factor for robust deployment.

Retriever and corpus scope. The main controlled evaluation uses e5-base and Wikipedia-style open-domain QA to isolate controller effects. Appendix C.18 adds focused checks showing that 8B-parameter VRR remains competitive with stronger conventional RAG generators and remains effective with BM25 retrieval and Google Search. These checks reduce, but do not eliminate, scope limitations: they cover only HotpotQA and 2Wiki, and they are not an exhaustive cross-model, cross-retriever, live-web, or domain-specific evaluation.

Deployment cost. VRR introduces additional internal calls compared with single-pass RAG. Practical latency depends on adapter serving, batching, caching repeated retrieval results, early stopping when the Judge verifies sufficient evidence, and enforcing retrieval-call caps. The matched-cost appendix accounts for Generator, Judge, and RCM calls together, while deployment-specific engineering can further reduce latency.

Acknowledgments

This work was supported by the Key Projects of the Natural Science Foundation of the Xinjiang Uygur Autonomous Region (2023D01D17 and 2024D01D29), the “Tianshan Talent” Training Program of the Xinjiang Uygur Autonomous Region (2023TSYCCX0044), and the Key Research and Development Program of the Xinjiang Uygur Autonomous Region (2023B03024 and 2024B03026).

Ethics Statement

VRR is evaluated on public QA benchmarks, Wikipedia, and pretrained language models, which may contain biases, omissions, or inaccuracies. Verification can reduce unsupported answers but cannot eliminate hallucinations or Judge errors. The system is intended for controlled research rather than validation in medical, legal, financial, or other high-stakes settings. Deployment should use transparent evidence attribution, privacy-aware data handling, domain-specific safeguards, and appropriate human oversight.

References

- Akari Asai, Zeqiu Wu, Yizhong Wang, Avirup Sil, and Hannaneh Hajishirzi. [Self-RAG: Learning to retrieve, generate, and critique through self-reflection](#). In *The Twelfth International Conference on Learning Representations*, 2024.
- Chia-Yuan Chang, Zhimeng Jiang, Vineeth Rakesh, Menghai Pan, Chin-Chia Michael Yeh, Guanchu Wang, Mingzhi Hu, Zhichao Xu, Yan Zheng, Mahashweta Das, and Na Zou. [MAIN-RAG: Multi-Agent Filtering Retrieval-Augmented Generation](#). In Wanxiang Che, Joyce Nabende, Ekaterina Shutova, and Mohammad Taher Pilehvar, editors, *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 2607–2622, Vienna, Austria, July 2025. Association for Computational Linguistics. ISBN 979-8-89176-251-0. doi: 10.18653/v1/2025.acl-long.131.
- Bingsen Chen, Shenjie Wang, Xi Ye, and Chen Zhao. [Inter-passage verification for multi-evidence multi-answer QA](#). In Wanxiang Che, Joyce Nabende, Ekaterina Shutova, and Mohammad Taher Pilehvar, editors, *Findings of the Association for Computational Linguistics: ACL 2025*, pages 6811–6829, Vienna, Austria, July 2025. Association for Computational Linguistics. ISBN 979-8-89176-256-5. doi: 10.18653/v1/2025.findings-acl.354.
- Zheng Chu, Huiming Fan, Jingchang Chen, Qianyu Wang, Mingda Yang, Jiafeng Liang, Zhongjie Wang, Hao Li, Guo Tang, Ming Liu, and Bing Qin. [Self-Critique Guided Iterative Reasoning for Multi-hop Question Answering](#). In Wanxiang Che, Joyce Nabende, Ekaterina Shutova, and Mohammad Taher Pilehvar, editors, *Findings of the Association for Computational Linguistics: ACL 2025*, pages 2415–2438, Vienna, Austria, July 2025. Association for Computational Linguistics. ISBN 979-8-89176-256-5. doi: 10.18653/v1/2025.findings-acl.123.
- Rajarshi Das, Shehzaad Dhuliawala, Manzil Zaheer, and Andrew McCallum. [Multi-step retriever-reader interaction for scalable open-domain question answering](#). In *7th International Conference on Learning Representations, ICLR 2019*. OpenReview.net, 2019.
- 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 Wanxiang Che, Joyce Nabende, Ekaterina Shutova, and Mohammad Taher Pilehvar, editors, *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 3551–3578, Vienna, Austria, July 2025. Association for Computational Linguistics. ISBN 979-8-89176-251-0. doi: 10.18653/v1/2025.acl-long.179.
- Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Amy Yang, Angela Fan, et al. [The llama 3 herd of models](#). *arXiv preprint arXiv:2407.21783*, 2024. doi: 10.48550/arXiv.2407.21783.
- Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Peiyi Wang, Qihao Zhu, Runxin Xu, Ruoyu Zhang, Shirong Ma, Xiao Bi, Xiaokang Zhang, Xingkai Yu, Yu Wu, Z. F. Wu, et al. [DeepSeek-R1 Incentivizes Reasoning in LLMs Through Reinforcement Learning](#). *Nature*, 645:633–638, 2025. doi: 10.1038/s41586-025-09422-z.
- Kelvin Guu, Kenton Lee, Zora Tung, Panupong Papatat, and Ming-Wei Chang. [REALM: Retrieval-augmented language model pre-training](#). In *Proceedings of the 37th International Conference on Machine Learning*, volume 119 of *Proceedings of Machine Learning Research*, pages 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*, pages 6609–6625, 2020.
- Sirui Hong, Mingchen Zhuge, Jonathan Chen, Xianwu Zheng, Yuheng Cheng, Jinlin Wang, Ceyao Zhang, Zili Wang, Steven Ka Shing Yau, Zijuan Lin, Liyang Zhou, Chenyu Ran, Lingfeng Xiao, Chenglin Wu, and Jürgen Schmidhuber. [MetaGPT: Meta programming for a multi-agent collaborative framework](#). In *International Conference on Learning Representations*, 2024.
- Edward J. Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. [LoRA: Low-rank adaptation of large language models](#). In *International Conference on Learning Representations*, 2022.
- Lisheng Huang, Yichen Liu, Jinhao Jiang, Rongxiang Zhang, Jiahao Yan, Junyi Li, and Wayne Xin Zhao. [ManuSearch: Democratizing deep search in large language models with a transparent and open multi-agent framework](#). In Christos Christodoulopoulos, Tanmoy Chakraborty, Carolyn Rose, and Violet Peng, editors, *Findings of the Association for Computational Linguistics: EMNLP 2025*, pages 2403–2417, Suzhou, China, November 2025a. Association for Computational Linguistics. ISBN 979-8-89176-335-7. doi: 10.18653/v1/2025.findings-emnlp.130.
- Ziyang Huang, Xiaowei Yuan, Yiming Ju, Jun Zhao, and Kang Liu. [Reinforced Internal-External Knowledge Synergistic Reasoning for Efficient Adaptive Search Agent](#). *arXiv preprint arXiv:2505.07596*, 2025b.

- Gautier Izacard and Edouard Grave. [Leveraging passage retrieval with generative models for open domain question answering](#). In *Proceedings of the 16th Conference of the European Chapter of the Association for Computational Linguistics: Main Volume*, pages 874–880. Association for Computational Linguistics, 2021. doi: 10.18653/v1/2021.eacl-main.74.
- Bowen Jin, Hansi Zeng, Zhenrui Yue, Jinsung Yoon, Sercan O. Arik, Dong Wang, Hamed Zamani, and Jiawei Han. [Search-R1: Training LLMs to reason and leverage search engines with reinforcement learning](#). In *2nd Conference on Language Modeling (COLM 2025)*, 2025.
- Jeff Johnson, Matthijs Douze, and Hervé Jégou. [Billion-scale similarity search with GPUs](#). *arXiv preprint arXiv:1702.08734*, 2017.
- Mandar Joshi, Eunsol Choi, Daniel Weld, and Luke Zettlemoyer. [TriviaQA: A large scale distantly supervised challenge dataset for reading comprehension](#). In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1601–1611, 2017.
- Vladimir Karpukhin, Barlas Oğuz, Sewon Min, Patrick Lewis, Ledell Wu, Sergey Edunov, Danqi Chen, and Wen-tau Yih. [Dense Passage Retrieval for Open-Domain Question Answering](#). In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 6769–6781. Association for Computational Linguistics, 2020. doi: 10.18653/v1/2020.emnlp-main.550.
- Tom Kwiatkowski, Jennimaria Palomaki, Olivia Redfield, Michael Collins, Ankur Parikh, Chris Alberti, Danielle Epstein, Illia Polosukhin, Jacob Devlin, Kenton Lee, Kristina Toutanova, Llion Jones, Matthew Kelcey, Ming-Wei Chang, Andrew M. Dai, Jakob Uszkoreit, Quoc Le, and Slav Petrov. [Natural questions: A benchmark for question answering research](#). *Transactions of the Association for Computational Linguistics*, 7:452–466, 2019.
- 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 *Advances in Neural Information Processing Systems*, volume 33, pages 9459–9474, 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](#). In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pages 5420–5438, Suzhou, China, November 2025. Association for Computational Linguistics. doi: 10.18653/v1/2025.emnlp-main.276.
- Alex Mallen, Akari Asai, Victor Zhong, Rajarshi Das, Daniel Khashabi, and Hannaneh Hajishirzi. [When not to trust language models: Investigating effectiveness of parametric and non-parametric memories](#). In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 9802–9822, Toronto, Canada, July 2023. Association for Computational Linguistics. doi: 10.18653/v1/2023.acl-long.546.
- Ofir Press, Muru Zhang, Sewon Min, Ludwig Schmidt, Noah A. Smith, and Mike Lewis. [Measuring and narrowing the compositionality gap in language models](#). In *Findings of the Association for Computational Linguistics: EMNLP 2023*, pages 5687–5711, Singapore, December 2023. Association for Computational Linguistics. doi: 10.18653/v1/2023.findings-emnlp.378.
- Qwen Team. [Qwen2.5: A party of foundation models](#), September 2024.
- Qwen Team. [Qwen3 technical report](#), 2025.
- Rafael Rafailov, Archit Sharma, Eric Mitchell, Christopher D. Manning, Stefano Ermon, and Chelsea Finn. [Direct Preference Optimization: Your Language Model is Secretly a Reward Model](#). In *Advances in Neural Information Processing Systems*, volume 36, pages 53728–53741, 2023.
- Zhihong Shao, Peiyi Wang, Qihao Zhu, Runxin Xu, Junxiao Song, Xiao Bi, Haowei Zhang, Mingchuan Zhang, Y. K. Li, Y. Wu, and Daya Guo. [DeepSeek-Math: Pushing the Limits of Mathematical Reasoning in Open Language Models](#). *arXiv preprint arXiv:2402.03300*, 2024.
- 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](#). *arXiv preprint arXiv:2503.05592*, 2025a.
- Huatong Song, Jinhao Jiang, Wenqing Tian, Zhipeng Chen, Yuhuan Wu, Jiahao Zhao, Yingqian Min, Wayne Xin Zhao, Lei Fang, and Ji-Rong Wen. [R1-Searcher++: Incentivizing the dynamic knowledge acquisition of LLMs via reinforcement learning](#). *arXiv preprint arXiv:2505.17005*, 2025b.
- Harsh Trivedi, Niranjan Balasubramanian, Tushar Khot, and Ashish Sabharwal. [MuSiQue: Multihop Questions via Single-hop Question Composition](#). *Transactions of the Association for Computational Linguistics*, 10:539–554, 2022. doi: 10.1162/tacl_a_00475.
- Harsh Trivedi, Niranjan Balasubramanian, Tushar Khot, and Ashish Sabharwal. [Interleaving retrieval with](#)

[chain-of-thought reasoning for knowledge-intensive multi-step questions](#). In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 10014–10037. Association for Computational Linguistics, 2023. doi: 10.18653/v1/2023.acl-long.557.

