RIPPLE: CORPUS-SCALE RETRIEVAL PLAN OPTIMISATION VIA ITERATIVE FAILURE-DRIVEN SUPERVISION

Anonymous authors
Paper under double-blind review

ABSTRACT

Retrieval-augmented generation (RAG) increasingly relies on iterative search when evidence cannot be recovered in a single retrieval pass. Existing iterative and RL-based search policies reformulate queries across search rounds, but fall short in certain aspects: the subqueries and documents considered per round bound how much of the corpus is ever brought into view, and although the planner adjusts its queries round to round, it does so mainly by reading its own accumulated search history within its context, with no trainable signal for how well a round resolved the information need, leaving little basis to learn which decisions closed the evidence gap. We introduce RIPPLE, which casts a wide initial retrieval net and compresses it with maximal marginal relevance (MMR) into a compact, diversity-optimized snapshot that anchors the planner's corpus view from the first round. Conditioned on this snapshot and prior search history, the planner issues subqueries; an LLM judge evaluates retrieved evidence and summarizes it as feedback for the next round. We train the planner with Group Relative Policy Optimization (GRPO), judging the answer generated from evidence gathered at each round to produce a reward at the level of individual rounds rather than only at the episode's end. On MSQA, TechQA, and ANTIQUE, RIPPLE improves LLM-based scores by up to 7% and CAR by up to 15% over strong baselines, including Search-R1, with ablations confirming both the diverse snapshot and search-round-level supervision are independently necessary. These gains generalize across planner backbones, an independent evaluator, multi-hop benchmarks (HotpotQA, 2WikiMultiHopQA), and zero-shot transfer, while also reducing search rounds needed.

1 INTRODUCTION

Retrieval-augmented generation (RAG) has become a dominant paradigm for augmenting Large Language Models (LLMs) with external knowledge and improving factual, knowledge-intensive question answering (Lewis et al., 2020a; Yu et al., 2024; Rezaei & Dieng, 2025): the quality and coverage of retrieved evidence bound the generator's information and resulting answer (Lewis et al., 2020b; Salemi & Zamani, 2024), and missing evidence increases the risk of unsupported or incomplete responses (Sun et al., 2025; Huang et al., 2025a; Belém et al., 2025; Rykov et al., 2025; Huang et al., 2025b). Yet a common RAG paradigm retrieves once from the original query and generates directly, without adapting to what has or has not been uncovered, acutely limiting complex, multi-faceted, or underspecified needs (Fang et al., 2024; Jiang et al., 2025b) whose evidence is scattered across the corpus. Human search instead begins with broad queries, inspects results, identifies gaps, and reformulates based on what search reveals (Liu et al., 2024; Katz et al., 2024; White & Roth, 2009; Hearst, 2009; Mo et al., 2023), inspiring a growing line of work that reframes RAG as *iterative search*, a closed-loop process that adapts search to its own outcomes.

Iterative and agentic RAG methods realize this closed loop by interleaving reasoning and retrieval through explicit planning (Yao et al., 2023; Press et al., 2023; Zhou et al., 2023), self-reflective retrieval control (Shinn et al., 2023; Gou et al., 2024; Patnaik et al., 2025a; Asai et al., 2024), inference-time tool use (Pan et al., 2025; Lumer et al., 2024; Schick et al., 2023), or RL-based query reformulation across search rounds (Jin et al., 2025; Li et al., 2025b; Jiang et al., 2025a; Tan et al., 2026). While these methods improve evidence coverage over single-pass retrieval, two aspects remain partially addressed. First, the subqueries and documents retrieved per round bound how much of the corpus is exposed to the planner, constraining subsequent queries and assessment of missing

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

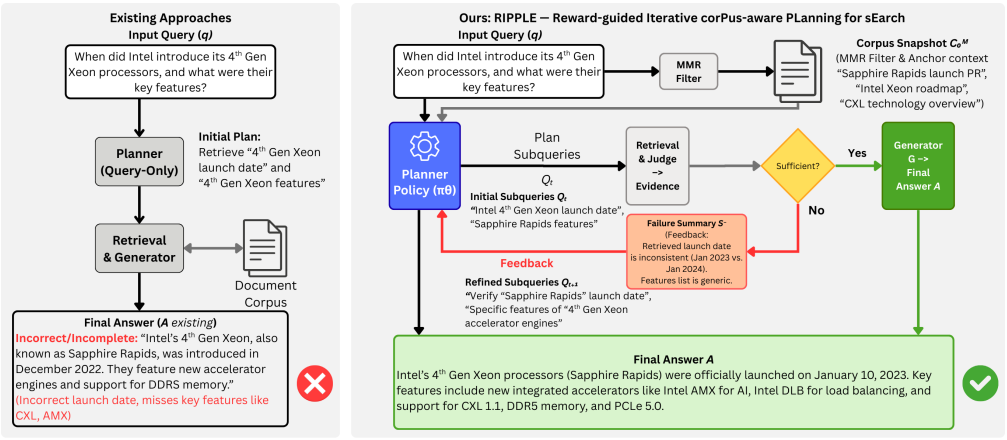


Figure 1: Comparison of static query-only planning vs. our iterative corpus-aware framework. Prior methods plan once and retrieve in a single pass, yielding incomplete answers, while RIPPLE iteratively refines subqueries using corpus feedback to obtain sufficient evidence and improve accuracy.

evidence, with limited awareness of complementary corpus regions (Zhu et al., 2025; Lee et al., 2024). Second, although the planner adapts queries across rounds, **this adaptation is driven by the LLM reading its accumulated search history, not by a trainable signal** indicating how much of the information need each round resolves. Such outcomes are assessed only after the episode, preventing the planner from learning which intermediate decisions close the evidence gap and limiting recovery from unsuccessful search rounds (Jiayang et al., 2025).

To this end, we ask: *can an iterative search policy be trained to plan each round against a broader view of corpus evidence, while also learning, round by round, how well that round resolves the information need?* This requires two capabilities. First, a planner grounded not just in the query or the evidence retrieved so far, but in a compact, diverse snapshot of the corpus that surfaces complementary aspects of the information need. Second, a learning signal computed from an analysis of each round's outcome, distinguishing rounds that meaningfully close the evidence gap from ones that leave it unresolved, rather than only after the episode ends. Together, these let the planner reason over a richer corpus view and learn from each round's outcome how to recover within the iterative process.

We propose RIPPLE (**R**eward-guided **I**terative cor**P**us-aware **PL**anning for s**E**arch), which operationalizes both capabilities within a single iterative search loop. Given a query, RIPPLE casts a wide initial retrieval net and narrows it via maximal marginal relevance (MMR) Carbonell & Goldstein (1998) into a compact, diversity-optimized snapshot preserving query relevance and complementary coverage rather than a narrow, redundant top-ranked set. This broadens the corpus evidence available to the planner from the first round and anchors planning throughout search. Conditioned on this snapshot, the query, and prior search history, the planner issues subqueries for missing evidence. An LLM-based judge evaluates and summarizes retrieved documents; when relevant evidence remains missing, these summaries are fed back into the next round, exposing previously explored but insufficient directions. Search continues until evidence suffices or the round limit is reached (Figure 1). Rather than supervising this process only at the episode's end, we train the planner with Group Relative Policy Optimization (GRPO) (Shao et al., 2024): at each round, an answer generated from the cumulative evidence is judged, and its quality directly supervises that search decision. Training across both successful and unsuccessful outcomes teaches the planner to search effectively over a diverse corpus view and recover from inadequate retrievals, coupling intermediate search behavior with downstream answer quality rather than relying on in-context adaptation alone.

Across MSQA (Yang et al., 2023), TechQA (Castelli et al., 2020), and ANTIQUE (Hashemi et al., 2020), RIPPLE outperforms strong retrieval, planning-and-reasoning, and RL-based search baselines, exceeding Search-R1 by 0.086/0.083 LLM-Eval on ANTIQUE/TechQA and 0.162 CAR on MSQA (Table 1). Ablations validate our core design choices: replacing the diverse corpus snapshot with plain top-k or query-only context costs up to 0.100 LLM-Eval and 0.121 CAR (Table 3), while replacing search-round-level, failure-aware supervision with an episode-final reward lowers both metrics across datasets (Table 2). These gains generalize across four planner backbones and an

independent evaluator (Table 6), trained HotpotQA and 2WikiMultiHopQA (Table 4), and zero-shot ANTIQUE-to-HotpotQA transfer (§A.6); the snapshot also reduces search rounds and latency relative to Search-R1 despite its construction cost (Table 5). Overall, RIPPLE's gains arise from a diverse, corpus-grounded planning view and search-round-level outcome supervision.

2 Related Work

Single-pass retrieval-augmented generation: Retrieval-augmented generation (RAG) grounds neural language models in external corpora, improving factuality and knowledge coverage over purely parametric models (Lewis et al., 2020a). Most systems follow a single-pass paradigm, providing the generator with a fixed set of documents retrieved using sparse or dense methods. Dense passage retrieval (DPR) (Karpukhin et al., 2020), late-interaction models such as ColBERT (Khattab & Zaharia, 2020), and hard-negative and contrastive training such as ANCE (Xiong et al., 2021) have substantially improved retrieval quality. Reader-side methods such as Fusion-in-Decoder (FiD) improve generation from larger evidence sets (Izacard & Grave, 2021), while hybrid sparse–dense retrieval, multi-vector indexing, and retriever–reader co-training further strengthen these pipelines (Yates et al., 2021; Sawarkar et al., 2024). Nevertheless, these approaches remain largely static, with limited adaptation to retrieval outcomes or failures.

Iterative retrieval and query decomposition for evidence coverage: Prior work relaxes single-pass retrieval through iterative search and query decomposition. Early multi-hop systems expand evidence chains (Sun et al., 2019); query decomposition surfaces complementary evidence through intermediate steps (Zhang et al., 2021); and follow-up generation and self-asking enable iterative querying (Press et al., 2023). Retrieval–generation loops condition subsequent retrieval on earlier results (Asai et al., 2024; Lyu et al., 2024), while hierarchical and program/graph-structured approaches support coarse-to-fine and controllable multi-step retrieval (Zhang et al., 2024; He et al., 2024; Yang et al., 2024). These methods establish the value of iterative evidence discovery, but many rely on prompting or heuristic control rather than learning from the downstream utility of intermediate retrieval decisions.

Learned planning and agentic retrieval control: A growing body of work frames RAG as planning over *how*, *when*, and *what* to retrieve. Agentic RAG interleaves reasoning and retrieval for dynamic subquerying and evidence aggregation (Yao et al., 2023), while planning-based prompting generates retrieval plans or sub-questions (Wang et al., 2023; Hao et al., 2023). RaDeR and ReasonIR incorporate reasoning signals into retrieval or reranking (Das et al., 2025; Shao et al., 2025), while Self-RAG uses self-reflection to control retrieval (Asai et al., 2024). Other work explores multi-agent coordination, planner–executor architectures, and tool-augmented retrieval (Chang et al., 2025; Besrour et al., 2025; Pan et al., 2025; Lumer et al., 2024). Reinforcement learning enables learned query reformulation and hierarchical search (Nogueira & Cho, 2017; Agrawal et al., 2023; Patnaik et al., 2025b), alongside learned stopping, evidence sufficiency, and failure-aware retrieval (Park et al., 2025; Li et al., 2025b). RIPPLE focuses on two complementary aspects of learned search: the planner's *state* and *supervision granularity*. It combines a persistent diversity-aware corpus snapshot and failure feedback with outcome supervision over answer quality from cumulative evidence, distinguishing it from inference-time planning and reflection methods and RL-based search policies such as Search-R1 that rely primarily on trajectory-level supervision.