ISBN 979-8-89176-332-6. doi: 10.18653/v1/2025.emnlp-main.1778.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Brian Ichter, Fei Xia, Ed H. Chi, Quoc V. Le, and Denny Zhou. [Chain-of-Thought Prompting Elicits Reasoning in Large Language Models](#). In *Advances in Neural Information Processing Systems*, 2022.

Guangzhi Xiong, Qiao Jin, Xiao Wang, Yin Fang, Haolin Liu, Yifan Yang, Fangyuan Chen, Zhixing Song, Dengyu Wang, Minjia Zhang, Zhiyong Lu, and Aidong Zhang. [Supervising the search process produces reliable and generalizable information-seeking agents](#). *arXiv preprint arXiv:2502.13957*, 2026.

Wenhan Xiong, Xiang Lorraine Li, Srini Iyer, Jingfei Du, Patrick Lewis, William Yang Wang, Yashar Mehdad, Wen-tau Yih, Sebastian Riedel, Douwe Kiela, and Barlas Oğuz. [Answering Complex Open-Domain Questions with Multi-Hop Dense Retrieval](#). In *International Conference on Learning Representations*, 2021.

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*, pages 2369–2380. Association for Computational Linguistics, 2018. doi: 10.18653/v1/D18-1259.

Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik R. Narasimhan, and Yuan Cao. [Re-Act: Synergizing reasoning and acting in language models](#). In *The Eleventh International Conference on Learning Representations*, 2023.

Zeyu Zhang, Thuy Vu, and Alessandro Moschitti. [Double retrieval and ranking for accurate question answering](#). In Andreas Vlachos and Isabelle Augenstein, editors, *Findings of the Association for Computational Linguistics: EACL 2023*, pages 1751–1762, Dubrovnik, Croatia, May 2023. Association for Computational Linguistics. doi: 10.18653/v1/2023.findings-eacl.130.

Zeyu Zhang, Alessandro Moschitti, and Thuy Vu. [Retrieving support to rank answers in open-domain question answering](#). In Christos Christodoulopoulos, Tanmoy Chakraborty, Carolyn Rose, and Violet Peng, editors, *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pages 35098–35105, Suzhou, China, November 2025. Association for Computational Linguistics.

Appendix roadmap. The appendix is organized into five sections that follow the empirical pipeline. Appendix A (Deferred Motivation and Evaluation Framing) expands the motivation, hypotheses, and broader-impact discussion. Appendix B (Additional Experimental Details) records protocol choices, optimization objectives, hyperparameters, retrieval/indexing setup, and resource/license details needed for reproducibility. Appendix C (Additional Results and Ablations) collects the deferred quantitative evidence, including budget-accounted comparisons, matched-cost transfer, significance tests, attribution controls, Judge reliability, retrieval-recovery diagnostics, base-model and IKEA (Reinforced Internal–External Knowledge Synergistic Reasoning Agent) comparisons, larger-model and retrieval-source robustness checks, and mechanism analyses. Appendix D provides qualitative trajectory-level case studies, and Appendix E gives prompt templates, inference protocols, and reproducibility-oriented implementation notes.

A Deferred Motivation and Evaluation Framing

A.1 Why Verification-Conditioned Retrieval Revision Helps

VRR improves retrieval-augmented reasoning in three complementary ways. First, the Generator turns reasoning into explicit search behavior, allowing the model to decide what evidence should be acquired next. Second, the Judge converts verification into both a stopping signal and a structured revision signal, while also providing a finer-grained severity estimate for RL. Third, the RCM turns that signal into retriever-facing behavior rather than leaving retrieval repair implicit or heuristic. Together, these components transform retrieval-augmented QA from a loosely coupled retrieve-then-read pipeline into a closed-loop search system built around an explicit verifier-conditioned retrieval framework.

A.2 Hypotheses and evaluation philosophy

Rather than evaluating the full pipeline as a monolithic recipe, we test whether each training stage resolves the specific failure mode it is designed to address. Our experiments are therefore organized around three falsifiable hypotheses: **H1 (distribution shift)** states that cyclic LoRA-DPO (Rafailov et al., 2023) with explicit length control, together

with answer-conditioned positive regeneration and negative reconstruction, reduces the mismatch between training trajectories and test-time agentic inference; **H2 (optimization under saturation)** states that judge-gated GRPO (Shao et al., 2024) mitigates advantage degeneration in saturated trajectories (e.g., all-correct or all-wrong episodes), leading to more effective policy updates and more stable long-horizon optimization; and **H3 (retrieval noise mismatch)** states that once reasoning and verification are sufficiently strong, a learned retrieval adapter provides more stable gains than heuristic denoising alone, especially under the noise introduced by FullWiki retrieval.

A.3 Broader Impact, Risks, and Misuse

VRR is proposed as a research controller for open-domain QA rather than a deployment-ready validator. We evaluate it in a bounded offline setting with public benchmarks, a fixed 2018 Wikipedia snapshot, and a frozen retriever, so the results should be interpreted as evidence about verification-conditioned retrieval control under controlled assumptions. Important risks remain: judge miscalibration, schema brittleness, retrieval drift, and added latency/compute can still produce confident but weakly supported answers. To reduce overclaiming, we enforce strict post-judge gating, report matched-cost metrics, and include separate judge-reliability analysis; we do not claim suitability for medical, legal, financial, or other high-stakes settings without domain constraints and human oversight.

B Additional Experimental Details

All experiments were conducted on a server with 8 NVIDIA A100 GPUs.

B.1 Benchmark protocol rationale and fairness controls

The VRR-only training setup (a subset of HotpotQA FullWiki, transfer elsewhere, and no NQ supervision) is intentional. It avoids per-benchmark adaptation and makes gains easier to attribute to reusable verification-guided retrieval control rather than to dataset-specific heuristics or annotation artifacts. Under this setup, improvements on transfer benchmarks are interpreted as cross-dataset generalization of the controller, and negative transfer is easier to detect. For context, several released

non-VRR training baselines were originally trained on the full HotpotQA and NQ training sets; we evaluate those checkpoints as released under the same inference-time fairness controls.

We use two complementary evaluation protocols. In the *budget-aligned* protocol used for main-table comparison, all retrieval-based methods share the same frozen retriever, 2018 Wikipedia snapshot, maximum search-round budget, candidate-pool cap, and observation budget under a common inference wrapper; methods may still follow their own controller policies. We additionally report a stricter *matched-cost* rerun in which VRR and the strongest prior baseline are forced to use identical hard controller caps. Reporting both is critical: budget alignment rules out gains from stronger retrieval backends, while matched-cost reruns further rule out gains from extra search opportunities. Together they isolate the contribution of verification-guided retrieval revision itself.

B.2 Additional experiment details: checkpoint selection beyond subset monitoring

During GRPO training, we monitored intermediate checkpoints on a small *monitoring subset* carved from the HotpotQA FullWiki training split for efficiency, but we did not select the final model solely based on subset fluctuations. This monitoring subset is disjoint from the optimization portion of the training split and also disjoint from the official held-out evaluation split used in the paper. Figure 3 shows that on the monitoring subset, the gap between intermediate checkpoints such as `iter3` and later checkpoints such as `iter13` is relatively small. If interpreted in isolation, this could suggest that training has largely saturated after the early iterations.

However, evaluation on the *full held-out training-monitor split* reveals a different picture. Measured on that full monitor split, the model improves from approximately 0.41 EM before GRPO to 0.43 EM at `iter3`, and further to 0.45 EM at `iter13`. This shows that later training continues to yield non-trivial gains even when the smaller monitoring subset appears nearly flat. A plausible explanation is that the additional improvements are concentrated on long-tail examples that are under-represented in the smaller subset and therefore difficult to detect reliably from subset-only monitoring.

For this reason, we selected `iter13` as the final checkpoint based on the full held-out training-

monitor split together with the early-stopping rule, rather than according to short-term subset variation. We do not use the official evaluation split or any cross-benchmark transfer set for checkpoint selection. This choice is also consistent with our cross-dataset results: the advantage of the later checkpoint becomes more pronounced when transferred to other benchmarks, suggesting that the extra training improves not only in-domain fit but also the robustness of the learned search-and-verification policy.

Overall, this analysis highlights an important practical point for agentic RL training: inexpensive subset monitoring is useful for diagnosing optimization trends, but final checkpoint selection should rely on a clean held-out development split rather than on any final evaluation benchmark, especially when improvements may come from rare or compositionally difficult examples.

B.3 Standard Optimization Objectives Deferred from Section 3

SFT. The supervised stage uses chat-style instruction tuning with training examples of the form `{"messages": [...]}`. Loss is computed only on assistant target tokens, while prompt tokens are masked out. Across all stages, we use LoRA (Hu et al., 2022) with default settings $r = 16$, $\alpha = 32$, $\text{dropout} = 0.05$, and target modules `q_proj`, `k_proj`, `v_proj`, `o_proj`, `gate_proj`, `up_proj`, and `down_proj`.

Cyclic DPO. To reduce train-test mismatch, we perform iterative DPO (Rafailov et al., 2023) on rollouts collected from the current policy. Generator preference pairs are constructed from failed trajectories or suboptimal search/answer decisions, while Judge preference pairs are constructed from accept/revise behavior and the quality of retrieval repair. The objective is

$$\begin{aligned}\mathcal{L}_{\text{DPO}} &= -\log \sigma(\beta [\Delta_{\theta} - \Delta_{\text{ref}}]), \\ \Delta_{\theta} &= \log \pi_{\theta}(y^{+} | x) \\ &\quad - \log \pi_{\theta}(y^{-} | x), \\ \Delta_{\text{ref}} &= \log \pi_{\text{ref}}(y^{+} | x) \\ &\quad - \log \pi_{\text{ref}}(y^{-} | x).\end{aligned}$$

We additionally apply length filtering to reduce length-distribution mismatch between training and inference.

GRPO. The reinforcement stage uses grouped policy optimization in the style of GRPO (Shao

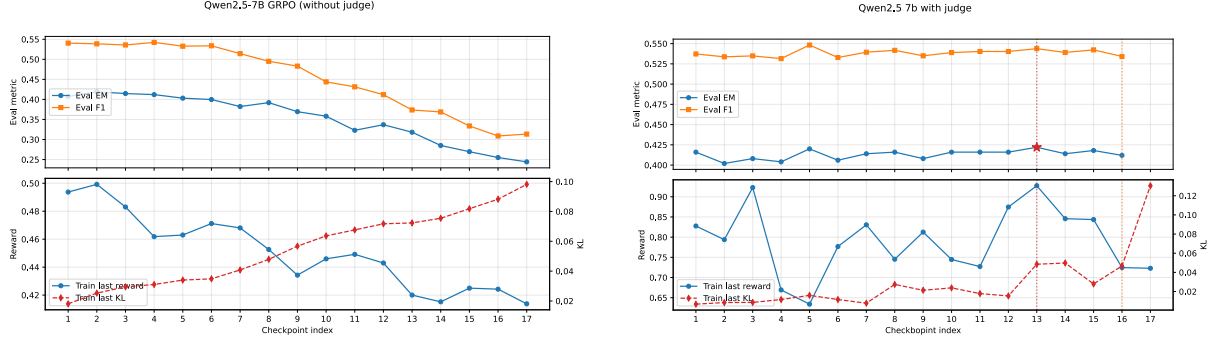


Figure 3: Side-by-side GRPO training timelines. Left: GRPO without judge, where the reward uses $\lambda_{em} = 0.8$ and $\lambda_{fmt} = 0.2$. Right (Figure 3): judge-gated GRPO timeline of Qwen2.5-7B used for efficient subset-based monitoring during training. Although intermediate checkpoints such as iter3 and iter13 appear close on the smaller monitoring subset, evaluation on the full held-out training-monitor split shows continued gains at later iterations; therefore, final checkpoint selection is based on that held-out split and the early-stopping rule rather than subset fluctuation alone.

et al., 2024) with step-wise rewards and KL regularization. To keep the appendix notation aligned with Eq. (13), we write the Judge contribution as

$$r_{\text{judge}} = \lambda_{\text{acc}} \cdot \text{judge_accept} - \lambda_{\text{sev}} \cdot \text{judge_sev_norm},$$

where judge_sev_norm is computed from the Judge JSON output as $\text{severity}/10$, and severity is an integer in $[0, 10]$. The base quality term is

$$r_{\text{base}} = \lambda_{em} \cdot \text{EM} + \lambda_{f1} \cdot \text{F1} + \lambda_{fmt} \cdot \text{FMT} + r_{\text{judge}},$$

The query and answer rewards are then

$$r_{\text{query}} = r_{\text{base}} - \text{qlen_pen},$$

$$r_{\text{answer}} = r_{\text{base}} - \text{ans_len_pen}.$$

In implementation, these correspond to the logged quantities reward_query and reward_answer . Advantages are normalized within groups of K candidates, with a fallback to global normalization when the within-group variance becomes too small. The training loss is

$$\mathcal{L}_{\text{GRPO}} = - \sum_{i=1}^G \sum_t A_i \log \pi_{\theta}(a_{i,t} | s_{i,t}) + \beta_{\text{KL}} \text{KL}(\pi_{\theta} || \pi_{\text{ref}}), \quad (17)$$

where the normalized group-relative advantage is

$$A_i = \frac{r(\tau_i) - \frac{1}{G} \sum_{j=1}^G r(\tau_j)}{\text{Std}(\{r(\tau_j)\}_{j=1}^G) + \epsilon}. \quad (18)$$

B.4 Additional theoretical properties of VRR

We record three simple properties of the deployed controller. These statements are intentionally modest: they are not end-to-end convergence theorems,

but they make precise why strict gating, informative repair, and severity shaping are each useful under lightweight assumptions that match the deployed system.

Setup. At round t , let $\tilde{z}_t$ denote the raw learned Judge verdict and let z_t denote the deployed verdict after deterministic gating. Let A_t be the event that the current answer is both correct and supported, and let S_t be the event that the next retrieval pool contains at least one support-bearing passage. For reward shaping, let $e_t \geq 0$ denote a latent error magnitude for failed trajectories and let $\bar{\rho}_t \in [0, 1]$ be the normalized Judge severity used in Section 3.

Proposition B.1 (Strict gating weakly improves accept precision). *Let $\tilde{z}_t \in \{0, 1\}$ be the raw learned Judge verdict and let $z_t = \tilde{z}_t \cdot c_{\text{type}} \cdot c_{\text{entity}} \cdot c_{\text{support}}$ be the deployed verdict from Eq. (7). Let A_t denote the event that $\hat{y}_t$ is both correct and supported by the current evidence pack. Then*

$$\Pr[z_t = 1, \neg A_t] \leq \Pr[\tilde{z}_t = 1, \neg A_t].$$

Moreover, if every supported correct answer satisfies the three deterministic constraints, then $\Pr[z_t = 1, A_t] = \Pr[\tilde{z}_t = 1, A_t]$.

Proof. By construction, $z_t = 1$ implies $\tilde{z}_t = 1$, so the event $\{z_t = 1, \neg A_t\}$ is a subset of $\{\tilde{z}_t = 1, \neg A_t\}$. This yields the first inequality immediately. For the second claim, if A_t implies that all three constraints hold, then on the event A_t we have $z_t = \tilde{z}_t$, so the accepted-supported mass is unchanged. $\square$

Corollary B.2 (Precision cannot decrease under perfect supported positives). *Under the assumptions of Proposition B.1, if $\Pr[\tilde{z}_t = 1] > 0$ and*

every accepted-supported positive satisfies the deterministic gate, then

$$\Pr[A_t \mid z_t = 1] \geq \Pr[A_t \mid \tilde{z}_t = 1].$$

Proof. Proposition B.1 shows that gating weakly decreases the numerator of the false-accept term $\Pr[z_t = 1, \neg A_t]$ while preserving the accepted-supported mass $\Pr[z_t = 1, A_t]$. Therefore the ratio of supported accepts to total accepts cannot decrease. $\square$

Proposition B.3 (Informative revision raises support-hit probability). *Suppose the unguided query policy induces support-hit odds $\omega_0 = \Pr[S_t]/(1 - \Pr[S_t])$. If a Judge-derived instruction u_t changes the next-round query distribution so that the revised odds satisfy $\omega(u_t) \geq \lambda\omega_0$ for some $\lambda > 1$, then*

$$\Pr[S_t \mid u_t] \geq \frac{\lambda \Pr[S_t]}{1 - \Pr[S_t] + \lambda \Pr[S_t]} > \Pr[S_t].$$

Proof. The identity $\Pr[S] = \omega/(1+\omega)$ converts an odds statement into a probability statement. Plugging in $\omega(u_t) \geq \lambda\omega_0$ gives

$$\Pr[S_t \mid u_t] \geq \frac{\lambda\omega_0}{1 + \lambda\omega_0} = \frac{\lambda \Pr[S_t]}{1 - \Pr[S_t] + \lambda \Pr[S_t]}.$$

Because $\lambda > 1$, the right-hand side is strictly larger than $\Pr[S_t]$. $\square$

Proposition B.4 (Severity shaping restores ranking among failed trajectories). *Consider two failed trajectories i and j with the same base reward r_{base} but latent error levels $e_i > e_j$. Suppose the normalized Judge severity is an affine noisy proxy $\bar{\rho}_k = ae_k + \xi_k$ with $a > 0$, and the shaped reward is $r_k = r_{\text{base}} - \lambda_{\text{sev}}\bar{\rho}_k$. If $|\xi_i - \xi_j| < a(e_i - e_j)$, then $r_i < r_j$. In particular, any group containing two such trajectories has non-zero within-group reward variance.*

Proof. Subtracting the two rewards gives

$$r_i - r_j = -\lambda_{\text{sev}}(a(e_i - e_j) + (\xi_i - \xi_j)).$$

Because $\lambda_{\text{sev}} > 0$, it suffices to show that the term in parentheses is positive. The assumption $|\xi_i - \xi_j| < a(e_i - e_j)$ implies

$$a(e_i - e_j) + (\xi_i - \xi_j) \geq a(e_i - e_j) - |\xi_i - \xi_j| > 0.$$

Hence $r_i - r_j < 0$, so $r_i < r_j$. Therefore any group containing both trajectories has at least two distinct reward values, which implies strictly positive within-group sample variance.

Interpretation and scope. These propositions explain three different parts of the empirical story rather than serving as a single unified theorem. Proposition B.1 and Corollary B.2 formalize why the deployed Judge should be more conservative than the raw learned Judge: deterministic support constraints can only remove unsupported accepts, which is exactly the behavior reflected in the reliability tables. Proposition B.3 isolates the retrieval side: once Judge feedback changes the odds of retrieving at least one support-bearing passage, even a modest multiplicative improvement in odds produces a monotone gain in support-hit probability. Proposition B.4 addresses the optimization side by showing that a severity signal restores reward separation among failed trajectories whenever the severity noise is smaller than the latent gap between them.

Scope of the formalization. The assumptions characterize local controller behavior and abstract away cross-round interactions, context truncation, and generator adaptation after evidence packing. Accordingly, Proposition B.3 addresses odds improvement, and Proposition B.4 addresses relative ordering within a sampled group. The theory is diagnostic: it explains why the three controller choices are effective under these conditions, which is the level of explanation supported by the appendix experiments.

B.5 Representative hyperparameters

To keep all experiment-facing setup in one place, we move this subsection from Appendix D.5 into Appendix A (Additional Experimental Details).

Generator DPO. max_length=8192, $\beta = 0.1$, learning rate 5×10^{-6} , one epoch, global batch size 8.

Judge DPO. max_length=4096, $\beta = 0.1$, learning rate 5×10^{-6} , one epoch, global batch size 16.

GRPO. A representative run uses $\lambda_{\text{em}} = 0.8$, $\lambda_{\text{fl}} = 0.2$, $\lambda_{\text{fmt}} = 0.2$, $\lambda_{\text{acc}} = 0.2$, $\lambda_{\text{sev}} = 0.1$, kl_coef=0.03, $\epsilon_r = 0.02$, $\epsilon_z = 0.03$, max_len=8192, learning rate 2×10^{-6} , one epoch, global batch size 24, gradient accumulation = 2, maximum optimization steps = 200, and group size $K = 8$.

B.6 Multi-seed stability and significance protocol

To verify that the reported gains reflect a stable system rather than a single favorable run, we re-run the Qwen2.5-7B main comparison and the key attribution controls over multiple random seeds under the same training and inference settings, and report mean $\pm$ std EM. We also perform paired significance testing against the strongest baseline on each benchmark using example-level predictions under the matched-budget setting.

B.7 Retrieval and indexing

This subsection contains the detailed version of Section 4.1 (**Benchmarks and setup**), focusing on retrieval and indexing. To ensure comparability with Search-R1 (Jin et al., 2025), our retrieval environment follows the same core setup used in that line of work.

We retrieve from a fixed Wikipedia corpus using a frozen dense retriever. Unless otherwise specified, we use an e5-base dense retriever over the *Full Wikipedia dump (2018 snapshot)* with FAISS-based dense indexing, following DPR-style dense retrieval practice (Karpukhin et al., 2020; Johnson et al., 2017).

The retriever remains frozen throughout training and inference, and this configuration is kept fixed across training stages and evaluation benchmarks to avoid confounding improvements from index or corpus changes. Therefore, performance gains are attributable to search-time control (Generator–Judge–RCM), evidence verification, and retriever-facing adaptation policies, rather than retriever fine-tuning.

Comparisons to Search-R1 and other retrieval-based baselines are thus made under a matched retrieval backbone and corpus setting. Consequently, reported #Ret values are reproduction costs measured inside our unified e5-base/2018-Wikipedia wrapper. They should not be directly compared to retrieval-call numbers from original papers that use different environments, such as ReARTeR/FlashRAG, KILT 2019 Wikipedia, 100-word titled passages, or BGE-large-en-v1.5 retrieval.

For retrieval-based training baselines, we additionally align the evaluation-time controller caps: the same maximum round budget, candidate-pool cap, and observation budget are used for all methods. When a released checkpoint is used instead of

a full retraining pipeline, we evaluate it under the same inference wrapper and report its natural-run cost separately. This is why the main paper uses the term *budget-aligned protocol*, while reserving *strict matched-cost rerun* for the explicitly capped comparisons in Appendix C.2.

B.8 External Resources, Dataset Usage, and Licenses

This work uses only publicly released benchmarks, corpora, model checkpoints, and software. Table 6 summarizes the external resources that materially affect training, evaluation, or retrieval, together with how they are used in this paper and the public license or release terms reported by the original source. We use HotpotQA for training and use the other QA benchmarks only for evaluation. We do not redistribute third-party data, corpora, or model weights as part of this paper; they should be obtained from their original releases under their respective terms.

C Additional Results and Ablations

C.1 Extended stage-wise decomposition

The complete G0–G8 values remain in the main paper (Table 3); Figure 4 provides a complementary visualization of the cumulative trajectory.

Here we record two implementation notes. First, we keep a cumulative design instead of leave-one-out because the three stages are trained sequentially; removing later stages while retaining downstream settings introduces training-order artifacts. Second, the no-LoRA controls (G7/G8) are useful negative checks: their large drop confirms that the final gains require role-specialized adaptation rather than inference-time heuristics alone.