3 Methodology

The central idea of RIPPLE (Figure 2) is to treat retrieval as a *corpus-aware, feedback-driven planning process*. A trainable planner conditions on (i) a compact, diversity-aware corpus snapshot and (ii) feedback from previously retrieved non-relevant documents, allowing search to adapt to corpus evidence and unsuccessful retrieval directions. This evolving state guides search, while round-level outcome supervision (Section 3.2) identifies decisions that improve downstream answer quality.

3.1 Iterative Retrieval-Augmented Generation

Retrieval-augmented generation (RAG) grounds language models in external corpora for knowledge-intensive QA, but its effectiveness depends critically on how evidence is searched and retrieved. Standard RAG typically retrieves a fixed set of documents from the original query, which can be

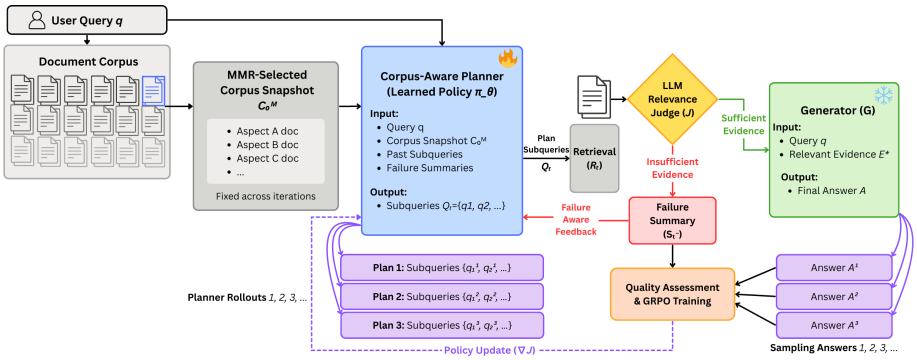


Figure 2: Overview of RIPPLE. A fixed MMR-based corpus snapshot and failure feedback guide the planner's subqueries across search rounds. Retrieved evidence is judged for relevance, with failures guiding refinement and sufficient evidence triggering generation. During training, GRPO optimizes search decisions using downstream answer quality.

insufficient for complex or multi-faceted information needs whose relevant evidence is dispersed across the corpus. Formally, let $\mathcal{D} = \{d_1, \ldots, d_N\}$ denote the document corpus and $q \in \mathcal{Q}$ a user query. The objective is to generate an answer $A \in \mathcal{A}$ grounded in evidence retrieved from $\mathcal{D}$. Given a query q' and retrieval budget L, a retriever R returns $R_L(q', \mathcal{D}) \subset \mathcal{D}$ with $|R_L(q', \mathcal{D})| = L$, which can subsequently be provided to a generator G to produce an answer.

We relax the single-pass formulation and model retrieval as a sequential decision process. At each iteration $t \in \{0, \ldots, T\}$, the system generates subqueries, retrieves and accumulates evidence, and determines whether further search is needed. Let $\mathcal{Q}_t = \{q_{t,1}, \ldots, q_{t,K}\}$ denote the subqueries at iteration t, with budget K. Each $q' \in \mathcal{Q}_t$ is executed with retrieval budget L_t, yielding the aggregated, de-duplicated set $\mathcal{R}_t = \bigcup_{q' \in \mathcal{Q}_t} R_{L_t}(q', \mathcal{D})$. The cumulative evidence is updated as $\mathcal{E}_t = \mathcal{E}_{t-1} \cup \mathcal{R}_t$, with $\mathcal{E}_{-1} = \emptyset$. Newly retrieved documents are evaluated by a relevance judge that assigns each a binary relevance label and abstractive summary, yielding

$$\mathcal{E}_t^+ = \{d \in \mathcal{E}_t \mid \text{relevance}(d) = 1\}, \quad \mathcal{E}_t^- = \mathcal{E}_t \setminus \mathcal{E}_t^+ \tag{1}$$

Search terminates early when $|\mathcal{E}_t^+| \geq \tau$, where τ is an operational relevance threshold set to $\tau = 1$ in our experiments, or when the maximum iteration budget T is reached. We use this threshold as a simple early-stopping criterion rather than as a complete measure of evidence sufficiency; the usefulness of retrieved evidence for downstream answering is instead incorporated into the planner's learning objective in Section 3.2. This iterative formulation raises two central questions: (i) how should the planner generate effective search actions at each iteration, and (ii) how should retrieval outcomes, particularly unsuccessful ones, inform subsequent search decisions.

3.2 Feedback-Guided Training of Planner

We formalize search using a unified *planner context* that captures query-relevant corpus evidence and feedback from prior retrieval attempts. Given query q, we first retrieve a large candidate set with budget L_0, $\mathcal{R}_{L_0}^{(0)} = R_{L_0}(q, \mathcal{D})$. While a large L_0 improves coverage, it can introduce redundancy; hence we apply maximal marginal relevance (MMR) to construct a compact, aspect-diverse snapshot. Let $\phi(\cdot)$ denote the retriever embedding function and $\text{sim}(\cdot, \cdot)$ cosine similarity. MMR selects a subset $\mathcal{C}_0^M \subset \mathcal{R}_{L_0}^{(0)}$ of size M by iteratively selecting

$$d^* = \arg \max_{d \in \mathcal{R}_{L_0}^{(0)} \setminus \mathcal{C}} \left[\lambda \, \text{sim}(\phi(q), \phi(d)) - (1 - \lambda) \max_{d' \in \mathcal{C}} \text{sim}(\phi(d), \phi(d')) \right], \tag{2}$$

where $\mathcal{C}$ denotes the documents selected so far and $\lambda \in [0, 1]$ balances query relevance and diversity. This yields the persistent, query-specific corpus snapshot $\mathcal{C}_0^M = \text{MMR}_M(\mathcal{R}_{L_0}^{(0)})$. At iteration t, the planner operates over context

$$\mathcal{X}_t = \begin{cases} \{q, \ \mathcal{C}_0^M, \ \mathcal{I}\}, & t = 0, \\ \{q, \ \mathcal{C}_0^M, \ \mathcal{Q}_{<t}, \ \mathcal{S}_{<t}^-, \ \mathcal{I}\}, & t > 0, \end{cases} \tag{3}$$

where $\mathcal{Q}_{<t} = \bigcup_{\tau=0}^{t-1} \mathcal{Q}_\tau$ contains previously generated subqueries, and

$$\mathcal{S}^-_{<t} = \bigcup_{\tau=0}^{t-1} \{\text{summ}(d) \mid d \in \mathcal{R}_\tau \wedge \text{relevance}(d) = 0\} \tag{4}$$

contains abstractive summaries of previously retrieved non-relevant documents. These summaries expose search directions that have already produced insufficient evidence, allowing subsequent plans to redirect retrieval rather than repeatedly explore the same evidence. The term $\mathcal{I}$ denotes high-level planner instructions, such as encouraging complementary rather than paraphrased subqueries.

Conditioned on $\mathcal{X}_t$, the planner autoregressively generates a structured output $Y_t \sim \pi_\theta(\cdot \mid \mathcal{X}_t)$, from which we parse a bounded set of subqueries $\mathcal{Q}_t = \{q_{t,1}, \ldots, q_{t,K}\}$. Generation uses a termination token and is constrained to produce at most K subqueries. Each subquery retrieves L' documents, where $L' \ll L_0$. Retrieved documents are aggregated, de-duplicated, and evaluated by the relevance judge; their relevance labels and summaries update the cumulative evidence pool and, when search continues, the planner context for the next iteration.

Planner Training with Round-Level Outcome Supervision: We train the planner policy π_θ using reinforcement learning to optimize search decisions for downstream answer quality. The iterative search process defines a sequential decision problem in which $\mathcal{X}_t$ is the state, Y_t represents the planner output encoding the search action $\mathcal{Q}_t$, and retrieval, relevance judgment, and context updates induce the transition to the next state. A complete search *trajectory* consists of successive decisions $\{(\mathcal{X}_t, \mathcal{Q}_t)\}_{t=0}^{T'}$, where $T' \leq T$ depends on early stopping.

Training only from a terminal reward after the complete trajectory provides sparse supervision for this long-horizon search problem. We instead use *round-level outcome supervision*. At state $\mathcal{X}_t$, a candidate plan $\mathcal{Q}_t$ is executed through retrieval and relevance judgment, and newly retrieved relevant evidence is added to the cumulative evidence. An intermediate answer A_t is generated from this evidence, and the plan receives utility U_t based on answer quality and auxiliary plan-level objectives. Thus, each search decision receives an outcome signal based on the evidence accumulated through that point in the trajectory, providing denser credit assignment than terminal-only supervision.

Dual-Mode State Construction. To align training with the states encountered during iterative inference, we train the planner under two complementary modes. *Initial-state training* operates on $\mathcal{X}_0$, where the planner conditions on the query and corpus snapshot to learn first-round exploration. *Failure-conditioned training* operates on later states $\mathcal{X}_t$ ($t > 0$), which additionally contain prior subqueries and summaries of non-relevant retrievals, training the planner to redirect search after unsuccessful attempts. Thus, the policy learns both how to initiate search from a corpus-grounded view and how to recover as retrieval feedback accumulates. We detail the construction and sampling of these training states in Appendix A.2.

For a candidate planner output Y_t, with parsed plan $\mathcal{Q}_t$, we define the round-level utility as

$$U_t = \lambda_{\text{ans}} U^{(t)}_{\text{ans}} + \lambda_{\text{fmt}} U^{(t)}_{\text{fmt}} + \lambda_{\text{rel}} U^{(t)}_{\text{rel}} + \lambda_{\text{div}} U^{(t)}_{\text{div}}. \tag{5}$$

The primary task-level signal is answer correctness. After executing $\mathcal{Q}_t$, the relevant evidence retrieved in the current round is combined with the relevant evidence accumulated in preceding rounds to form $\mathcal{E}^+_t$. We generate an intermediate answer $A_t = G(q, \mathcal{E}^+_t)$, and define $U^{(t)}_{\text{ans}} = J_{\text{ans}}(A_t, A^*)$, where A^* is the reference answer and $J_{\text{ans}}(\cdot)$ is an LLM-based evaluator. Thus, $U^{(t)}_{\text{ans}}$ measures downstream answer quality after incorporating the evidence available through round t, rather than rewarding retrieved documents independently of their usefulness for answering the question. Format adherence encourages the planner to follow the prescribed output structure and respect the subquery budget, $U^{(t)}_{\text{fmt}} = \mathbb{I}[\text{ValidFmt}(Y_t)] \cdot \mathbb{I}[|\mathcal{Q}_t| \leq K]$. Query relevance measures alignment between the generated subqueries and the original information need, $U^{(t)}_{\text{rel}} = \frac{1}{|\mathcal{Q}_t|} \sum_{q' \in \mathcal{Q}_t} \text{sim}(\phi(q), \phi(q'))$, while query diversity encourages complementary search directions, $U^{(t)}_{\text{div}} = \frac{2}{|\mathcal{Q}_t|(|\mathcal{Q}_t|-1)} \sum_{i<j} (1 - \text{sim}(\phi(q_{t,i}), \phi(q_{t,j})))$. Answer correctness therefore provides the primary downstream task signal, while format, relevance, and diversity serve as auxiliary shaping objectives over the planner output and resulting search plan.