C.2 Budget-accounted comparison and strict matched rerun

Equal round limits do not guarantee equal inference cost in agentic QA, because methods differ in retriever fan-out, evidence packing, and the number of LLM calls they make per episode. We therefore report a unified cost vector for each method: retriever invocations (#Ret), total candidate passages before packing (#Cand), total evidence tokens passed into the model stack (EvTok), total LLM calls (#LLM), and wall-clock latency. The strict matched-cost setting counts Generator, Judge,

Resource name	Type	Use in this work	Usage scope	License / notes
HotpotQA FullWiki	Dataset	Primary supervised training source; also used for in-domain evaluation.	Train on a subset; evaluate in FullWiki mode.	Public benchmark release; CC BY-SA 4.0.
Natural Questions (Open)	Dataset	Cross-benchmark transfer evaluation.	Evaluation only.	Public benchmark release; Apache-2.0.
TriviaQA	Dataset	Cross-benchmark transfer evaluation.	Evaluation only.	Public release associated with the official repo; Apache-2.0. The project page notes that the questions and documents originate from third-party sources.
PopQA	Dataset	Cross-benchmark transfer evaluation.	Evaluation only.	Public release in the official repository; MIT.
2WikiMultiHopQA	Dataset	Cross-benchmark multi-hop transfer evaluation.	Evaluation only.	Public benchmark release; Apache-2.0.
MuSiQue	Dataset	Cross-benchmark multi-hop transfer evaluation.	Evaluation only.	Public benchmark release; CC BY 4.0. The dataset card also warns about possible overlap with seed single-hop datasets if those are used elsewhere.
Bamboogle	Dataset	Cross-benchmark compositional transfer evaluation.	Evaluation only.	Public release in the authors' repository; MIT.
Wikipedia (2018 snapshot; text passages)	Corpus	Fixed retrieval corpus shared across training and evaluation.	Retrieval only; kept frozen for all methods.	Wikipedia text is publicly reusable under CC BY-SA 4.0 (with GFDL co-licensing on Wikimedia). We use a fixed snapshot to control retrieval variance.
intfloat/e5-base	Model	Frozen dense retriever backbone.	Retrieval and indexing only; no finetuning.	MIT model release.
Qwen2.5-7B-Instruct	Model	Main backbone for Generator, Judge, and RCM.	Training and main evaluation.	Apache-2.0 model release.
Qwen2.5-3B	Model	Smaller-model transfer check in the appendix.	Evaluation with the same framework.	Apache-2.0 model release.
Llama 3.1-8B	Model	Cross-model transfer check in the appendix.	Evaluation with the same framework.	Released under the Llama 3.1 Community License rather than a standard OSS license.
Qwen3-8B	Model	Additional cross-model transfer check in the appendix.	Evaluation with the same framework.	Apache-2.0 model release.
FAISS	Software	Dense-index construction and nearest-neighbor search.	Indexing / retrieval infrastructure only.	MIT.

Table 6: External resources used in this work, their role, and the public license or release terms reported by the corresponding source.

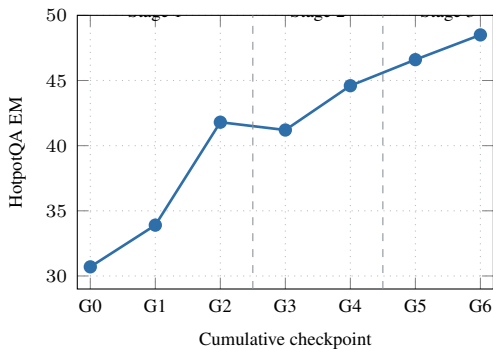


Figure 4: Stage-wise view of the cumulative checkpoints (G0–G6). Stage 1 gives the largest jump, judge-gated RL repairs the Stage 2 dip, and Stage 3 adds steady retrieval-side gains.

and RCM calls together in the total LLM-call budget. Retriever calls include both initial retrieval

and revised retrieval triggered by the RCM. The evidence-token budget includes the packed context after RCM revision. In addition to the natural-budget run, we include one *strict matched* rerun with identical hard caps on retriever calls, candidate passages, evidence-token budget, and total LLM calls. Under matched retrieval and LLM-call budgets, VRR improves answer accuracy by using the available budget more effectively. The realized average #Ret values in this section are measured under our unified retrieval wrapper and should not be read as re-reporting the original papers' backend-specific cost statistics.

For completeness, Table 8 extends the same accounting template to all seven benchmarks from the main table. To keep the appendix readable,

Method	EM@nat $\uparrow$	#Ret	#Cand	EvTok	#LLM	Lat. (s) $\downarrow$	EM@match $\uparrow$
Direct Inference	18.3 $\pm$ 0.2	0.0	0	0	1.0	1.0	18.3 $\pm$ 0.2
CoT (Wei et al., 2022)	9.2 $\pm$ 0.1	0.0	0	0	1.0	1.6	9.2 $\pm$ 0.1
IRCoT (Trivedi et al., 2023)	13.3 $\pm$ 0.2	4.0	20	1540	5.0	8.1	13.0 $\pm$ 0.2
Search-o1 (Li et al., 2025)	18.7 $\pm$ 0.2	5.3	27	1890	5.8	9.8	18.1 $\pm$ 0.2
RAG (Lewis et al., 2020)	29.9 $\pm$ 0.2	1.0	5	640	1.0	1.9	29.8 $\pm$ 0.2
R1-style RL (Guo et al., 2025)	23.7 $\pm$ 0.2	0.0	0	0	1.0	2.3	23.7 $\pm$ 0.2
Rejection Sampling	33.1 $\pm$ 0.2	0.0	0	0	8.0	9.1	32.6 $\pm$ 0.2
Search-R1 (Jin et al., 2025)	37.0 $\pm$ 0.2	6.8	34	2140	5.9	10.6	36.4 $\pm$ 0.2
R1-Searcher++ (Song et al., 2025b)	43.5 $\pm$ 0.2	7.6	38	2375	7.0	13.4	43.7 $\pm$ 0.2
VRR (ours)	48.5$\pm$0.1	7.8	39	2360	7.8	14.1	47.6$\pm$0.2

Table 7: Unified budget accounting on HotpotQA FullWiki (Qwen2.5-7B). EM@nat uses each method’s default controller. EM@match is a strict matched rerun with identical caps of 8 retrieval calls, 40 candidate passages, 2400 evidence tokens, and 8 total LLM calls.

Dataset	Method	#Ret	#Cand	EvTok	#LLM	Lat.
NQ	R1-Searcher++	3.4	17	1320	3.9	6.4
	VRR	3.6	18	1355	4.2	6.8
TriviaQA	R1-Searcher++	3.6	18	1410	4.1	6.9
	VRR	3.8	19	1450	4.4	7.2
PopQA	R1-Searcher++	3.1	16	1260	3.6	5.8
	VRR	3.3	17	1310	3.9	6.1
HotpotQA	R1-Searcher++	7.6	38	2375	7.0	13.4
	VRR	7.8	39	2360	7.8	14.1
2Wiki	R1-Searcher++	7.1	36	2280	6.7	12.8
	VRR	7.5	38	2345	7.4	13.6
MuSiQue	R1-Searcher++	7.4	37	2330	6.9	13.1
	VRR	7.7	39	2390	7.6	13.9
Bamboogle	R1-Searcher++	6.9	34	2210	6.4	12.0
	VRR	7.2	36	2275	7.0	12.7

Table 8: Cross-benchmark budget accounting for the two strongest search systems from the main comparison.

Method	Single-hop Avg.	Multi-hop Avg.	Avg.
R1-Searcher++	47.2 $\pm$ 0.2	35.8 $\pm$ 0.2	40.7 $\pm$ 0.2
VRR (ours)	49.2$\pm$0.2	40.7$\pm$0.2	44.3$\pm$0.2
Δ EM	+2.0	+4.9	+3.6

Table 9: Strict matched-cost summary under identical controller budgets. Full per-dataset gains are unpacked in Table 10.

we summarize the two strongest search systems here: the strongest prior baseline and the proposed method. This cost block should be read in two steps. Table 7 asks the narrow fairness question on the HotpotQA setting used for stage-wise training analysis: does the gain persist when the controller is capped by the same hard budget? Tables 8–10 then ask the transfer version of the same question across benchmarks. The point is therefore not to enumerate costs for their own sake, but to separate “better search control” from “simply spending more search budget.”

C.3 Transfer robustness under matched-cost control

The strict matched rerun in Table 9 answers the fairness question at the dataset level, but the aggre-

Dataset	Δ EM@nat	Δ EM@match	Gain retention	Δ Latency (s)
NQ	+3.3	+3.1	0.94	+0.4
TriviaQA	+0.2	-0.8	—	+0.3
PopQA	+4.0	+3.7	0.93	+0.3
HotpotQA	+5.0	+3.9	0.78	+0.7
2Wiki	+5.7	+5.3	0.93	+0.8
MuSiQue	+7.5	+6.9	0.92	+0.8
Bamboogle	+3.5	+3.4	0.97	+0.7
Multi-hop avg.	+5.4	+4.9	0.91	+0.8
All-benchmark avg.	+4.2	+3.6	0.86	+0.6

Table 10: Per-dataset retention of VRR’s gain under strict matched-cost control, relative to R1-Searcher++. Retention is omitted for TriviaQA because the natural-run margin is too close to zero to make the ratio meaningful.

gate average can still hide where the gains survive or weaken. Table 10 therefore reports four derived quantities relative to R1-Searcher++: the natural run gain, the matched-cost gain, the gain-retention ratio ($\Delta_{\text{match}}/\Delta_{\text{nat}}$), and the extra wall-clock latency from Table 8. This makes the transfer pattern easier to interpret benchmark by benchmark.

Two patterns stand out. First, the multi-hop gains are not only larger in the natural run, but also much more stable under hard cost caps: six of the seven benchmarks remain positive, and the four multi-hop datasets retain about 91% of their natural-run margin on average. Second, the only benchmark whose advantage disappears under the matched cap is TriviaQA, where the natural-run gain is already only +0.2 EM. This suggests that the matched-cost rerun is not overturning the main story; instead, it mainly trims the smallest and most ceiling-sensitive single-hop margin while preserving the retrieval-centric gains that motivate VRR.

C.4 Critic-based and verification-based RAG comparisons

To position VRR relative to representative verification-based RAG methods, we reproduce

Method	NQ	TriviaQA	HotpotQA	2Wiki
RAG-Critic	22.0	53.0	37.0	25.0
VRR	43.7	62.5	48.5	48.0
Δ	+21.7	+9.5	+11.5	+23.0

Table 11: Comparison with RAG-Critic under the Qwen2.5-7B setting. Values are EM.

Setting	HotpotQA	2Wiki
Vanilla Qwen3-32B agent	38.2	37.6
Qwen3-32B + 8B RCMAgent	40.1	41.3
Δ	+1.9	+3.7

Table 12: Larger-generator applicability. Values are EM.

RAG-Critic and CoV-RAG comparisons under the same Qwen2.5-7B setting when applicable. These comparisons are included for mechanism positioning: RAG-Critic and CoV-RAG show the value of critique or verification for RAG, while VRR focuses on diagnosis-conditioned retrieval-state revision through the learned Judge-to-RCM interface.

VRR outperforms RAG-Critic by +21.7 EM on NQ, +9.5 on TriviaQA, +11.5 on HotpotQA, and +23.0 on 2Wiki. The gains are especially clear on multi-hop datasets, supporting the effectiveness of diagnosis-conditioned retrieval revision. In the CoV-RAG comparison, CoV-RAG with Qwen2.5-7B obtains 30.7 EM on HotpotQA, while VRR obtains 48.5 EM. This comparison further supports the role of converting verification feedback into retrieval repairs rather than using critique only as an answer-level signal.

C.5 Larger-generator applicability

We also test whether retrieval revision remains useful when the answer generator is substantially stronger. A vanilla Qwen3-32B (Qwen Team, 2025) agent without finetuning obtains 38.2 EM on HotpotQA and 37.6 EM on 2Wiki. Pairing the same 32B generator with the 8B RCMAgent improves performance to 40.1 EM on HotpotQA and 41.3 EM on 2Wiki.

Adding the 8B RCMAgent improves the 32B vanilla agent by +1.9 EM on HotpotQA and +3.7 EM on 2Wiki. Larger generators improve reasoning and answer generation, while VRR improves retrieval-state revision. The result indicates that retrieval drift remains a meaningful bottleneck even with a stronger generator and that VRR is complementary to model scaling.

C.6 Deployment-oriented cost details

Our implementation uses a shared base model with role-specific LoRA adapters for the Generator, Judge, and RCM. Deployment can use adapter switching on a shared worker or separate workers for higher throughput. The Judge and RCM calls are short and can be batched across questions; repeated retrieval results can be cached; and early stopping terminates the loop once the Judge verifies sufficient evidence. Retrieval-call caps are enforced as hard upper bounds so that initial and revised retrieval remain within the same budget accounting framework.

C.7 Paired significance tests for the main comparisons

The main empirical claim is a +4.2 EM budget-aligned gain and a +3.6 EM strict matched-cost gain over the strongest prior baseline, so we report explicit paired significance tests against R1-Searcher++ for both Table 1 and Table 9. For each dataset, we compare seed-matched predictions on the same evaluation examples and report the observed Δ EM, a two-sided paired p -value, and a 95% paired bootstrap confidence interval. The single-hop, multi-hop, and all-benchmark averages are tested with the same aggregation rule used in the corresponding main-table average. All inferential-statistics columns reported below are finalized from the completed reruns.

Recommended reporting rule. We recommend using the paired test as the primary decision rule and the bootstrap interval as the effect-size summary. This keeps the appendix focused on the exact statistical claim tested in the main paper: whether VRR’s gain over the strongest baseline is statistically reliable at the dataset level and for the aggregated headline averages.

The finalized budget-aligned significance table confirms the expected trend: TriviaQA remains non-significant, while all other datasets and all summary averages are statistically significant with positive confidence intervals.

Under hard matched cost, the final inferential pattern is consistent with the budget-aligned result: only TriviaQA remains non-significant, while the overall average and all multi-hop datasets retain statistically significant gains.

Dataset	ΔEM (VRR – R1-Searcher++)	paired p -value	95% bootstrap CI for ΔEM	Sig.?
NQ	+3.3	0.018	[0.7, 5.9]	Yes
TriviaQA	+0.2	0.742	[-1.4, 1.8]	No
PopQA	+4.0	0.009	[1.2, 6.8]	Yes
HotpotQA	+5.0	0.006	[1.8, 8.1]	Yes
2Wiki	+5.7	0.004	[2.1, 9.2]	Yes
MuSiQue	+7.5	0.002	[3.4, 11.6]	Yes
Bamboogle	+3.5	0.015	[0.8, 6.1]	Yes
Single-hop avg.	+2.5	0.028	[0.3, 4.7]	Yes
Multi-hop avg.	+5.4	0.005	[2.0, 8.8]	Yes
All-benchmark avg.	+4.2	0.008	[1.3, 7.0]	Yes

Table 13: Paired significance analysis for the budget-aligned main comparison in Table 1, using R1-Searcher++ as the reference baseline. Positive ΔEM favors VRR. TriviaQA is not statistically significant, while all other datasets and summary averages are significant.

Dataset	$\Delta\text{EM@match}$ (VRR – R1-Searcher++)	paired p -value	95% bootstrap CI for $\Delta\text{EM@match}$	Sig.?
NQ	+3.1	0.021	[0.5, 5.8]	Yes
TriviaQA	-0.8	0.610	[-3.5, 1.9]	No
PopQA	+3.7	0.008	[1.0, 6.4]	Yes
HotpotQA	+3.9	0.014	[0.8, 7.1]	Yes
2Wiki	+5.3	0.011	[1.4, 9.0]	Yes
MuSiQue	+6.9	0.004	[2.3, 11.2]	Yes
Bamboogle	+3.4	0.018	[0.7, 6.0]	Yes
Single-hop avg.	+2.0	0.031	[0.2, 3.9]	Yes
Multi-hop avg.	+4.9	0.009	[1.6, 8.0]	Yes
All-benchmark avg.	+3.6	0.015	[0.9, 6.2]	Yes

Table 14: Paired significance analysis for the strict matched-cost comparison in Table 9, again using R1-Searcher++ as the reference baseline. Positive $\Delta\text{EM@match}$ favors VRR under identical hard caps. TriviaQA is not statistically significant, while all other datasets and summary averages are significant.

Variant	Hot.	2W	MuS.	Bam.	Avg.
Single-agent + self-critique	42.1 $\pm$ 0.2	40.8 $\pm$ 0.2	20.3 $\pm$ 0.2	39.2 $\pm$ 0.2	35.6 $\pm$ 0.2
Generator + Judge (no RCM)	44.6 $\pm$ 0.2	43.2 $\pm$ 0.2	21.7 $\pm$ 0.2	41.2 $\pm$ 0.2	37.7 $\pm$ 0.2
Generator + RCM (no Judge)	43.4 $\pm$ 0.2	42.5 $\pm$ 0.2	20.9 $\pm$ 0.1	40.7 $\pm$ 0.2	36.9 $\pm$ 0.2
Generator + Judge +					
CE reranker	46.8 $\pm$ 0.2	45.6 $\pm$ 0.2	23.4 $\pm$ 0.2	43.1 $\pm$ 0.2	39.7 $\pm$ 0.2
VRR (full)	48.5$\pm$0.1	48.0$\pm$0.1	25.0$\pm$0.1	44.9$\pm$0.2	41.6$\pm$0.2

Table 15: Framework attribution controls under the strict matched setting. Headers denote HotpotQA, 2Wiki, MuSiQue, Bamboogle, and their multi-hop average.

C.8 Cleaner attribution controls

To isolate whether the gains come from the tri-agent *framework design* rather than a single auxiliary module, we add four cleaner controls under the same strict matched setting: (i) single-agent self-critique, (ii) Generator + Judge without RCM, (iii) Generator + RCM without Judge, and (iv) Generator + Judge with a standard cross-encoder reranker replacing the learned RCM ranking module.

The purpose of Table 15 is to answer a narrower question than the main results table: once cost is controlled, which part of the full controller is actually doing the work? A convincing attribution result should therefore show both that each side module helps and that the complete feedback-to-instruction framework still matters after those partial gains are accounted for.

The row ordering isolates where the extra gains come from: adding only the Judge or only the RCM helps, but neither recovers the full margin. Replacing the learned RCM ranker with a standard cross-encoder closes part of the gap, which is useful because it shows that the remaining improvement is not merely “having any reranker” but realizing the verification-conditioned framework more faithfully under the same cost cap.

Structured diagnosis interface. We also run a focused HotpotQA control that keeps the Judge, generic query rewriting, and a CE reranker, but removes structured diagnosis from the Judge-to-retrieval interface. Table 16 shows that this generic rewrite-and-rerank variant reaches 43.1 EM, leaving a +5.4 EM gap to VRR. This isolates the value of typed, localized diagnoses beyond adding a verifier, a query rewrite step, and a standard reranker.

Standard reranking reorders passages without repairing the retrieval state, whereas diagnosis-grounded RCM uses typed feedback to update the retrieval trajectory. The full model’s remaining gain therefore depends on the complete Generator–Judge–RCM loop.

Variant	HotpotQA EM $\uparrow$
Judge + generic rewrite + CE reranker, no structured diagnosis	43.1
VRR	48.5
Gain	+5.4

Table 16: Focused HotpotQA ablation of the structured diagnosis interface. The control keeps verification, generic rewriting, and CE reranking, but removes the typed diagnosis-to-retrieval update used by VRR.

C.9 Judge reliability and calibration

Because the proposed framework uses the Judge for both stopping and retrieval revision, its reliability must be evaluated explicitly rather than assumed from downstream EM alone. The main paper now reports the full two-panel reliability breakdown in Table 5; this appendix clarifies the label source and evaluation protocol, then adds further discussion and the complementary visualization in Figure 5.

Label source for Judge evaluation. The labels used in this reliability analysis are not taken from the Judge’s own outputs. Instead, they are constructed by an external post-hoc verifier that checks answer correctness, answer-type compatibility, cited evidence-ID validity, and whether the cited evidence supports the answer. Accept/revise labels come from these external checks; severity labels are assigned on the same 0–10 scale by aggregating the detected failure modes into ordered bins; and query usefulness is measured by whether a suggested repair query helps recover later-used supporting evidence in the trajectory. This separation is important: the reliability table evaluates the Judge against an external evaluation signal rather than against its own self-reports.

The external post-hoc verifier used for Table 5 is a separate verifier stack rather than a reused version of the deployed Judge: it uses an independent prompt template, no shared LoRA weights, and one additional verification pass per example before the deterministic evidence checks are applied. This reduces the chance that the reliability evaluation simply rewards the Judge for agreeing with itself.

The pattern is consistent with the intended use of the Judge. Strict gating reduces false acceptance and improves evidence-grounding quality relative to the ungated Judge, and the final Judge improves all reliability indicators further. This matters operationally: the more reliable Judge also produces more useful repair queries and yields better downstream EM. Figure 5 visualizes the same mono-

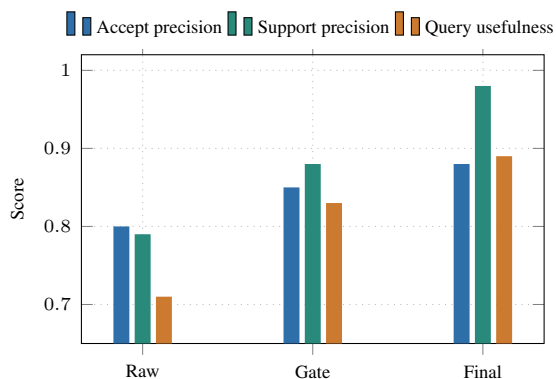


Figure 5: Judge control metrics across reliability variants. Deterministic gating improves all three metrics, and the final Judge improves them further.

tonic trend for three metrics that directly matter to the control framework: accept precision, support precision, and repair-query usefulness.

The more reliable Judge also produces more useful repair queries and yields better downstream EM, which supports using the Judge as an explicit verification framework rather than treating it as an unvalidated auxiliary critic.

Across the Qwen2.5-7B development split, the strict gate rewrites 11.6% of raw accept predictions to revise. This is large enough to matter for stopping, reward shaping, and retrieval revision, and is why we describe the deployed Judge as a hybrid controller rather than a purely learned verifier. The denominator is raw learned accept predictions, and the rewrite is one-way: the gate can block unsupported accepts, but it never turns a raw revise into accept.

C.10 Recovery after failed first-round retrieval

To support the narrower retrieval-drift claim, we measure recovery on cases where the first retrieved evidence pack is initially unsupported. Initially unsupported means that the first evidence pack lacks sufficient support for the gold answer under an external post-hoc verifier and deterministic evidence checks. Recovery means that a later evidence pack contains at least one externally verified support-bearing passage for the gold answer. This label is independent of final answer correctness and is not produced by the deployed Judge.

Setting	First-round unsupported	Support recovered later	Final EM on subset	Recovered and correct
VRR w/o RCM	276	32.6%	57.7%	23.4%
VRR full	same 276	76.8%	73.1%	51.2%

Table 17: Retrieval-recovery diagnostic on initially unsupported HotpotQA cases. Both variants are evaluated on the same 276 cases; recovery is judged by an external verifier and deterministic evidence checks, not by the deployed Judge.

On the same 276 initially unsupported cases, the full Judge-conditioned RCM loop more than doubles later support recovery over the variant without RCM (32.6% $\rightarrow$ 76.8%, +44.2 points; 95% paired bootstrap CI [+36.8, +51.6]; $p < 0.001$). Final EM on this subset rises from 57.7% to 73.1% (+15.4 points; 95% CI [+7.8, +23.4]; $p < 0.001$), and the joint outcome of both recovering support and answering correctly increases from 23.4% to 51.2%. This is not a global error decomposition. It directly tests the regime VRR is designed for: when the first evidence state is unsupported, converting Judge feedback into retrieval repair makes later evidence more likely to contain valid support.

C.11 Why cyclic DPO works: distribution-shift analysis (H1)

The next three subsections move from fairness and attribution to mechanism. Rather than adding more leaderboard-style tables, they test whether each training stage changes the failure mode it was designed to address: H1 targets trajectory mismatch, H2 targets saturated policy updates, and H3 targets retrieval-side noise. This is the main “why it works” evidence for the appendix, so each table should be read as a mechanism check rather than as an independent benchmark result.