We optimize the planner using Group Relative Policy Optimization (GRPO) (Shao et al., 2024). For a fixed planner state $\mathcal{X}_t$, we sample G candidate planner outputs, $\{Y^{(1)}_t, \ldots, Y^{(G)}_t\} \sim \pi_\theta(\cdot \mid \mathcal{X}_t)$,

where each $Y_t^{(i)}$ is parsed into a candidate plan $\mathcal{Q}_t^{(i)}$ containing at most K subqueries. All candidates therefore begin from the same planner state and the same evidence accumulated before round t. Executing candidate plan $\mathcal{Q}_t^{(i)}$ produces retrieved documents $\mathcal{R}_t^{(i)}$. Let $\mathcal{R}_t^{+(i)} = \left\{ d \in \mathcal{R}_t^{(i)} \mid \text{relevance}(d) = 1 \right\}$ denote the relevant documents retrieved by candidate i. Its candidate-specific cumulative evidence is $\mathcal{E}_t^{+(i)} = \mathcal{E}_{t-1}^{+} \cup \mathcal{R}_t^{+(i)}$, from which we generate $A_t^{(i)} = G(q, \mathcal{E}_t^{+(i)})$. Each candidate consequently receives utility $U_t^{(i)}$ according to Eq. 5, with its corresponding $Y_t^{(i)}$, $\mathcal{Q}_t^{(i)}$, and $A_t^{(i)}$. Because all candidates share the same planner state and previously accumulated evidence, their relative utilities measure how effectively alternative search plans advance search from the same point in the trajectory.

The group-relative advantage is computed as $\hat{A}_t^{(i)} = U_t^{(i)} - \frac{1}{G} \sum_{j=1}^{G} U_t^{(j)}$. The planner parameters are updated by maximizing the clipped GRPO objective with $\rho_t^{(i)} = \pi_\theta(Y_t^{(i)} \mid \mathcal{X}_t) / \pi_{\theta_{\text{old}}}(Y_t^{(i)} \mid \mathcal{X}_t)$ and ϵ is the clipping parameter as depicted by Eq. 6

$$\mathcal{L}_{\text{GRPO}}(\theta) = \mathbb{E}_{t,i} \left[\min \left(\rho_t^{(i)} \hat{A}_t^{(i)}, \text{clip}(\rho_t^{(i)}, 1-\epsilon, 1+\epsilon) \hat{A}_t^{(i)} \right) \right], \tag{6}$$

3.3 Inference-Time Execution

At inference time, the learned planner π_θ uses the same context construction as during training. We first retrieve $\mathcal{R}_{L_0}^{(0)} = R_{L_0}(q, \mathcal{D})$ and construct a fixed, aspect-diverse corpus snapshot $\mathcal{C}_0^M = \text{MMR}_M(\mathcal{R}_{L_0}^{(0)})$. We initialize $\mathcal{E}_{-1} = \emptyset$, $\mathcal{Q}_{<0} = \emptyset$, and $\mathcal{S}_{<0}^{-} = \emptyset$. At each iteration $t \in \{0, \ldots, T\}$, the planner context $\mathcal{X}_t$ combines the query and fixed snapshot with, for $t > 0$, previous subqueries and summaries of non-relevant retrievals. The planner generates $Y_t \sim \pi_\theta(\cdot \mid \mathcal{X}_t)$, parsed into subqueries $\mathcal{Q}_t = \{q_{t,1}, \ldots, q_{t,K}\}$, which retrieve $\mathcal{R}_t = \bigcup_{q' \in \mathcal{Q}_t} R_{L'}(q', \mathcal{D})$. After de-duplication and relevance judgment, evidence is accumulated as $\mathcal{E}_t = \mathcal{E}_{t-1} \cup \mathcal{R}_t$, with $\mathcal{E}_t^{+} = \{d \in \mathcal{E}_t \mid \text{relevance}(d) = 1\}$. Search terminates when $|\mathcal{E}_t^{+}| \geq \tau$ or the iteration budget is exhausted; otherwise, the generated subqueries and summaries of newly retrieved non-relevant documents update the next planner context. At the terminating iteration t^*, the final answer is generated as $A = G(q, \mathcal{E}_{t^*}^{+})$.

4 Experimental Protocol

Datasets and Evaluation: We evaluate RIPPLE on three knowledge-intensive RAG benchmarks: ANTIQUE, MSQA, and TechQA. ANTIQUE contains real, expert-annotated non-factoid queries spanning complex, ambiguous, and multi-aspect information needs (Hashemi et al., 2020), with 2,426 training and 200 test queries. MSQA requires evidence aggregation across heterogeneous sources and domains (Yang et al., 2023), with 9,517 training and 572 test samples. TechQA contains domain-specific procedural questions from technical forums and documentation requiring specialized long-form evidence (Castelli et al., 2020), with 600 training and 310 test samples. We evaluate answer quality using ROUGE-L (Lin, 2004), BERTScore (F1) (Zhang* et al., 2020), and LLM-based evaluation following Plan-and-Refine (Salemi et al., 2025), and evidence sufficiency using Can-Answer Rate (CAR) (Yang et al., 2023). Appendix A.3 defines the metrics and Appendix A.4 reports all prompts. GPT-4o serves as the primary LLM judge; we additionally report evaluation with an independent judge in Table 6 to control for dependence on models used elsewhere in the pipeline.

Baselines: We compare RIPPLE against four categories of baselines: (i) *Vanilla RAG* (Lewis et al., 2020a), which performs single-shot retrieval without iterative adaptation; (ii) *retrieval-centric methods*, including BM25 (Robertson et al., 1993), DIVER (Long et al., 2025), and ReasonIR (Shao et al., 2025); (iii) *planning and reasoning-based approaches*, including Chain-of-Thought-RAG (CoT-RAG) (Li et al., 2025a), Plan-and-Refine (Salemi et al., 2025), ReAct (Yao et al., 2023), Self-RAG (Asai et al., 2024), and RaDeR (Das et al., 2025); and (iv) *RL-based and iterative retrieval-generation methods*, including R3-RAG (Li et al., 2025b), RAG-R1 (Tan et al., 2026), DeepRetrieval (Jiang et al., 2025a), and Search-R1 (Jin et al., 2025). To ensure controlled comparison, every trainable baseline is retrained separately on each benchmark using its original training recipe and hyperparameters. Across trainable methods, we use the same base LLM, retrieval corpus and chunking, dense retriever, and answer generator wherever applicable; prompt-only methods are evaluated without additional training. Table 1 reports the resulting comparison.

Table 1: Performance comparison of RIPPLE against retrieval-based, planning-based, and iterative RAG baselines on **ANTIQUE**, **TechQA**, and **MSQA**. We report ROUGE-L (R-L), BERTScore F1 (BERT), LLM-based evaluation (LLM), and Can-Answer Rate (CAR); all metrics are fractions in $[0, 1]$ and higher is better. **Bold** marks the best value per column and underline the strongest baseline. RIPPLE is run with $t = 5$ and outperforms every baseline on all 12 benchmark–metric pairs.

	ANTIQUE				TechQA				MSQA			
Method	**R-L**	**BERT**	**LLM**	**CAR**	**R-L**	**BERT**	**LLM**	**CAR**	**R-L**	**BERT**	**LLM**	**CAR**
Vanilla RAG (Lewis et al., 2020a)	0.172	0.759	0.272	0.655	0.157	0.737	0.190	0.437	0.060	0.678	0.034	0.084
Retrieval-centric Approaches												
BM25 (Robertson et al., 1993)	0.159	0.748	0.245	0.610	0.153	0.749	0.241	0.601	0.096	0.700	0.077	0.250
DIVER (Long et al., 2025)	0.165	0.753	0.259	0.632	0.159	0.752	0.290	0.616	0.124	0.724	0.114	0.501
ReasonIR (Shao et al., 2025)	0.169	0.756	0.266	0.645	0.161	0.753	0.304	0.628	0.125	0.724	0.132	0.441
Planning & Reasoning-based Approaches												
CoT-RAG (Li et al., 2025a)	0.136	0.735	0.115	0.345	0.085	0.695	0.037	0.186	0.136	0.735	0.089	0.353
Plan-and-Refine (Salemi et al., 2025)	0.113	0.727	0.032	0.220	0.111	0.727	0.023	0.218	0.071	0.683	0.030	0.103
ReAct (Yao et al., 2023)	0.146	0.749	0.178	0.425	0.144	0.746	0.151	0.399	0.114	0.719	0.039	0.189
Self-RAG (Asai et al., 2024)	0.141	0.744	0.158	0.395	0.148	0.749	0.169	0.422	0.103	0.708	0.084	0.310
RaDeR (Das et al., 2025)	0.173	0.760	0.320	0.640	0.140	0.742	0.143	0.375	0.085	0.695	0.049	0.214
RL-based & Iterative Retrieval-Generation Methods												
R3-RAG (Li et al., 2025b)	0.176	0.762	0.308	0.668	0.165	0.755	0.312	0.641	0.132	0.731	0.148	0.472
RAG-R1 (Tan et al., 2026)	0.178	0.764	0.315	0.681	0.168	0.758	0.321	0.655	0.136	0.735	0.156	0.498
DeepRetrieval (Jiang et al., 2025a)	0.174	0.759	0.301	0.652	0.166	0.754	0.298	0.636	0.139	0.737	0.161	0.505
Search-R1 (Jin et al., 2025)	0.181	0.768	0.332	0.695	0.171	0.761	0.336	0.668	0.142	0.741	0.174	0.528
RIPPLE $(t = 5)$	**0.195**	**0.781**	**0.418**	**0.762**	**0.195**	**0.785**	**0.419**	**0.769**	**0.159**	**0.790**	**0.233**	**0.690**

Implementation Details: We use LLAMA-3.1-8B-INSTRUCT as the planner model π_θ and GPT-4O as the generator G, relevance judge J, and primary LLM-based evaluator. Dense retrieval uses Snowflake-Large-v2, with an initial retrieval budget $L_0 = 50$ followed by MMR compression to a fixed snapshot of size $M = 10$ with $\lambda = 0.4$. We adapt the chunking strategy of Salemi et al. (2025) with a chunk size of 512. For each planner-generated subquery, we retrieve $L' = 5$ documents, with a subquery budget $K = 5$, maximum iteration budget $T = 5$, and operational early-stopping threshold $\tau = 1$. The planner is trained using GRPO (Shao et al., 2024) through the `verl` framework (Sheng et al., 2025), sampling $G = 4$ candidate planner outputs per state for 2 epochs. We use AdamW with learning rate 2×10^{-5}, `bfloat16` precision, per-device batch size 2, gradient accumulation of 4, and 8 A100 (80GB) GPUs, with DeepSpeed ZeRO-3 and gradient checkpointing. Reward weights are $\lambda_{\text{ans}} = 0.5$, $\lambda_{\text{fmt}} = 0.1$, $\lambda_{\text{rel}} = 0.15$, and $\lambda_{\text{div}} = 0.15$. During inference, planner decoding uses sampling-based beam search with 3 beams. All hyperparameters are held fixed across datasets; we report sensitivity to the MMR trade-off λ in Appendix A.8, to L_0 and M in Appendix A.12, to K in Appendix A.13, to T in Appendix A.7, and to G in Appendix A.11.