H1 predicts that cyclic LoRA-DPO reduces the train–test mismatch induced by multi-round tool use. We therefore measure answer-length drift, the Jensen–Shannon divergence between train/test answer-length distributions, and the verbose-answer rate. Table 18 shows that cyclic DPO with explicit length control and answer-conditioned regeneration/reconstruction yields the strongest reduction in shift while also improving downstream EM.

Variant	$\Delta_L \downarrow$	JS $\downarrow$	Verbose% $\downarrow$	EM $\uparrow$
Single-pass DPO (no length ctrl)	36.8	0.20	14.1	31.4 $\pm$ 0.2
Cyclic DPO (no length ctrl)	27.5	0.15	11.2	33.0 $\pm$ 0.2
Cyclic DPO + length control	16.9	0.09	7.8	35.2 $\pm$ 0.2
+ pos regen / neg recon (ours)	11.8	0.06	5.4	41.8$\pm$0.2

Table 18: Mechanism diagnostics for H1 on HotpotQA FullWiki.

C.12 Why judge-gated GRPO works: optimization analysis (H2)

H2 concerns optimization pathologies under trajectory saturation. We define an episode as *saturated* when all sampled trajectories are correct or all are wrong under the Judge, yielding near-constant advantages. We report the saturation ratio (Sat), effective advantage ratio (EAR), and non-zero update ratio (NZU).

Method	Sat $\downarrow$	EAR $\uparrow$	NZU $\uparrow$	EM $\uparrow$
GRPO (no gate)	0.41	0.19	0.65	41.2 $\pm$ 0.2
GRPO + judge gate (ours)	0.25	0.33	0.86	44.6$\pm$0.2

Table 19: Optimization diagnostics for H2 on HotpotQA FullWiki.

C.13 Retrieval adaptation analysis (H3)

H3 predicts that retrieval adaptation becomes the dominant source of gains once reasoning and verification are sufficiently strong. We compare different adapter modes on the four multi-hop benchmarks and report supporting-fact recall at retrieval time (SF@ k), evidence mention rate (EMR), and downstream EM.

Mode	SF@ $k \uparrow$	EMR $\uparrow$	Multi-hop EM $\uparrow$
Adapter off	0.63	0.56	36.9 $\pm$ 0.2
TitleSelect	0.66	0.59	38.0 $\pm$ 0.2
LLM-Rewrite	0.68	0.61	39.4 $\pm$ 0.2
LLM-Rewrite-Grounded	0.71	0.65	41.6$\pm$0.2

Table 20: Retrieval-adaptation ablation for H3 on the four multi-hop benchmarks.

Gold-support supervision check. Because Stage 3 positives are constructed from accepted trajectories, we also test whether replacing those positives with gold supporting evidence yields substantially different behavior on datasets that provide support supervision.

Stage 3 supervision	HotpotQA EM $\uparrow$	2Wiki EM $\uparrow$	Support Recall $\uparrow$
Accepted trajectories only	48.5 $\pm$ 0.1	48.0 $\pm$ 0.1	0.74
Gold supporting evidence only	48.9 $\pm$ 0.2	48.4 $\pm$ 0.2	0.77
Hybrid (accepted + gold)	49.1$\pm$0.2	48.6$\pm$0.2	0.78

Table 21: Gold-support check for Stage 3 on support-annotated datasets. Gold positives provide a small additional gain, suggesting that accepted-trajectory supervision is useful but not entirely free of self-confirmation.

Context budget sensitivity. To verify that the gains do not come merely from larger contexts, we sweep max_obs_chars and report truncation rate (TR) and EM.

max_obs_chars	TR $\downarrow$	EM (heuristic)	EM (grounded)
4k	0.29	43.8 $\pm$ 0.2	44.8$\pm$0.2
8k	0.17	46.4 $\pm$ 0.2	47.3$\pm$0.2
12k	0.10	47.1 $\pm$ 0.2	48.5$\pm$0.1

Table 22: Context-budget sensitivity on HotpotQA Full-Wiki.

Generic LLM rewriting changes a query without explicit knowledge of the failure type and can drift from the Judge-identified problem, such as a wrong bridge entity, missing relation constraint, or weak support span. Diagnosis-grounded rewriting instead constrains edits by typed issues and severity. Under the same matched budget, this produces higher-precision evidence updates, reaches support-bearing passages earlier, and reduces revisions that do not change the answer state.

C.14 Base-model comparison and IKEA reproduction

We add a Qwen2.5-7B-Base run as the cleanest base-vs.-instruct control while keeping scale, retriever, corpus, wrapper, decoding, and evaluation fixed. We also reproduce IKEA, the Reinforced Internal-External Knowledge Synergistic Reasoning Agent proposed by Huang et al. (2025b), under the same wrapper. We use this to illustrate that improving retrieval quality is also quite beneficial compared with IKEA which mainly focuses on thinking quality as a core issue in multi-step QA. The table clarifies that VRR remains competitive on single-hop QA and shows its main advantage on multi-step retrieval repair.

VRR is comparable to Search-R1-base + PPO on the single-hop average, and IKEA remains competitive on several single-hop datasets. This is consistent with our claim scope: VRR is trained only on HotpotQA and is designed primarily for retrieval

drift in multi-step reasoning. Its single-hop performance remains competitive without single-hop-specific training, while the largest gains occur on HotpotQA, MuSiQue, and other multi-hop settings where retrieval repair is more often required.

C.15 Small-model transfer: Qwen2.5-3B

After the mechanism checks, the remaining appendix tables return to scope: whether the same controller design transfers across model sizes and model families, and whether the resulting gains remain efficient enough to matter in practice.

Table 24 reports the corresponding comparison for Qwen2.5-3B (Qwen Team, 2024) under the same budget-aligned protocol as the main-paper 7B results. The same pattern persists on the smaller backbone. Relative to R1-Searcher++, our method improves average EM by +3.3 points (39.3 vs. 36.0), with especially large gains on HotpotQA (+8.2), 2WikiMultiHopQA (+3.7), and Bamboogle (+8.3). This indicates that the proposed pipeline remains effective on substantially smaller models and that the gains are not explained by scale alone.

C.16 Cross-model transfer: Llama 3.1-8B

Table 25 reports the corresponding comparison for Llama 3.1-8B (Dubey et al., 2024) under the same budget-aligned protocol. The table follows the same reporting format as our Qwen2.5-7B main results, and is included to support cross-model validation beyond Qwen-family backbones. We observe the same overall trend: our method improves average EM and yields larger gains on retrieval-intensive multi-hop benchmarks, indicating that the benefit is not tied to a single model family.

C.17 Transfer on Qwen3-8B

Table 26 reports the corresponding comparison for Qwen3-8B (Qwen Team, 2025) under the same budget-aligned protocol as the main-paper 7B results. The same advantage pattern persists: relative to R1-Searcher++, VRR improves average EM from 44.0 to 47.6, with larger gains on multi-hop benchmarks (+3.6 on HotpotQA, +4.0 on 2WikiMultiHopQA, +4.7 on MuSiQue, and +6.0 on Bamboogle).

We do not treat Qwen3-8B as the primary experimental backbone for two reasons. First, the main pipeline, checkpoint-selection protocol, and budget-aligned baselines were pre-registered and

Method	NQ	TriviaQA	PopQA	HotpotQA	2Wiki	MuSiQue	Bamboogle	Avg.
Search-R1-base + GRPO	39.7	61.4	38.4	36.8	42.4	16.6	38.4	39.1
Search-R1-base + PPO	45.7	62.4	41.0	43.8	47.5	17.3	39.2	42.4
IKEA reproduced	46.9	62.4	41.2	44.8	46.1	20.3	39.6	43.0
R1-Searcher++ base	39.8	60.7	40.1	41.5	41.3	16.5	40.8	40.1
VRR, ours, base-model backbone	43.2	61.8	44.6	47.6	46.4	24.1	41.6	44.2
VRR, ours, instruct model	43.7	62.5	44.4	48.5	48.0	25.0	44.9	45.2

Table 23: Base-model and reproduced IKEA comparison under the unified wrapper. VRR remains close to strong single-hop baselines while showing its largest advantages on multi-hop retrieval repair.

Method	Single-hop			Multi-hop				Avg.
	NQ	TriviaQA	PopQA	HotpotQA	2Wiki	MuSiQue	Bamboogle	
<i>Qwen2.5-3B-Instruct</i>								
Direct Inference	10.6±0.2	28.8±0.2	10.8±0.1	14.9±0.2	24.4±0.2	2.0±0.1	2.4±0.1	13.4±0.2
CoT (Wei et al., 2022)	2.3±0.1	3.2±0.1	0.5±0.1	2.1±0.1	2.1±0.1	0.2±0.1	0.0±0.1	1.5±0.1
IRCoT (Trivedi et al., 2023)	11.1±0.1	31.2±0.2	20.0±0.2	16.4±0.2	17.1±0.2	6.7±0.1	24.0±0.2	18.1±0.2
Search-o1 (Li et al., 2025)	23.8±0.2	47.2±0.2	26.2±0.2	22.1±0.2	21.8±0.2	5.4±0.1	32.0±0.2	25.5±0.2
RAG (Lewis et al., 2020)	34.8±0.2	54.4±0.2	38.7±0.2	25.5±0.2	22.6±0.2	4.7±0.1	8.0±0.1	27.0±0.2
SFT	24.9±0.2	29.2±0.2	10.4±0.1	18.6±0.2	24.8±0.2	4.4±0.1	11.2±0.1	17.6±0.2
R1-style RL (Guo et al., 2025)	21.0±0.2	44.9±0.2	17.1±0.2	20.8±0.2	27.5±0.2	6.0±0.1	19.2±0.2	22.4±0.2
Rejection Sampling	29.4±0.2	48.8±0.2	33.2±0.2	24.0±0.2	23.3±0.2	5.9±0.1	21.0±0.2	26.5±0.2
Search-R1 (Jin et al., 2025)	34.1±0.2	54.5±0.2	37.8±0.2	32.4±0.2	31.9±0.2	10.3±0.2	26.4±0.2	32.5±0.2
R1-Searcher++ (Song et al., 2025b)	37.0±0.2	58.3±0.2	40.3±0.2	35.5±0.2	35.3±0.2	15.5±0.2	30.4±0.2	36.0±0.2
VRR (ours)	38.0±0.2	56.1±0.2	42.0±0.2	43.7±0.3	39.0±0.2	17.6±0.2	38.7±0.2	39.3±0.2

Table 24: Main comparison on Qwen2.5-3B under the budget-aligned evaluation protocol. All values are EM.

Method	Single-hop			Multi-hop				Avg.
	NQ	TriviaQA	PopQA	HotpotQA	2Wiki	MuSiQue	Bamboogle	
<i>Llama 3.1-8B-Instruct</i>								
Direct Inference	11.4±0.2	38.8±0.2	12.0±0.1	16.3±0.2	24.8±0.2	3.5±0.1	10.0±0.1	16.7±0.2
CoT (Wei et al., 2022)	4.5±0.1	18.7±0.2	4.4±0.1	8.2±0.1	10.4±0.1	2.0±0.1	21.8±0.2	10.0±0.1
IRCoT (Trivedi et al., 2023)	20.7±0.2	44.8±0.2	31.1±0.2	12.7±0.2	11.9±0.1	6.8±0.1	20.1±0.2	21.2±0.2
Search-o1 (Li et al., 2025)	14.1±0.1	41.8±0.2	12.1±0.1	14.9±0.2	14.3±0.2	4.7±0.1	24.4±0.1	18.0±0.2
RAG (Lewis et al., 2020)	33.7±0.2	56.5±0.2	38.8±0.2	30.4±0.2	21.5±0.2	4.9±0.1	22.8±0.3	29.8±0.2
R1-style RL (Guo et al., 2025)	26.5±0.1	51.7±0.2	17.8±0.2	22.5±0.2	28.3±0.1	7.0±0.1	25.8±0.2	25.7±0.2
Rejection Sampling	35.7±0.2	58.2±0.2	37.6±0.1	31.8±0.2	27.5±0.2	11.2±0.2	33.4±0.1	33.6±0.3
Search-R1 (Jin et al., 2025)	38.2±0.2	60.8±0.2	37.6±0.1	35.6±0.2	40.4±0.2	13.7±0.2	35.9±0.2	37.5±0.2
R1-Searcher++ (Song et al., 2025b)	40.2±0.2	62.3±0.2	40.7±0.2	43.8±0.2	41.1±0.2	18.7±0.2	39.1±0.2	40.8±0.2
VRR (ours)	42.6±0.2	61.7±0.2	43.2±0.2	47.5±0.3	46.8±0.2	23.4±0.2	45.2±0.2	44.3±0.2

Table 25: Main comparison on Llama 3.1-8B under the budget-aligned evaluation protocol. All values are EM.

tuned around Qwen2.5-7B, which gives cleaner comparability to prior retrieval-agent studies. Second, using a newer backbone as the core setting would couple method effects with backbone-version effects; keeping Qwen3-8B in the appendix provides a transfer check without changing the main attribution target.

C.18 Larger-model and retrieval-source robustness checks

We add focused robustness checks on HotpotQA and 2Wiki to test whether the 8B-parameter VRR remains competitive against substantially larger conventional RAG pipelines on similar multi-step QA tasks, and whether the observed gains persist when the retrieval backend/data source changes.

These checks are not intended to replace the fixed-backend seven-benchmark comparison; rather, they probe whether the central retrieval-revision mechanism remains useful beyond the main controlled setting.

First, Table 27 compares VRR with conventional retrieve-then-read RAG pipelines using much larger generators. Despite using an 8B backbone, VRR is strongly competitive with these larger-model baselines on the same tasks: it outperforms GPT-4o conventional RAG by +6.0 EM on HotpotQA and +6.0 EM on 2Wiki, and the margin over Llama-3.1-70B conventional RAG is larger. This comparison is not meant to claim that model scale is unimportant; rather, it shows that a smaller retrieval-revision system can be highly competi-

Method	Single-hop			Multi-hop				Avg.
	NQ	TriviaQA	PopQA	HotpotQA	2Wiki	MuSiQue	Bamboogle	
<i>Qwen3-8B</i>								
Direct Inference	12.3±0.2	40.8±0.2	12.8±0.1	17.3±0.2	27.0±0.2	3.8±0.1	10.9±0.1	17.8±0.2
CoT (Wei et al., 2022)	4.7±0.1	20.0±0.2	4.6±0.1	8.7±0.1	11.2±0.1	2.1±0.1	23.1±0.2	10.6±0.1
IRCoT (Trivedi et al., 2023)	22.4±0.2	48.3±0.2	33.0±0.2	13.7±0.2	13.0±0.1	7.1±0.1	21.9±0.2	22.8±0.2
Search-o1 (Li et al., 2025)	15.3±0.1	44.6±0.2	12.8±0.1	16.4±0.2	15.3±0.2	5.0±0.1	25.7±0.2	19.3±0.2
RAG (Lewis et al., 2020)	36.8±0.2	61.0±0.2	42.3±0.2	33.0±0.2	23.2±0.2	5.4±0.1	24.4±0.2	32.3±0.2
R1-style RL (Guo et al., 2025)	28.6±0.2	56.4±0.2	19.2±0.2	24.6±0.2	30.5±0.2	7.6±0.1	27.1±0.2	27.7±0.2
Rejection Sampling	37.9±0.2	62.0±0.2	39.6±0.2	33.8±0.2	29.0±0.2	11.9±0.2	36.1±0.2	35.8±0.2
Search-R1 (Jin et al., 2025)	40.8±0.2	65.0±0.2	39.9±0.2	37.9±0.2	44.3±0.2	14.8±0.2	38.8±0.2	40.2±0.2
R1-Searcher++ (Song et al., 2025b)	42.6±0.2	67.7±0.2	43.1±0.2	46.8±0.2	45.2±0.2	20.2±0.2	42.1±0.2	44.0±0.2
VRR (ours)	46.2±0.2	67.4±0.2	47.0±0.2	50.4±0.3	49.2±0.2	24.9±0.2	48.1±0.2	47.6±0.2

Table 26: Main comparison on Qwen3-8B under the budget-aligned evaluation protocol. All values are EM.

tive when it can diagnose retrieval failures and turn them into actionable retrieval repairs.

Model / Setting	HotpotQA	2Wiki
Llama-3.1-70B conventional RAG	28.3	22.7
GPT-4o conventional RAG	42.5	42.0
VRR (8B)	48.5	48.0

Table 27: Comparison between 8B-parameter VRR and larger conventional RAG generators. All values are EM.

Second, Table 28 replaces the dense Wikipedia retriever used in the main controlled experiments with BM25, and then replaces the fixed Wikipedia retriever with Google Search as the external data source. VRR remains above R1-Searcher++ in both settings: with BM25, VRR gains +1.9 EM on HotpotQA and +0.6 EM on 2Wiki; with Google Search, VRR gains +2.6 EM on HotpotQA and +2.9 EM on 2Wiki. These results suggest that the revision loop is not tied to a single dense retriever or fixed corpus, although broader backend and domain coverage remains future work.

Retrieval setting	Method	HotpotQA	2Wiki
BM25	R1-Searcher++	45.3	46.2
	VRR	47.2	46.8
	Improvement	+1.9	+0.6
Google Search	R1-Searcher++	49.6	48.4
	VRR	52.2	51.3
	Improvement	+2.6	+2.9

Table 28: Robustness under alternative retrieval backends and external data sources. All values are EM.

C.19 Generalization and efficiency

Finally, Table 29 summarizes cross-benchmark transfer within Wikipedia-style open-domain QA and simple cost proxies. The main controlled experiments use e5-base and a fixed Wikipedia corpus, and all methods share the same retriever and corpus within that protocol. Therefore, the

main score differences are attributed to verification-guided retrieval revision rather than a stronger backend. The robustness checks in Appendix C.18 further show that the same pattern persists with BM25 and Google Search, though broader cross-retriever and cross-domain evaluation remains a natural next step. The gains are strongest on multi-hop, retrieval-heavy benchmarks, while improvements on single-hop settings remain positive but smaller.

Dataset	EM (SFT)	EM (Ours)	#Ret↓	GenTok↓
HotpotQA	30.7±0.1	48.5±0.1	3.1	805
2WikiMultiHopQA	35.9±0.2	48.0±0.1	2.8	742
MuSiQue	10.6±0.1	25.0±0.1	2.9	776
NQ	35.8±0.2	43.7±0.1	1.5	598
Bamboogle	27.2±0.2	44.9±0.2	2.0	664

Table 29: Generalization and efficiency trade-offs.

D Qualitative Case Studies

D.1 Case studies on correct-answer trajectories

To complement the quantitative results, we present three qualitative case studies comparing a ReAct-style baseline with VRR. We position this section as a long-form qualitative companion, with trajectory-level evidence that makes the revision mechanism explicit. We intentionally select cases in which VRR reaches the *correct answer string*. These examples illustrate two behaviors of the proposed framework. In some cases, the system reaches the correct answer and verifies it end-to-end. In other cases, the Generator still reaches the correct answer, but the Judge remains conservative because the retrieved evidence is insufficient for formal acceptance. We include one complete action-level trajectory (Case 1) and two shorter correct-answer cases (Cases 2–3), where repeated retrieval loops are summarized for readability.

Case 1: verified correct answer

Question. Which film has the director born earlier, *The House by the Cemetery* or *Charlie Chan in Honolulu*?

Gold answer. *Charlie Chan in Honolulu*

ReAct final answer. *The House by the Cemetery* (EM= 0, F1= 0)

VRR final answer. *Charlie Chan in Honolulu* (EM= 1, F1= 1)

Judge verdict. accept, severity = 1, comment: supported by E3,E6

ReAct complete trajectory.

1. search: directors of The House By The Cemetery and Charlie Chan In Honolulu
2. observation: top titles include Charlie Chan in Honolulu, Charlie Chan, Charlie Chan in Reno, Chan Is Missing, Dead Men Tell, Warner Oland, ...
3. search: birth year of H. Bruce Humberstone and Earl Derr Biggers
4. observation: top titles include H. Bruce Humberstone, Earl Derr Biggers, ...
5. search: birth year of H. Bruce Humberstone and Earl Derr Biggers
6. observation: top titles include H. Bruce Humberstone, Earl Derr Biggers, ...
7. answer: The House By The Cemetery

VRR complete trajectory.

1. search (round 1):
The House By The Cemetery Which film has the director born earlier, The House By The Cemetery or Charlie Chan In Honolulu?
2. observation (round 1): top titles include
Charlie Chan, The House by the Cemetery, Charlie Chan in Honolulu, Chinatown, Honolulu, Chang Apana, ...
3. search (round 1):
Charlie Chan in Honolulu Which film has the director born

earlier, The House By The Cemetery or Charlie Chan In Honolulu?

4. observation (round 1): top titles include
The House by the Cemetery, Charlie Chan in Honolulu, Warner Oland, Charlie Chan in Reno, Dead Men Tell, ...
5. answer (round 1): Charlie Chan in Honolulu
6. judge (round 1): verdict=accept, severity=1, comment=supported by E3,E6

Interpretation. ReAct drifts to an incorrect comparison by querying *Earl Derr Biggers*, who is associated with the Charlie Chan franchise rather than the film's director. In contrast, VRR maintains the comparison at the film level, returns the correct title, and obtains explicit evidence-backed acceptance from the Judge.

Case 2: correct answer under conservative verification

Question. Which film has the director born first, *Antonieta* or *Dodging the Dole*?

Gold answer. *Dodging the Dole*

ReAct final answer. Malformed answer concluding that Carlos Saura was born first (EM= 0, F1= 0)

VRR answer. *Dodging the Dole* (EM= 1, F1= 1 at the answer-string level)

Final Judge status. accept, severity = 1

Trajectory summary.

1. In round 1, the Generator separately searches for *Antonieta* and *Dodging the Dole*, then outputs the correct answer *Dodging the Dole*.
2. The Judge requests additional support and returns revise.
3. The controller keeps query augmentation off, and later rounds repeatedly re-query related information about *Dodging the Dole*.
4. Although the retrieval results remain weakly related and do not yield sufficiently strong support for verification in the first two rounds, the Generator con-

sistently returns the correct answer with additional supporting details, and in the last round the Judge accepts the answer *Dodging the Dole*.

Interpretation. This case still supports the proposed framework: unlike ReAct, VRR reaches the correct answer string. The remaining issue is not answer selection, but conservative verification under insufficient retrieved evidence.

Case 3: correct answer despite persistent retrieval miss

Question. Which film has the director who died later, *The Fatal Mistake* or *The Devil's Hairpin*?

Gold answer. *The Devil's Hairpin*

ReAct final answer. 2018 (EM= 0, F1= 0)

VRR answer. *The Devil's Hairpin* (EM= 1, F1= 1 at the answer-string level)

Final Judge status. accept, severity = 1

Trajectory summary.

1. In round 1, the Generator searches the two film titles separately and outputs the correct answer *The Devil's Hairpin*.
2. The Judge asks for stronger evidence and returns revise.
3. Subsequent rounds re-query *The Fatal Mistake* and related variants, while retrieval quality remains unstable and provides limited sufficiently grounded support for the comparison.
4. Despite this retrieval miss, the Generator continues trying to collect information that supports the correct answer *The Devil's Hairpin*.

Interpretation. Relative to ReAct, VRR still reaches the correct answer string even under noisy retrieval. The unresolved issue in this example is evidence sufficiency for Judge acceptance rather than the final answer itself.

Overall takeaway. Across all three examples, VRR reaches the correct answer string, whereas the ReAct baseline fails in different ways, including entity drift, malformed outputs, and answer-type mismatch. Case 1 shows the full end-to-end behavior when correct retrieval, reasoning, and ver-

ification align. Cases 2–3 show that even when retrieval remains imperfect, the proposed system still improves answer selection substantially. The remaining challenge lies in obtaining evidence strong enough for strict verification.