5 RESULTS AND DISCUSSION

Comparison of RIPPLE with Baselines: Table 1 compares RIPPLE with retrieval, planning, and recent RL-based RAG methods, including Search-R1, RAG-R1, DeepRetrieval, and R3-RAG. RIPPLE consistently outperforms all baselines across ANTIQUE, TechQA, and MSQA on all metrics. On ANTIQUE, the strongest baseline, Search-R1 (Jin et al., 2025), achieves 0.332 LLM and 0.695 CAR, while RIPPLE improves these to 0.418 and 0.762. On TechQA, RIPPLE similarly improves over Search-R1 from 0.336 to 0.419 LLM and 0.668 to 0.769 CAR, with additional gains in ROUGE-L and BERTScore. On MSQA, it achieves 0.233 LLM and 0.690 CAR compared with 0.174 and 0.528 for Search-R1. These results show consistent gains in retrieval relevance and answer correctness from corpus-aware, feedback-driven planning over strong learned-search baselines.

Qualitative Analysis: We present a TechQA case study tracing the search loop in Figure 2 and highlighting corpus-aware planning in complex troubleshooting. Consider the query: *"Why does a distributed Spark job intermittently fail with out-of-memory errors despite sufficient executor memory being configured?"* This underspecified query admits multiple failure modes spanning configuration, runtime behavior, and data skew. Self-RAG, relying on retrieval conditioned on the original query and generation-time reflection, retrieves high-level documents recommending increased executor or

heap memory; these fail to address intermittency or surface diagnostic evidence. In contrast, RIPPLE refines retrieval using corpus-grounded feedback. At the first iteration (t=1), the planner issues subqueries such as *"Spark out of memory error executor memory"* and *"Spark OOM despite sufficient memory"*, retrieving documents emphasizing naïve memory scaling that are judged non-relevant. Summaries of these failures, together with corpus-level hints (e.g., mentions of non-heap memory and workload-dependent execution), are incorporated into planner memory. At the second iteration (t=2), conditioned on these signals, the planner explores alternatives via *"Spark intermittent OOM data skew"* and *"Spark executor OOM shuffle memory"*. These surface factors such as skewed partitions and shuffle spills, but the evidence remains insufficient. At the third iteration (t=3), leveraging accumulated feedback, the planner generates targeted subqueries including *"Spark OOM caused by broadcast joins"* and *"Spark off-heap memory shuffle OOM"*, retrieving documents explaining workload-dependent execution paths that trigger executor OOMs despite adequate heap memory. This evidence satisfies the relevance criterion and early stopping threshold τ=1, after which the generator produces a grounded explanation. Thus, unlike Self-RAG, which lacks corpus-level feedback in subquery formulation, RIPPLE progressively eliminates uninformative search directions and converges on diagnostic evidence through failure-aware planning.

Reward design and evidence utility: The upper block of Table 2 analyzes the planner's reward components. Removing diversity or query relevance degrades answer quality, with ANTIQUE LLM Eval dropping from 0.418 to 0.402 and 0.393, respectively, highlighting their role in complementary, query-aligned search. TechQA CAR is a minor exception, increasing slightly without relevance or format rewards (0.777/0.779 vs. 0.769), despite lower answer quality. Removing both relevance and diversity causes the largest degradation, reducing LLM Eval to 0.374/0.375 and CAR to 0.612/0.694 on ANTIQUE/TechQA. Format adherence enforces the output structure and subquery budget. Answer correctness remains the primary signal. We additionally investigate whether directly rewarding evidence coverage provides complementary supervision beyond answer-level correctness; results and reward construction are provided in Appendix A.10.

Table 2: Planner reward design (upper) and supervision granularity (lower). Removing reward components or round-level supervision affects LLM evaluation (LLM) and Can-Answer Rate (CAR) performance of RIPPLE.

Setting	ANTIQUE		TechQA		MSQA	
	LLM	CAR	LLM	CAR	LLM	CAR
Reward Design						
All rewards	0.418	0.762	0.419	0.769	0.233	0.690
– similarity diversity	0.402	0.664	0.405	0.749	0.222	0.663
– query relevance	0.393	0.642	0.405	0.777	0.214	0.642
– format adherence	0.399	0.653	0.417	0.779	0.219	0.653
– diversity & relevance	0.374	0.612	0.375	0.694	0.202	0.612
+ explicit evidence coverage	**0.427**	**0.778**	**0.431**	**0.786**	**0.245**	**0.712**
Supervision Granularity						
Terminal-only	0.389	0.724	0.391	0.731	0.211	0.648
Round-level	**0.418**	**0.762**	**0.419**	**0.769**	**0.233**	**0.690**

Effect of round-level outcome supervision: The lower block of Table 2 isolates supervision granularity. For the terminal-only variant, we remove the second training mode that supervises intermediate, failure-conditioned states and train only from the final trajectory outcome. This consistently reduces performance across all three datasets; on ANTIQUE, LLM Eval drops from 0.418 to 0.389 and CAR from 0.762 to 0.724, with similar reductions on TechQA and MSQA. These results show that supervising intermediate search states provides denser credit for useful decisions than terminal outcomes alone.

Table 3: Effect of planner context design. A diversity-aware MMR snapshot and failure summaries consistently improve LLM evaluation (LLM) and Can-Answer Rate (CAR).

Setting	ANTIQUE		TechQA		MSQA	
	LLM	CAR	LLM	CAR	LLM	CAR
Corpus Snapshot						
No snapshot (query only)	0.318	0.641	0.342	0.678	0.171	0.582
Plain top-k (no diversity)	0.371	0.701	0.400	0.739	0.219	0.654
MMR snapshot	**0.418**	**0.762**	**0.419**	**0.769**	**0.233**	**0.690**
Failure Feedback						
No summaries	0.378	0.708	0.395	0.729	0.214	0.642
With summaries	**0.418**	**0.762**	**0.419**	**0.769**	**0.233**	**0.690**

Effect of corpus snapshot and failure feedback: Table 3 evaluates the two sources of corpus-derived planner context. Corpus conditioning substantially improves performance: on ANTIQUE, LLM Eval increases from 0.318 with no snapshot to 0.371 with plain top-k context and 0.418 with the diversity-optimized MMR snapshot, while CAR improves from 0.641 to 0.701 and 0.762, respectively. Failure feedback provides complementary gains: incorporating summaries of non-relevant retrievals improves LLM Eval from 0.378 to 0.418 and CAR from 0.708 to 0.762, with similar trends across datasets. These results show that planning benefits from both a diverse, persistent view of corpus evidence and explicit feedback about previously unsuccessful search directions.

Table 6: Robustness across planner backbones (left) and an independent LLM evaluator (right). RIPPLE consistently improves LLM evaluation (LLM) and Can-Answer Rate (CAR) across planner families, with gains persisting when final evaluator is different from judge LLM used during training.

Planner	Method	ANTIQUE		TechQA		MSQA	
		LLM	CAR	LLM	CAR	LLM	CAR
DeepSeek-7B	Base	0.305	0.609	0.180	0.443	0.110	0.306
	RIPPLE	**0.382**	**0.713**	**0.357**	**0.682**	**0.222**	**0.668**
Mistral-7B	Base	0.332	0.655	0.198	0.475	0.127	0.354
	RIPPLE	**0.391**	**0.726**	**0.381**	**0.713**	**0.227**	**0.676**
Qwen-7B	Base	0.357	0.691	0.214	0.498	0.138	0.385
	RIPPLE	**0.425**	**0.771**	**0.402**	**0.743**	**0.234**	**0.690**
Llama-3.1-8B	Base	0.332	0.695	0.336	0.668	0.174	0.528
	RIPPLE	**0.418**	**0.762**	**0.419**	**0.769**	**0.233**	**0.690**

Method	ANTIQUE		TechQA		MSQA	
	LLM	CAR	LLM	CAR	LLM	CAR
Vanilla RAG	0.261	0.642	0.183	0.429	0.031	0.079
RAG-R1	0.308	0.675	0.318	0.649	0.151	0.491
Search-R1	0.338	0.701	0.342	0.675	0.179	0.535
RIPPLE	**0.422**	**0.758**	**0.415**	**0.773**	**0.238**	**0.696**

Independent-judge evaluation using Claude Sonnet 4.6; answer generator remains GPT-4o.

Performance on standard multi-hop benchmarks: Table 4 evaluates RIPPLE, Search-R1, and RAG-R1 after benchmark-specific training on HotpotQA and 2WikiMultiHopQA. RIPPLE outperforms both learned search baselines, reaching $0.498/0.813$ LLM Eval/CAR on HotpotQA and $0.478/0.798$ on 2WikiMultiHopQA, extending its gains to standard multi-hop settings.

Table 4: Performance on standard multi-hop QA benchmarks. RIPPLE improves LLM evaluation (LLM) and Can-Answer Rate (CAR) over baselines after benchmark-specific training.

Benchmark	**Search-R1**		**RAG-R1**		**RIPPLE**	
	LLM	CAR	LLM	CAR	LLM	CAR
HotpotQA	0.452	0.788	0.438	0.774	**0.498**	**0.813**
2WikiMultiHopQA	0.432	0.768	0.418	0.752	**0.478**	**0.798**

End-to-end efficiency: Table 5 analyzes end-to-end search cost. Although snapshot construction adds 0.73s per query, it reduces average search iterations from 3.55 to 2.81, yielding 11.58s total latency versus 12.28s without the snapshot and 12.08s for Search-R1. Thus, its upfront cost is offset by fewer search steps.

Robustness across planner models and evaluators: Table 6 evaluates robustness to planner backbone and LLM evaluator. Across DeepSeek-7B, Mistral-7B, Qwen-7B, and Llama-3.1-8B, RIPPLE consistently improves LLM Eval and CAR over the corresponding policy of Search-R1; with Qwen-7B, for example, ANTIQUE improves from 0.357 to 0.425 LLM Eval and 0.691 to 0.771 CAR. Using Claude Sonnet 4.6 as an independent judge while retaining GPT-4o as the generator, RIPPLE outperforms Vanilla RAG, RAG-R1, and Search-R1 across all datasets, indicating that gains persist when evaluation is separated from answer generation and training-time reward estimation. Generator robustness across GPT-5.1, Claude Sonnet/Opus 4.6, and Gemini 2.5 Pro is reported in A.5.

Table 5: End-to-end planner inference latency. Snapshot construction adds upfront cost but reduces search iterations and total latency relative to Search-R1 and RIPPLE without the snapshot.

Method	**Snapshot**	**Snapshot Cost**	**Total Latency**	**Avg. Iter.**
Search-R1	No	–	12.08s	3.80
RIPPLE w/o snapshot	No	–	12.28s	3.55
RIPPLE	Yes	0.73s	**11.58s**	**2.81**

Appendix contains information on limitations (A.1), dual-mode training states (A.2), evaluation (A.3), prompts (A.4), generator robustness (A.5), OOD HotpotQA transfer (A.6), inference iterations (A.7), MMR sensitivity (A.8), retriever robustness, additional claim-based evidence reward (A.10) (A.9), GRPO candidate plans (A.11), snapshot design (A.12), and subquery budget (A.13). We further analyze Search-R1 with RIPPLE context without retraining (A.14), corpus-size scalability (A.15), and planner training dynamics (A.16).

6 Conclusion

We presented RIPPLE, which couples a diversity-aware, corpus-grounded planner state and failure feedback with outcome supervision over intermediate search decisions. This enables the planner to learn which decisions close evidence gaps and recover when retrieval falls short, rather than relying on terminal outcomes alone. Across benchmarks, planner backbones, and evaluators, RIPPLE consistently outperforms strong retrieval and learned-search baselines, demonstrating the value of jointly grounding and supervising search over long horizons. Current limitations include reliance on an LLM relevance judge, a fixed stopping threshold, and additional search cost, motivating adaptive evidence sufficiency and more efficient planning.