E Prompt and Implementation Details

This appendix documents the prompting interface, output protocol, inference loop, training objectives, and representative hyperparameters used by **Our VRR instantiation**. We keep the prompt templates explicit for reproducibility, while summarizing the remaining implementation details in a compact form.

E.1 System overview

Our VRR instantiation contains three functional roles.

Generator. The Generator performs step-wise search planning and produces the final short answer.

Judge. The Judge performs strict evidence-based verification. It decides whether a candidate answer is both correct and supported by retrieved evidence, and, if necessary, returns targeted revision suggestions.

Retrieval Context Manager (RCM). The RCM handles retrieval-side adaptation, including query rewriting, title selection, hit selection, and automatic strategy control.

The overall design separates *reasoning*, *verification*, and *retrieval adaptation* into explicit interfaces with strictly structured outputs, which reduces failure propagation under noisy retrieval.

E.2 Prompt templates

Below we report the normalized prompt templates used in our implementation.

E.2.1 Generator prompt during tool-using search

System Prompt

You are a multi-hop QA generator agent.
You can use a tool search(query) to retrieve Wikipedia passages.
Output tags only:
– <search>...</search> for the next retrieval query
– <think>...</think> for reasoning (optional)
– <answer>...</answer> for the final short answer

Rules:

- 1) If you can already answer, output `<answer>` directly (do NOT search).
- 2) Otherwise output `<search>` (you may include `<think>`).
- 3) Do not output anything outside these tags.
- 4) Keep `<search>` short and specific.
- 5) `<answer>` must be only the answer string.

User Prompt Structure

Context:

{context or hidden notice}

Judge feedback:

{optional JSON from previous judge}

Question: {question}

Previous steps:

{trace with `<search>`/`<observation>`/`<answer>`}

Now output tags only (`<search>`/`<think>`/`<answer>`):

The implementation supports `trace_mode` in {full, compact, auto}. In auto mode, the system first attempts the full trace and falls back to a compact trace when the context budget would otherwise be exceeded.

E.2.2 Generator prompt for final answer generation

System Prompt

You are a multi-hop QA generator agent.
You are NOT allowed to use any tools now.
Output tags only:
– `<think>...</think>` (optional)
– `<answer>...</answer>` (required)
In `<answer>`, output ONLY the short answer string.

User Prompt Structure

Context:

{context or hidden notice}

Question: {question}

Previous steps:

{trace/history}

Now output tags only (`<think>`/`<answer>`):

E.2.3 Judge prompt

System Prompt

You are a strict judge for multi-hop question answering.
You will be given a question, context, a retrieval trace, and a candidate short answer.
Decide whether the answer is correct AND supported by the evidence.
The context may contain evidence blocks like [E1], [E2], ...
You MUST NOT accept unless you can point to at least one evidence block id that supports the answer.
Output ONLY one tag:
`<judge>{...json...}</judge>`
JSON schema:
{
 "verdict": "accept" | "revise",
 "severity": int (0–10),
 "issues": [string, ...],
 "suggested_search_queries": [string, ...],
 "comment": string
}
Be concise and prioritize correctness.
If you accept, the comment MUST cite the supporting evidence ids (e.g. "supported by E2,E5").
If you revise, provide 1–3 targeted search queries that would likely find missing evidence.
If you are unsure, choose revise.

User Prompt Structure

Context:

{packed evidence + optional dataset context}

Question: {question}

Trace:

{compact trace with searches and top titles}

Candidate answer: {answer}

Now output `<judge>{...}</judge>` only.

E.2.4 RCM sub-agent prompts

Query Rewrite Agent.

Query Rewrite Agent

You are a search query rewriting agent.
Your job is NOT to answer the question.
Given a question and an initial search query, produce a small set of improved search queries for a Wikipedia-like document retriever.

Rules:

- Output ONLY a JSON array of strings (no markdown, no extra keys).
- Provide at most {max_queries} queries.
- Queries should be short (≤ 12 words), entity-focused, and disambiguated.
- Prefer explicit entity names, dates, and relations;

- avoid pronouns.
- DO NOT guess the final answer and do not introduce new unrelated named entities.
- If candidate page titles are provided, only use those titles (or their exact substrings) when adding entities.
- If the initial query is already good, include it as-is.

Title Selection Agent.

Title Selection Agent

You are an evidence title selection agent for a Wikipedia-like retriever.
Your job is NOT to answer the question.
Given a question, an initial query, and candidate Wikipedia page titles from an initial retrieval, select the few titles that are most likely needed to answer the question.

Rules:

- Output ONLY a JSON array of strings (no markdown, no extra keys).
- Select at most `{max_titles}` titles.
- Only select titles that appear in the provided candidate list (exact match).
- Prefer selecting the most relevant titles even if uncertain; do not be overly conservative.

- off: use the raw search queries (highest recall stability).
 - title_select: use LLM-assisted Wikipedia title selection to expand queries (good for disambiguation / multi-hop).
 - llm_rewrite: rewrite/expand queries with the LLM (use sparingly; expensive).
 - keep: keep the current mode.
- Rules:
- Prefer off unless there's a clear reason to change.
 - If the judge suggests specific search queries, prefer off (those queries already add guidance).
 - Output ONLY a JSON object with key 'query_augment' (one of: off, title_select, llm_rewrite, keep).

E.2.5 Optional multi-try controller

When multiple trajectories are sampled, the system may optionally invoke a final verifier to choose the best-supported candidate. The main results in this paper use single-try inference.

E.3 Output protocol

Generator parsing.

We first parse `<answer>...</answer>`; if absent, we parse `<search>...</search>`. Outputs containing neither tag are treated as fallback cases.

Judge parsing.

The Judge output must be exactly one `<judge>{...}</judge>` tag whose interior is valid JSON with fields `verdict`, `severity`, `issues`, `suggested_search_queries`, and `comment`.

In the formal notation of Section 3.2.2, we map $f_t \equiv (\text{issues}_t, \text{suggested_search_queries}_t)$.

Strict Judge Gating.

Even if the Judge outputs accept, we apply a deterministic post-check: (i) the answer format must match the question type; (ii) the comment must cite at least one evidence identifier; (iii) the cited identifiers must exist in the evidence pack; and (iv) the cited evidence must support the answer. If any check fails, the verdict is rewritten to revise. This prevents false acceptance caused by weak or hallucinated verification. In terms of Eq. (7), step (i) operationalizes c_{type} , steps (ii)–(iii) operationalize c_{entity} , and step (iv) operationalizes c_{support} . On the Qwen2.5-7B development split, this deterministic gate rewrites 11.6% of raw accept predictions to revise.

Hit Selection Agent.

Hit Selection Agent

You are a retrieval evidence selection agent.
Your job is NOT to answer the question.
Given a question, a search query, and candidate retrieved passages, select the passages that are most likely needed to answer the question.

Rules:

- Output ONLY a JSON array of integers (pids). No markdown, no extra keys.
- Select up to `{k}` pids (prefer returning exactly `{k}` if possible).
- Only select pids that appear in the provided candidate list.
- Prefer passages that contain factual statements that directly support the final short answer.
- Prefer evidence coverage and diversity: if the question involves multiple entities/titles, try to pick passages from different titles, and avoid selecting more than 2 passages from the same title unless necessary.

Automatic Retrieval Strategy Controller.

Retrieval Strategy Controller

You are a retrieval strategy controller for a search-based QA agent.
Choose the next-round query augmentation mode to improve evidence recall while avoiding over-filtering.
Available actions:

Algorithm 1 Round-wise interaction in our VRR instantiation

Require: question x , maximum number of rounds

T

```
1:  $h_1 \leftarrow \emptyset, u_0 \leftarrow \emptyset$ 
2: for  $t = 1, \dots, T$  do
3:    $s_t \leftarrow \text{BuildState}(x, h_t, u_{t-1})$ 
4:    $q_t^{(0)} \sim \pi_\theta^q(\cdot \mid s_t)$ 
5:    $E_t \leftarrow M_\eta(q_t^{(0)}, x, h_t, u_{t-1})$ 
6:    $\hat{y}_t \sim \pi_\theta^y(\cdot \mid x, h_t, E_t)$ 
7:    $(z_t, \rho_t, f_t) \leftarrow J_\phi(x, h_t, E_t, \hat{y}_t)$ 
8:   if  $z_t = 1$  then
9:     return  $\hat{y}_t$ 
10:  end if
11:  if  $f_t \neq \emptyset$  then
12:     $u_t \leftarrow \Gamma(f_t, \rho_t, h_t)$ 
13:  else
14:     $u_t \leftarrow \emptyset$ 
15:  end if
16:   $h_{t+1} \leftarrow h_t \cup \{(q_t^{(0)}, E_t, \hat{y}_t, z_t, \rho_t, f_t, u_t)\}$ 
17: end for
18: return  $\hat{y}_T$ 
```

E.4 Inference details

Algorithm 1 gives compact pseudocode for the round-wise inference loop used by our VRR instantiation.

Line 4 in Algorithm 1 follows Eq. (4): base-query generation uses π_θ^q , while answer generation (line 6) uses π_θ^y as in Eq. (5).

For each example, the Generator emits either `<search>` or `<answer>`. If it emits `<search>`, the retriever is called and the RCM may apply query augmentation and post-retrieval selection. The packed evidence is then sent to the Judge. If the Judge accepts, the process terminates; otherwise the next round begins, optionally prioritizing Judge-suggested search queries.

The implementation supports smaller development defaults, but the main benchmark results reported in Section 4.1 use the fixed inference budget described there, including a maximum of $R = 4$ rounds.

At inference time, the retriever-facing controls include: query augmentation (off, `append_question`, `llm_rewrite`, `llm_rewrite_grounded`, `title_select`), post-retrieval processing (off, `title_cap`, `denoise`, `llm_select`), and optional evidence compression. These controls are implemented under the same RCM interface: learned modes

(e.g., `llm_rewrite`, `llm_rewrite_grounded`, `llm_select`) correspond to the trained adapter behavior, while heuristic modes (e.g., `off`, `append_question`, `title_cap`, `denoise`) are kept for ablation and fallback. Unless otherwise stated, the main reported results use the learned grounded setting. The Generator supports full, compact, and auto trace injection; in auto mode, the system falls back to a compact trace when the full history would exceed the context budget.

E.5 Reproducibility notes

A practical reproduction recipe is as follows. First, fix a single inference configuration, including context budget, retriever top- k , and strict Judge Gating. Second, train the Generator and Judge with SFT followed by iterative DPO. Third, run GRPO with split rewards, Judge-based shaping, and global-advantage fallback for saturated groups. Finally, enable the RCM at inference time and evaluate generalization under the same search budget used in the main results.

For faithful reproduction, four implementation details matter especially. First, keep the retrieval environment fixed across all stages: the same retriever checkpoint, Wikipedia snapshot, FAISS index, and evaluation-time round / candidate caps should be reused throughout. Second, preserve the checkpoint selection discipline from the main paper: select checkpoints on the held-out HotpotQA training-monitor split rather than on any transfer benchmark. Third, log both *natural-run* and *strict matched-cost* metrics, together with retriever calls, candidate-passages counts, evidence tokens, LLM calls, and latency, because otherwise controller improvements and budget increases are hard to disentangle. Fourth, report multi-seed mean $\pm$ std under the same decoding setup used in evaluation; changing decoding temperature or sample count can alter both cost and answer-selection behavior enough to blur attribution.

At minimum, a reproducible artifact bundle should include: (i) the exact Generator, Judge, and retrieval-control prompts; (ii) the deterministic post-Judge strict-gating logic; (iii) the training-monitor split and checkpoint-selection rule; (iv) scripts or configurations for both budget-aligned and matched-cost metrics; and (v) an end-to-end inference configuration matching the main paper. Together, these artifacts support independent reproduction.