AI USE STATEMENT

Generative AI tools were used solely for editorial assistance during manuscript preparation. Specifically, an LLM-based assistant was used to improve readability and concision, revise table and figure captions, and perform editorial consistency checks, including cross-references, citation keys, and agreement between values reported in the text and those shown in tables and figures. Generative AI was not used to develop the research ideas, theoretical formulation, methodology, experimental design, implementation, datasets, hypotheses, or scientific interpretation of the results. All research decisions, experiments, analyses, and conclusions were produced by the authors. The authors reviewed and verified all AI-assisted edits against the underlying experimental results and cited sources and take full responsibility for the final manuscript.

REPRODUCIBILITY STATEMENT

We provide the information required to reproduce our results in the main paper, appendix, and anonymized code repository. Section 3 specifies the complete method, including corpus snapshot construction, planner context and training-state construction, round-level utility (Eq. 5), GRPO optimization (Eq. 6), and inference procedure. Section 4 reports dataset splits, evaluation metrics, model backbones, training and inference hyperparameters, hardware, and the protocol used to retrain trainable baselines. Appendix A.3 defines the evaluation metrics, while Appendix A.4 provides the prompts used by the planner, relevance judge, answer generator, and evaluators. Additional analyses report sensitivity to inference iterations, MMR configuration, retriever backbone, GRPO candidate plans, corpus-snapshot design, and subquery budget (Appendices A.7–A.13), as well as robustness across planner and generator backbones (Table 6; Appendix A.5) and training dynamics (Appendix A.16). Limitations relevant to reproducibility are discussed in Appendix A.1. Our anonymized repository provides the implementation, training and evaluation scripts, and configuration files used in the experiments: `https://anonymous.4open.science/r/RIPPLE-24A8/README.md`.

REFERENCES

Sanjay Agrawal, Srujana Merugu, and Vivek Sembium. Enhancing e-commerce product search through reinforcement learning-powered query reformulation. In *Proceedings of the 32nd ACM International Conference on Information and Knowledge Management*, CIKM '23, pp. 4488–4494, New York, NY, USA, 2023. Association for Computing Machinery. ISBN 9798400701245. doi: 10.1145/3583780.3615474. URL https://doi.org/10.1145/3583780.3615474.

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. URL https://openreview.net/forum?id=hSyW5go0v8.

Catarina G Belém, Pouya Pezeshkpour, Hayate Iso, Seiji Maekawa, Nikita Bhutani, and Estevam Hruschka. From single to multi: How LLMs hallucinate in multi-document summarization. In Luis Chiruzzo, Alan Ritter, and Lu Wang (eds.), *Findings of the Association for Computational Linguistics: NAACL 2025*, pp. 5276–5309, Albuquerque, New Mexico, April 2025. Association for Computational Linguistics. ISBN 979-8-89176-195-7. doi: 10.18653/v1/2025.findings-naacl.293. URL https://aclanthology.org/2025.findings-naacl.293/.

Ines Besrour, Jingbo He, Tobias Schreieder, and Michael Färber. RAGentA: Multi-Agent Retrieval-Augmented Generation for Attributed Question Answering, 2025. URL https://arxiv.org/abs/2506.16988. Presented at the SIGIR 2025 LiveRAG Challenge, held in conjunction with the 48th International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR 2025), July 13–17, 2025, Padua, Italy.

Jaime Carbonell and Jade Goldstein. The use of mmr, diversity-based reranking for reordering documents and producing summaries. In *Proceedings of the 21st annual international ACM SIGIR conference on Research and development in information retrieval*, pp. 335–336, 1998.

Vittorio Castelli, Rishav Chakravarti, Saswati Dana, Anthony Ferritto, Radu Florian, Martin Franz, Dinesh Garg, Dinesh Khandelwal, Scott McCarley, Michael McCawley, Mohamed Nasr,

Lin Pan, Cezar Pendus, John Pitrelli, Saurabh Pujar, Salim Roukos, Andrzej Sakrajda, Avi Sil, Rosario Uceda-Sosa, Todd Ward, and Rong Zhang. The TechQA dataset. In Dan Jurafsky, Joyce Chai, Natalie Schluter, and Joel Tetreault (eds.), *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pp. 1269–1278, Online, July 2020. Association for Computational Linguistics. doi: 10.18653/v1/2020.acl-main.117. URL https://aclanthology.org/2020.acl-main.117/.

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 (eds.), *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 2607–2622, Vienna, Austria, July 2025. Association for Computational Linguistics. ISBN 979-8-89176-251-0. doi: 10.18653/v1/2025.acl-long.131. URL https://aclanthology.org/2025.acl-long.131/.

Debrup Das, Sam O'Nuallain, and Razieh Rahimi. RaDeR: Reasoning-aware dense retrieval models. In Christos Christodoulopoulos, Tanmoy Chakraborty, Carolyn Rose, and Violet Peng (eds.), *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pp. 19970–19997, Suzhou, China, November 2025. Association for Computational Linguistics. ISBN 979-8-89176-332-6. doi: 10.18653/v1/2025.emnlp-main.1011. URL https://aclanthology.org/2025.emnlp-main.1011/.

Jinyuan Fang, Zaiqiao Meng, and Craig MacDonald. TRACE the evidence: Constructing knowledge-grounded reasoning chains for retrieval-augmented generation. In Yaser Al-Onaizan, Mohit Bansal, and Yun-Nung Chen (eds.), *Findings of the Association for Computational Linguistics: EMNLP 2024*, pp. 8472–8494, Miami, Florida, USA, November 2024. Association for Computational Linguistics. doi: 10.18653/v1/2024.findings-emnlp.496. URL https://aclanthology.org/2024.findings-emnlp.496/.

Zhibin Gou, Zhihong Shao, Yeyun Gong, yelong shen, Yujiu Yang, Nan Duan, and Weizhu Chen. CRITIC: Large language models can self-correct with tool-interactive critiquing. In *The Twelfth International Conference on Learning Representations*, 2024. URL https://openreview.net/forum?id=Sx038qxjek.

Shibo Hao, Yi Gu, Haodi Ma, Joshua Hong, Zhen Wang, Daisy Wang, and Zhiting Hu. Reasoning with language model is planning with world model. In Houda Bouamor, Juan Pino, and Kalika Bali (eds.), *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pp. 8154–8173, Singapore, December 2023. Association for Computational Linguistics. doi: 10.18653/v1/2023.emnlp-main.507. URL https://aclanthology.org/2023.emnlp-main.507/.

Helia Hashemi, Mohammad Aliannejadi, Hamed Zamani, and W. Bruce Croft. Antique: A non-factoid question answering benchmark. In *Advances in Information Retrieval: 42nd European Conference on IR Research, ECIR 2020, Lisbon, Portugal, April 14–17, 2020, Proceedings, Part II*, pp. 166–173, Berlin, Heidelberg, 2020. Springer-Verlag. ISBN 978-3-030-45441-8. doi: 10.1007/978-3-030-45442-5_21. URL https://doi.org/10.1007/978-3-030-45442-5_21.

Xiaoxin He, Yijun Tian, Yifei Sun, Nitesh V Chawla, Thomas Laurent, Yann LeCun, Xavier Bresson, and Bryan Hooi. G-retriever: Retrieval-augmented generation for textual graph understanding and question answering. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*, 2024. URL https://openreview.net/forum?id=MPJ3oXtTZl.

Marti A. Hearst. *Models of the Information Seeking Process*, pp. 64–90. Cambridge University Press, 2009.

Lei Huang, Weijiang Yu, Weitao Ma, Weihong Zhong, Zhangyin Feng, Haotian Wang, Qianglong Chen, Weihua Peng, Xiaocheng Feng, Bing Qin, and Ting Liu. A survey on hallucination in large language models: Principles, taxonomy, challenges, and open questions. *ACM Trans. Inf. Syst.*, 43 (2), January 2025a. ISSN 1046-8188. doi: 10.1145/3703155. URL https://doi.org/10.1145/3703155.

Yanwen Huang, Yong Zhang, Ning Cheng, Zhitao Li, Shaojun Wang, and Jing Xiao. Dynamic attention-guided context decoding for mitigating context faithfulness hallucinations in large language models. In Wanxiang Che, Joyce Nabende, Ekaterina Shutova, and Mohammad Taher Pilehvar (eds.), *Findings of the Association for Computational Linguistics: ACL 2025*, pp. 5174–5193, Vienna, Austria, July 2025b. Association for Computational Linguistics. ISBN 979-8-89176-256-5. doi: 10.18653/v1/2025.findings-acl.269. URL https://aclanthology.org/2025.findings-acl.269/.

Gautier Izacard and Edouard Grave. Leveraging passage retrieval with generative models for open domain question answering. In Paola Merlo, Jorg Tiedemann, and Reut Tsarfaty (eds.), *Proceedings of the 16th Conference of the European Chapter of the Association for Computational Linguistics: Main Volume*, pp. 874–880, Online, April 2021. Association for Computational Linguistics. doi: 10.18653/v1/2021.eacl-main.74. URL https://aclanthology.org/2021.eacl-main.74/.

Pengcheng Jiang, Jiacheng Lin, Lang Cao, Runchu Tian, SeongKu Kang, Zifeng Wang, Jimeng Sun, and Jiawei Han. Deepretrieval: Hacking real search engines and retrievers with large language models via reinforcement learning. In *Second Conference on Language Modeling*, 2025a. URL https://openreview.net/forum?id=u9JXu4L17I.

Zhouyu Jiang, Mengshu Sun, Lei Liang, and Zhiqiang Zhang. Retrieve, summarize, plan: Advancing multi-hop question answering with an iterative approach. In *Companion Proceedings of the ACM on Web Conference 2025*, WWW '25, pp. 1677–1686, New York, NY, USA, 2025b. Association for Computing Machinery. ISBN 9798400713316. doi: 10.1145/3701716.3716889. URL https://doi.org/10.1145/3701716.3716889.

Cheng Jiayang, Qianqian Zhuang, Haoran Li, Chunkit Chan, Xin Liu, Lin Qiu, and Yangqiu Song. InteGround: On the evaluation of verification and retrieval planning in integrative grounding. In Christos Christodoulopoulos, Tanmoy Chakraborty, Carolyn Rose, and Violet Peng (eds.), *Findings of the Association for Computational Linguistics: EMNLP 2025*, pp. 13587–13602, Suzhou, China, November 2025. Association for Computational Linguistics. ISBN 979-8-89176-335-7. doi: 10.18653/v1/2025.findings-emnlp.732. URL https://aclanthology.org/2025.findings-emnlp.732/.

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 *Second Conference on Language Modeling*, 2025. URL https://openreview.net/forum?id=Rwhi91ideu.

Vladimir Karpukhin, Barlas Oguz, Sewon Min, Patrick Lewis, Ledell Wu, Sergey Edunov, Danqi Chen, and Wen-tau Yih. Dense passage retrieval for open-domain question answering. In Bonnie Webber, Trevor Cohn, Yulan He, and Yang Liu (eds.), *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pp. 6769–6781, Online, November 2020. Association for Computational Linguistics. doi: 10.18653/v1/2020.emnlp-main.550. URL https://aclanthology.org/2020.emnlp-main.550/.

Uri Katz, Mosh Levy, and Yoav Goldberg. Knowledge navigator: LLM-guided browsing framework for exploratory search in scientific literature. In Yaser Al-Onaizan, Mohit Bansal, and Yun-Nung Chen (eds.), *Findings of the Association for Computational Linguistics: EMNLP 2024*, pp. 8838–8855, Miami, Florida, USA, November 2024. Association for Computational Linguistics. doi: 10.18653/v1/2024.findings-emnlp.516. URL https://aclanthology.org/2024.findings-emnlp.516/.

Omar Khattab and Matei Zaharia. Colbert: Efficient and effective passage search via contextualized late interaction over bert. In *Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '20, pp. 39–48, New York, NY, USA, 2020. Association for Computing Machinery. ISBN 9781450380164. doi: 10.1145/3397271.3401075. URL https://doi.org/10.1145/3397271.3401075.

Myeonghwa Lee, Seonho An, and Min-Soo Kim. PlanRAG: A plan-then-retrieval augmented generation for generative large language models as decision makers. In Kevin Duh, Helena Gomez, and Steven Bethard (eds.), *Proceedings of the 2024 Conference of the North American Chapter*

of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers), pp. 6537–6555, Mexico City, Mexico, June 2024. Association for Computational Linguistics. doi: 10.18653/v1/2024.naacl-long.364. URL https://aclanthology.org/2024.naacl-long.364/.

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 *Proceedings of the 34th International Conference on Neural Information Processing Systems*, NIPS '20, Red Hook, NY, USA, 2020a. Curran Associates Inc. ISBN 9781713829546.

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 H. Larochelle, M. Ranzato, R. Hadsell, M.F. Balcan, and H. Lin (eds.), *Advances in Neural Information Processing Systems*, volume 33, pp. 9459–9474. Curran Associates, Inc., 2020b. URL https://proceedings.neurips.cc/paper_files/paper/2020/file/6b493230205f780e1bc26945df7481e5-Paper.pdf.

Feiyang Li, Peng Fang, Zhan Shi, Arijit Khan, Fang Wang, Weihao Wang, Zhangxin-hw, and Cui Yongjian. CoT-RAG: Integrating chain of thought and retrieval-augmented generation to enhance reasoning in large language models. In Christos Christodoulopoulos, Tanmoy Chakraborty, Carolyn Rose, and Violet Peng (eds.), *Findings of the Association for Computational Linguistics: EMNLP 2025*, pp. 3119–3171, Suzhou, China, November 2025a. Association for Computational Linguistics. ISBN 979-8-89176-335-7. doi: 10.18653/v1/2025.findings-emnlp.168. URL https://aclanthology.org/2025.findings-emnlp.168/.

Yuan Li, Qi Luo, Xiaonan Li, Bufan Li, Qinyuan Cheng, Bo Wang, Yining Zheng, Yuxin Wang, Zhangyue Yin, and Xipeng Qiu. R3-RAG: Learning step-by-step reasoning and retrieval for LLMs via reinforcement learning. In Christos Christodoulopoulos, Tanmoy Chakraborty, Carolyn Rose, and Violet Peng (eds.), *Findings of the Association for Computational Linguistics: EMNLP 2025*, pp. 10491–10507, Suzhou, China, November 2025b. Association for Computational Linguistics. ISBN 979-8-89176-335-7. doi: 10.18653/v1/2025.findings-emnlp.554. URL https://aclanthology.org/2025.findings-emnlp.554/.

Chin-Yew Lin. ROUGE: A package for automatic evaluation of summaries. In *Text Summarization Branches Out*, pp. 74–81, Barcelona, Spain, July 2004. Association for Computational Linguistics. URL https://aclanthology.org/W04-1013/.

Yaxi Liu, Chunxiu Qin, Xubu Ma, Fan Li, and Yulong Wang. Comparison of information search behavior for different exploratory tasks: Evidence from experiments in online knowledge communities. *Inf. Process. Manage.*, 61(5), September 2024. ISSN 0306-4573. doi: 10.1016/j.ipm.2024.103794. URL https://doi.org/10.1016/j.ipm.2024.103794.

Meixiu Long, Duolin Sun, Dan Yang, Junjie Wang, Yecheng Luo, Yue Shen, Jian Wang, Hualei Zhou, Chunxiao Guo, Peng Wei, Jiahai Wang, and Jinjie Gu. Diver: A multi-stage approach for reasoning-intensive information retrieval, 2025. URL https://arxiv.org/abs/2508.07995.

Elias Lumer, Vamse Kumar Subbiah, James A. Burke, Pradeep Honaganahalli Basavaraju, and Austin Huber. Toolshed: Scale tool-equipped agents with advanced rag-tool fusion and tool knowledge bases. *ArXiv*, abs/2410.14594, 2024. URL https://api.semanticscholar.org/CorpusID:273507442.

Yuanjie Lyu, Zihan Niu, Zheyong Xie, Chao Zhang, Tong Xu, Yang Wang, and Enhong Chen. Retrieve-plan-generation: An iterative planning and answering framework for knowledge-intensive LLM generation. In Yaser Al-Onaizan, Mohit Bansal, and Yun-Nung Chen (eds.), *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pp. 4683–4702, Miami, Florida, USA, November 2024. Association for Computational Linguistics. doi: 10.18653/v1/2024.emnlp-main.270. URL https://aclanthology.org/2024.emnlp-main.270/.

Fengran Mo, Kelong Mao, Yutao Zhu, Yihong Wu, Kaiyu Huang, and Jian-Yun Nie. ConvGQR: Generative query reformulation for conversational search. In Anna Rogers, Jordan Boyd-Graber,

and Naoaki Okazaki (eds.), *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 4998–5012, Toronto, Canada, July 2023. Association for Computational Linguistics. doi: 10.18653/v1/2023.acl-long.274. URL https://aclanthology.org/2023.acl-long.274/.

Rodrigo Nogueira and Kyunghyun Cho. Task-oriented query reformulation with reinforcement learning. In Martha Palmer, Rebecca Hwa, and Sebastian Riedel (eds.), *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pp. 574–583, Copenhagen, Denmark, September 2017. Association for Computational Linguistics. doi: 10.18653/v1/D17-1061. URL https://aclanthology.org/D17-1061/.

Yu Pan, Xiaocheng Li, and Hanzhao Wang. Online-optimized rag for tool use and function calling, 2025. URL https://arxiv.org/abs/2509.20415.

Jaewan Park, Solbee Cho, and Jay-Yoon Lee. Stop-RAG: Value-based retrieval control for iterative RAG. In *First Workshop on Multi-Turn Interactions in Large Language Models*, 2025. URL https://openreview.net/forum?id=fN4PJNYgs3.

Sohan Patnaik, Milan Aggarwal, Sumit Bhatia, and Balaji Krishnamurthy. Learning together to perform better: Teaching small-scale LLMs to collaborate via preferential rationale tuning. In Wanxiang Che, Joyce Nabende, Ekaterina Shutova, and Mohammad Taher Pilehvar (eds.), *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 15493–15512, Vienna, Austria, July 2025a. Association for Computational Linguistics. ISBN 979-8-89176-251-0. doi: 10.18653/v1/2025.acl-long.754. URL https://aclanthology.org/2025.acl-long.754/.

Sohan Patnaik, Milan Aggarwal, Sumit Bhatia, and Balaji Krishnamurthy. It helps to take a second opinion: Teaching smaller LLMs to deliberate mutually via selective rationale optimisation. In *The Thirteenth International Conference on Learning Representations*, 2025b. URL https://openreview.net/forum?id=NHxwxc3ql6.

Ofir Press, Muru Zhang, Sewon Min, Ludwig Schmidt, Noah Smith, and Mike Lewis. Measuring and narrowing the compositionality gap in language models. In Houda Bouamor, Juan Pino, and Kalika Bali (eds.), *Findings of the Association for Computational Linguistics: EMNLP 2023*, pp. 5687–5711, Singapore, December 2023. Association for Computational Linguistics. doi: 10.18653/v1/2023.findings-emnlp.378. URL https://aclanthology.org/2023.findings-emnlp.378/.

Mohammad Reza Rezaei and Adji Bousso Dieng. Vendi-rag: Adaptively trading-off diversity and quality significantly improves retrieval augmented generation with llms. *ArXiv*, abs/2502.11228, 2025. URL https://api.semanticscholar.org/CorpusID:276409258.

Stephen E. Robertson, Stephen Walker, Micheline Hancock-Beaulieu, Aarron Gull, and Marianna Lau. Okapi at trec. In *Proceedings of the First Text REtrieval Conference (TREC-1)*, NIST Special Publication 500-207, pp. 21–30, Gaithersburg, MD, USA, 1993. National Institute of Standards and Technology (NIST).

Elisei Rykov, Kseniia Petrushina, Maksim Savkin, Valerii Olisov, Artem Vazhentsev, Kseniia Titova, Alexander Panchenko, Vasily Konovalov, and Julia Belikova. When models lie, we learn: Multilingual span-level hallucination detection with PsiloQA. In Christos Christodoulopoulos, Tanmoy Chakraborty, Carolyn Rose, and Violet Peng (eds.), *Findings of the Association for Computational Linguistics: EMNLP 2025*, pp. 11663–11682, Suzhou, China, November 2025. Association for Computational Linguistics. ISBN 979-8-89176-335-7. doi: 10.18653/v1/2025.findings-emnlp.626. URL https://aclanthology.org/2025.findings-emnlp.626/.

Alireza Salemi and Hamed Zamani. Evaluating retrieval quality in retrieval-augmented generation. In *Proceedings of the 47th International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '24, pp. 2395–2400, New York, NY, USA, 2024. Association for Computing Machinery. ISBN 9798400704314. doi: 10.1145/3626772.3657957. URL https://doi.org/10.1145/3626772.3657957.

Alireza Salemi, Chris Samarinas, and Hamed Zamani. Plan-and-refine: Diverse and comprehensive retrieval-augmented generation, 2025. URL https://arxiv.org/abs/2504.07794.

Kunal Sawarkar, Abhilasha Mangal, and Shivam Raj Solanki. Blended rag: Improving rag (retriever-augmented generation) accuracy with semantic search and hybrid query-based retrievers. In *2024 IEEE 7th International Conference on Multimedia Information Processing and Retrieval (MIPR)*, pp. 155–161. IEEE, August 2024. doi: 10.1109/mipr62202.2024.00031. URL http://dx.doi.org/10.1109/MIPR62202.2024.00031.

Timo Schick, Jane Dwivedi-Yu, Roberto Dessi, Roberta Raileanu, Maria Lomeli, Eric Hambro, Luke Zettlemoyer, Nicola Cancedda, and Thomas Scialom. Toolformer: Language models can teach themselves to use tools. In A. Oh, T. Naumann, A. Globerson, K. Saenko, M. Hardt, and S. Levine (eds.), *Advances in Neural Information Processing Systems*, volume 36, pp. 68539–68551. Curran Associates, Inc., 2023. URL https://proceedings.neurips.cc/paper_files/paper/2023/file/d842425e4bf79ba039352da0f658a906-Paper-Conference.pdf.

Rulin Shao, Rui Qiao, Varsha Kishore, Niklas Muennighoff, Xi Victoria Lin, Daniela Rus, Bryan Kian Hsiang Low, Sewon Min, Wen tau Yih, Pang Wei Koh, and Luke Zettlemoyer. ReasonIR: Training retrievers for reasoning tasks. In *Second Conference on Language Modeling*, 2025. URL https://openreview.net/forum?id=kkBCNLMbGj.

Zhihong Shao, Peiyi Wang, Qihao Zhu, Runxin Xu, Junxiao Song, Xiao Bi, Haowei Zhang, Mingchuan Zhang, Y. K. Li, Y. Wu, and Daya Guo. Deepseekmath: Pushing the limits of mathematical reasoning in open language models, 2024. URL https://arxiv.org/abs/2402.03300.

Guangming Sheng, Chi Zhang, Zilingfeng Ye, Xibin Wu, Wang Zhang, Ru Zhang, Yanghua Peng, Haibin Lin, and Chuan Wu. Hybridflow: A flexible and efficient rlhf framework. In *Proceedings of the Twentieth European Conference on Computer Systems*, EuroSys '25, pp. 1279–1297, New York, NY, USA, 2025. Association for Computing Machinery. ISBN 9798400711961. doi: 10.1145/3689031.3696075. URL https://doi.org/10.1145/3689031.3696075.

Noah Shinn, Federico Cassano, Ashwin Gopinath, Karthik Narasimhan, and Shunyu Yao. Reflexion: language agents with verbal reinforcement learning. In A. Oh, T. Naumann, A. Globerson, K. Saenko, M. Hardt, and S. Levine (eds.), *Advances in Neural Information Processing Systems*, volume 36, pp. 8634–8652. Curran Associates, Inc., 2023. URL https://proceedings.neurips.cc/paper_files/paper/2023/file/1b44b878bb782e6954cd888628510e90-Paper-Conference.pdf.

Haitian Sun, Tania Bedrax-Weiss, and William Cohen. PullNet: Open domain question answering with iterative retrieval on knowledge bases and text. In Kentaro Inui, Jing Jiang, Vincent Ng, and Xiaojun Wan (eds.), *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pp. 2380–2390, Hong Kong, China, November 2019. Association for Computational Linguistics. doi: 10.18653/v1/D19-1242. URL https://aclanthology.org/D19-1242/.

Yiyou Sun, Yu Gai, Lijie Chen, Abhilasha Ravichander, Yejin Choi, Nouha Dziri, and Dawn Song. Why and how LLMs hallucinate: Connecting the dots with subsequence associations. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*, 2025. URL https://openreview.net/forum?id=K2Q4Jp4RbB.

Zhiwen Tan, Jiaming Huang, Qintong Wu, Hongxuan Zhang, Chenyi Zhuang, and Jinjie Gu. Rag-r1:incentivizing the search and reasoning capabilities of llms through multi-query parallelism. *Proceedings of the AAAI Conference on Artificial Intelligence*, 40(39):33187–33195, Mar. 2026. doi: 10.1609/aaai.v40i39.40603. URL https://ojs.aaai.org/index.php/AAAI/article/view/40603.

Lei Wang, Wanyu Xu, Yihuai Lan, Zhiqiang Hu, Yunshi Lan, Roy Ka-Wei Lee, and Ee-Peng Lim. Plan-and-solve prompting: Improving zero-shot chain-of-thought reasoning by large language models. In Anna Rogers, Jordan Boyd-Graber, and Naoaki Okazaki (eds.), *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 2609–2634, Toronto, Canada, July 2023. Association for Computational Linguistics. doi: 10.18653/v1/2023.acl-long.147. URL https://aclanthology.org/2023.acl-long.147/.

Ryen W. White and Resa A. Roth. Exploratory search: Beyond the query-response paradigm. *Synthesis Lectures on Information Concepts, Retrieval, and Services*, 1:98, 2009. URL https://api.semanticscholar.org/CorpusID:35584347.

Lee Xiong, Chenyan Xiong, Ye Li, Kwok-Fung Tang, Jialin Liu, Paul N. Bennett, Junaid Ahmed, and Arnold Overwijk. Approximate nearest neighbor negative contrastive learning for dense text retrieval. In *International Conference on Learning Representations*, 2021. URL https://openreview.net/forum?id=zeFrfgyZln.

Diji Yang, Jinmeng Rao, Kezhen Chen, Xiaoyuan Guo, Yawen Zhang, Jie Yang, and Yi Zhang. Im-rag: Multi-round retrieval-augmented generation through learning inner monologues. In *Proceedings of the 47th International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '24, pp. 730–740, New York, NY, USA, 2024. Association for Computing Machinery. ISBN 9798400704314. doi: 10.1145/3626772.3657760. URL https://doi.org/10.1145/3626772.3657760.

Fangkai Yang, Pu Zhao, Zezhong Wang, Lu Wang, Bo Qiao, Jue Zhang, Mohit Garg, Qingwei Lin, Saravan Rajmohan, and Dongmei Zhang. Empower large language model to perform better on industrial domain-specific question answering. In Mingxuan Wang and Imed Zitouni (eds.), *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing: Industry Track*, pp. 294–312, Singapore, December 2023. Association for Computational Linguistics. doi: 10.18653/v1/2023.emnlp-industry.29. URL https://aclanthology.org/2023.emnlp-industry.29/.

Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik R Narasimhan, and Yuan Cao. React: Synergizing reasoning and acting in language models. In *The Eleventh International Conference on Learning Representations*, 2023. URL https://openreview.net/forum?id=WE_vluYUL-X.

Andrew Yates, Rodrigo Nogueira, and Jimmy Lin. Pretrained transformers for text ranking: BERT and beyond. In Greg Kondrak, Kalina Bontcheva, and Dan Gillick (eds.), *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies: Tutorials*, pp. 1–4, Online, June 2021. Association for Computational Linguistics. doi: 10.18653/v1/2021.naacl-tutorials.1. URL https://aclanthology.org/2021.naacl-tutorials.1/.

Yue Yu, Wei Ping, Zihan Liu, Boxin Wang, Jiaxuan You, Chao Zhang, Mohammad Shoeybi, and Bryan Catanzaro. RankRAG: Unifying context ranking with retrieval-augmented generation in LLMs. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*, 2024. URL https://openreview.net/forum?id=S1fc92uemC.

Tianyi Zhang*, Varsha Kishore*, Felix Wu*, Kilian Q. Weinberger, and Yoav Artzi. Bertscore: Evaluating text generation with bert. In *International Conference on Learning Representations*, 2020. URL https://openreview.net/forum?id=SkeHuCVFDr.

Xiaoming Zhang, Ming Wang, Xiaocui Yang, Daling Wang, Shi Feng, and Yifei Zhang. Hierarchical retrieval-augmented generation model with rethink for multi-hop question answering. *ArXiv*, abs/2408.11875, 2024. URL https://api.semanticscholar.org/CorpusID:271924151.

Yuyu Zhang, Ping Nie, Arun Ramamurthy, and Le Song. Answering any-hop open-domain questions with iterative document reranking. In *Proceedings of the 44th International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '21, pp. 481–490, New York, NY, USA, 2021. Association for Computing Machinery. ISBN 9781450380379. doi: 10.1145/3404835.3462853. URL https://doi.org/10.1145/3404835.3462853.

Denny Zhou, Nathanael Schärli, Le Hou, Jason Wei, Nathan Scales, Xuezhi Wang, Dale Schuurmans, Claire Cui, Olivier Bousquet, Quoc V Le, and Ed H. Chi. Least-to-most prompting enables complex reasoning in large language models. In *The Eleventh International Conference on Learning Representations*, 2023. URL https://openreview.net/forum?id=WZH7099tgfM.

Rongzhi Zhu, Xiangyu Liu, Zequn Sun, Yiwei Wang, and Wei Hu. Mitigating lost-in-retrieval problems in retrieval augmented multi-hop question answering. In Wanxiang Che, Joyce Nabende, Ekaterina Shutova, and Mohammad Taher Pilehvar (eds.), *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 22362–22375, Vienna, Austria, July 2025. Association for Computational Linguistics. ISBN 979-8-89176-251-0. doi: 10.18653/v1/2025.acl-long.1089. URL https://aclanthology.org/2025.acl-long.1089/.

A Appendix

A.1 Limitations and Future Work

While RIPPLE demonstrates strong gains through iterative, corpus-informed planning, several aspects remain open for future exploration. First, the framework currently relies on an LLM-based relevance judge and a fixed stopping threshold (τ) to determine when sufficient evidence has been retrieved, which assumes a relatively discrete notion of relevance and does not model graded evidence accumulation or dynamically adapt the search horizon based on uncertainty or partial coverage. Developing more principled mechanisms for adaptive stopping and evidence sufficiency estimation remains an important direction. Second, iterative planning introduces additional computational overhead compared to single-pass search methods. Although our ablations show that performance saturates with a small number of iterations and rollouts (Appendices A.7 and A.11), improving efficiency through lightweight relevance estimation and learned early stopping remains a promising avenue for future work.

A.2 Dual-Mode Training State Construction

RIPPLE is executed iteratively at inference time, so the planner encounters two qualitatively different classes of states: an initial state before any planner-generated retrieval has occurred, and feedback-conditioned states arising after one or more search attempts. Training only on the former would teach corpus-grounded query decomposition without exposing the policy to retrieval failures, whereas training only on the latter would omit the first-round exploration behavior required to initiate search. We therefore construct training states in two complementary modes.

Initial-state mode. For each training query q, we retrieve the initial candidate set $\mathcal{R}_{L_0}^{(0)}$ and construct the MMR corpus snapshot $\mathcal{C}_0^M$ as described in Section 3.2. The resulting state is $\mathcal{X}_0 = \{q, \mathcal{C}_0^M, \mathcal{I}\}$. No planner-generated subqueries or failure summaries are available at this point. Candidate plans sampled from $\pi_\theta(\cdot \mid \mathcal{X}_0)$ therefore train the policy to identify complementary search directions from the original information need and its corpus-grounded snapshot.

Failure-conditioned mode. Later training states correspond to points reached after one or more search attempts. For a state at round $t > 0$, we retain the fixed query-specific snapshot $\mathcal{C}_0^M$ and augment it with the preceding subqueries $\mathcal{Q}_{<t}$ and summaries $\mathcal{S}_{<t}^-$ of retrieved documents judged non-relevant, yielding $\mathcal{X}_t = \{q, \mathcal{C}_0^M, \mathcal{Q}_{<t}, \mathcal{S}_{<t}^-, \mathcal{I}\}$. These states expose the planner to search directions already attempted and evidence found insufficient, allowing candidate plans to be learned specifically in the context of previous retrieval outcomes.

State collection and sampling. Later-round states are generated online by executing the current planner policy through the same retrieval loop used at inference time. Starting from $\mathcal{X}_0$, the planner generates $\mathcal{Q}_0$, the resulting documents are retrieved and judged, and the cumulative evidence and failure summaries are updated. If the stopping criterion is not satisfied, these outputs define the next state $\mathcal{X}_1$. This process continues recursively, producing $\mathcal{X}_t$ from the actual search history generated in preceding rounds. Thus, failure-conditioned states are not synthetically constructed: they arise from the planner's own unsuccessful retrieval trajectories and contain the corresponding prior subqueries and summaries of non-relevant evidence. During training, states from both the initial and later rounds are used for GRPO updates, exposing the policy to first-round exploration as well as recovery from its own retrieval failures.

For either mode, once a state $\mathcal{X}_t$ is selected, we sample G candidate planner outputs from the same policy state. Each candidate is executed independently through retrieval and relevance judgment, its

newly retrieved relevant evidence is combined with the evidence accumulated before round t, and an intermediate answer is generated from this candidate-specific cumulative evidence. Candidate utilities are then compared through the GRPO objective described in Section 3.2. Consequently, dual-mode construction determines *which search states the policy learns from*, while round-level outcome supervision determines *how alternative actions at each state are credited*.

This distinction is important: failure feedback is part of the planner state rather than a separate reward. The later-state mode teaches the policy to condition its next search decision on unsuccessful retrieval outcomes, while round-level answer utility provides the learning signal for whether the resulting decision improves downstream answering.

A.3 Evaluation Protocol

For each dataset instance, let q_i denote the input question, A_i denote the generated answer, A_i^* denote the reference answer, and $\mathcal{E}_i$ denote the final evidence set used by the generator. For RIPPLE, $\mathcal{E}_i$ consists of the retrieved documents judged relevant during the iterative search process. For baselines, $\mathcal{E}_i$ denotes the documents retrieved or selected by that method and passed to the generator. All systems are evaluated using the same metric computation pipeline, and the exact evaluator prompts are reported in Appendix A.4.

LLM-Based Evaluation. We use GPT-4o as the LLM evaluator, following prior work using LLMs for answer-quality assessment and response evaluation (Salemi et al., 2025; Yang et al., 2023). For each example, the evaluator receives the generated answer A_i and the reference answer A_i^*, and returns a binary judgment $s_i^{\mathrm{LLM}} \in \{0, 1\}$ indicating whether A_i conveys the same core information as A_i^*; exact lexical agreement is not required. The final reported LLM evaluation score is the mean over the N evaluation examples:

$$\text{LLM-Eval} = \frac{1}{N} \sum_{i=1}^{N} s_i^{\mathrm{LLM}}. \tag{7}$$

Can-Answer Rate (CAR). Following Yang et al. (2023), we use Can-Answer Rate (CAR) to measure how often the evidence passed to the generator is sufficient for it to commit to an answer. Because the answer-generation prompt instructs the generator to abstain when the provided contexts do not support an answer, the evaluator receives A_i and assigns $c_i = 0$ if A_i is an abstention and $c_i = 1$ otherwise. CAR is therefore an end-to-end measure of evidence sufficiency that is independent of how closely A_i matches A_i^*:

$$\text{CAR} = \frac{1}{N} \sum_{i=1}^{N} c_i. \tag{8}$$

ROUGE-L and BERTScore. We employ ROUGE-L (Lin, 2004) as the lexical-overlap metric to measure sequence-level overlap between the generated and reference answers. To evaluate semantic overlap between the generated and reference answers, we use BERTScore (Zhang* et al., 2020) and report the F1 score.

A.4 Prompt Templates

All prompts used in our experiments are reported verbatim below.

Planner Prompt

```
You are a query planning assistant. Given a user question and some
initial context documents (and/or failure subqueries generated in
previous iterations along with summaries of documents retrieved
using those subqueries), generate specific search queries to find
MISSING information needed to answer the question comprehensively.
**IMPORTANT: You MUST follow the EXACT output format shown below.
Each query must have TWO lines: RATIONALE and QUERY.**
**OUTPUT FORMAT (you must use this exact structure):** RATIONALE:
<explain why this specific information is missing and needed> QUERY:
<concise 10-12 word search query>
RATIONALE: <explain why another piece of information is missing and
needed> QUERY: <another concise 10-12 word search query>
**EXAMPLE OUTPUT:** RATIONALE: The context lacks information about
the founding date of the company QUERY: When was Microsoft
corporation founded year
RATIONALE: Need to know the current CEO to answer leadership
question QUERY: Who is the current CEO of Microsoft
**INSTRUCTIONS:** 1. Analyze the provided context to identify
MISSING information 2. For each missing piece, write RATIONALE:
then explain why it's needed 3. On the next line, write QUERY:
then a concise 10-12 word search query 4. Generate between 1 and
<max subqueries> query pairs 5. DO NOT add numbering, bullets, or
any other formatting 6. Each query pair must be exactly 2 lines:
RATIONALE line, then QUERY line
**User Question:** <question>
**Available Context:** context text contains corpus snapshot,
failure subqueries from previous iterations and summaries of
retrieved documents from those subqueries
**Now generate your search queries in the EXACT format shown above
(RATIONALE: ... QUERY: ...):**
```

Relevance Judge Prompt

```
You are a relevance judge. Evaluate if the following document is
relevant for answering the user's question.
**User Question:** <question>
**Retrieved via:** <source>
**Document:** <document>
**Instructions:** 1. Summarize the key information in the document
(1-2 sentences) 2. Determine if this information is relevant to
answering the user's question 3. Respond with EXACTLY this format:
SUMMARY: [your 1-2 sentence summary]
RELEVANT: [YES or NO]
```

Answer Generation Prompt

```
Answer the question only using the provided contexts. If the
contexts are irrelevant or incorrect and do not contain or support
the answer, then respond that you cannot answer based on the
provided context.
Do not add any other information.
Question: <question>
Contexts: <context>
Answer:
```

Table 7: Generator-agnostic evaluation across ANTIQUE, TechQA, and MSQA. We compare RIPPLE with Search-R1 using different answer generators while holding the planner and retriever fixed, and report LLM-based evaluation (LLM) and Can-Answer Rate (CAR) in $[0, 1]$. **Bold** marks the better value within each generator pair.

Generator	Method	ANTIQUE		TechQA		MSQA	
		LLM	CAR	LLM	CAR	LLM	CAR
GPT-5.1	Search-R1	0.356	0.708	0.348	0.682	0.189	0.552
	RIPPLE	**0.452**	**0.801**	**0.448**	**0.812**	**0.261**	**0.721**
Claude Sonnet 4.6	Search-R1	0.348	0.696	0.339	0.671	0.182	0.538
	RIPPLE	**0.438**	**0.784**	**0.433**	**0.798**	**0.252**	**0.705**
Claude Opus 4.6	Search-R1	0.352	0.702	0.343	0.676	0.186	0.545
	RIPPLE	**0.446**	**0.793**	**0.441**	**0.806**	**0.257**	**0.713**
Gemini 2.5 Pro	Search-R1	0.341	0.689	0.332	0.662	0.178	0.529
	RIPPLE	**0.431**	**0.771**	**0.427**	**0.786**	**0.245**	**0.692**

LLM Evaluation Prompt

```
You are an expert evaluator. Your task is to determine if the
assistant's answer matches semantically or aligns closely to the
ground truth answer. Match semantically or align closely can mean
the assistant's answer conveys the same core information and
meaning as the ground truth answer. The assistant's answer need
not exactly verbatim match the ground truth answer. Respond with a
single digit only: '1' if the answers align or match, '0' if they
do not. Do not provide any explanation or other text.
Ground-truth Answer: <groundtruth answer>
Assistant's Answer: <assistant answer>
Response:
```

Can Answer Rate Prompt

```
You are an expert at analyzing AI assistant responses. Your task
is to determine if the following response is a refusal to answer.
A refusal might be a phrase like "Sorry, I cannot give a confident
answer", "I do not understand your question", or similar statements
indicating an inability to answer. Respond with only "Yes" if it
IS a refusal, or "No" if it IS NOT a refusal.
Assistant's Answer: <assistant answer>
Response:
```

A.5 Generator-Agnostic Performance

We evaluate whether the gains from RIPPLE depend on a specific answer generator by comparing against the strongest baseline, Search-R1, across multiple state-of-the-art LLMs, including GPT-5.1, Claude Sonnet 4.6, Claude Opus 4.6, and Gemini 2.5 Pro, on ANTIQUE, TechQA, and MSQA. As shown in Table 7, RIPPLE consistently outperforms Search-R1 across all datasets and generators. On ANTIQUE, for example, RIPPLE improves LLM evaluation from 0.356 to 0.452 and CAR from 0.708 to 0.801 with GPT-5.1, with similar gains observed across other generators. On TechQA, improvements remain consistent, with RIPPLE achieving 0.448 LLM and 0.812 CAR compared to 0.348 and 0.682 for Search-R1. On the more challenging MSQA benchmark, RIPPLE continues to provide substantial gains, improving LLM evaluation by 6.7–7.2 points and CAR by 16.3–16.9 points across generators. Importantly, the relative improvements remain stable across different LLM backbones, indicating that the gains are not tied to a specific generator. Instead, they arise from improved retrieval quality and more effective planning. Overall, these results demonstrate that

Table 8: Out-of-distribution evaluation on **HotpotQA**. The planner is trained on ANTIQUE and evaluated directly on HotpotQA without task-specific fine-tuning. We report ROUGE-L (R-L), BERTScore F1 (BERT), LLM-based evaluation (LLM), and Can-Answer Rate (CAR) in $[0, 1]$. **Bold** marks the best value per column and underline the strongest baseline. Unlike Table 4, no method is trained on HotpotQA here.

Method	R-L	BERT	LLM	CAR
Vanilla RAG (Lewis et al., 2020a)	0.211	0.781	0.298	0.612
Retrieval-centric Approaches				
BM25 (Robertson et al., 1993)	0.205	0.774	0.281	0.598
DIVER (Long et al., 2025)	0.219	0.786	0.305	0.631
ReasonIR (Shao et al., 2025)	0.224	0.790	0.318	0.648
Planning & Reasoning-based Approaches				
CoT-RAG (Li et al., 2025a)	0.180	0.752	0.162	0.402
Plan-and-Refine (Salemi et al., 2025)	0.168	0.741	0.140	0.365
ReAct (Yao et al., 2023)	0.202	0.770	0.223	0.498
Self-RAG (Asai et al., 2024)	0.195	0.763	0.201	0.451
RaDeR (Das et al., 2025)	0.232	0.796	0.342	0.672
RL-based & Iterative Retrieval-Generation Methods				
R3-RAG (Li et al., 2025b)	0.235	0.798	0.351	0.688
RAG-R1 (Tan et al., 2026)	0.238	0.801	0.359	0.701
DeepRetrieval (Jiang et al., 2025a)	0.233	0.795	0.346	0.679
Search-R1 (Jin et al., 2025)	0.242	0.806	0.372	0.714
RIPPLE (trained on ANTIQUE)	**0.261**	**0.821**	**0.417**	**0.756**

RIPPLE enhances retrieval sufficiency and answer quality in a generator-agnostic manner, making it broadly applicable across different LLM architectures.

A.6 Out-of-Distribution Generalization on HotpotQA

We evaluate the generalization ability of RIPPLE by training the planner exclusively on ANTIQUE and directly evaluating on HotpotQA without any task-specific fine-tuning. This setting tests whether the model learns transferable search strategies rather than dataset-specific heuristics. As shown in Table 8, RIPPLE consistently outperforms both classical retrieval baselines and recent reinforcement learning-based approaches. Among the baselines, Search-R1 achieves the strongest performance with an LLM score of 0.372 and CAR of 0.714, reflecting the benefits of jointly optimizing retrieval and reasoning. R3-RAG and RAG-R1 likewise improve over earlier planning-based methods, highlighting the effectiveness of learned query reformulation. Despite these strong baselines, RIPPLE achieves substantial gains, improving the LLM score to 0.417 and CAR to 0.756, a $+4.5$ point improvement in LLM evaluation and $+4.2$ points in CAR over Search-R1. Notably, these gains are achieved without any exposure to HotpotQA during training. These results indicate that RIPPLE learns a corpus-aware planning policy that generalizes across datasets with different structures and reasoning requirements. In contrast to prior approaches that rely on prompt-based decomposition or dataset-specific optimization, our method explicitly models retrieval feedback and failure signals, enabling more robust and transferable search behavior.

A.7 Effect of Inference-Time Search Iterations

Figure 3 varies the number of planner inference iterations t at a fixed iteration budget $T = 5$. Both metrics improve monotonically with additional iterations on all three datasets. On ANTIQUE, LLM Eval increases from 0.350 at one iteration to 0.418 at five, while CAR improves from 0.690 to 0.762; on TechQA the corresponding values move from 0.327 to 0.419 and 0.671 to 0.769, and on MSQA from 0.162 to 0.233 and 0.547 to 0.690. Gains flatten beyond four iterations, indicating diminishing returns from further search. Notably, from the second iteration onward RIPPLE already exceeds the final scores of Search-R1 in Table 1 on all three datasets, even though Search-R1 also performs

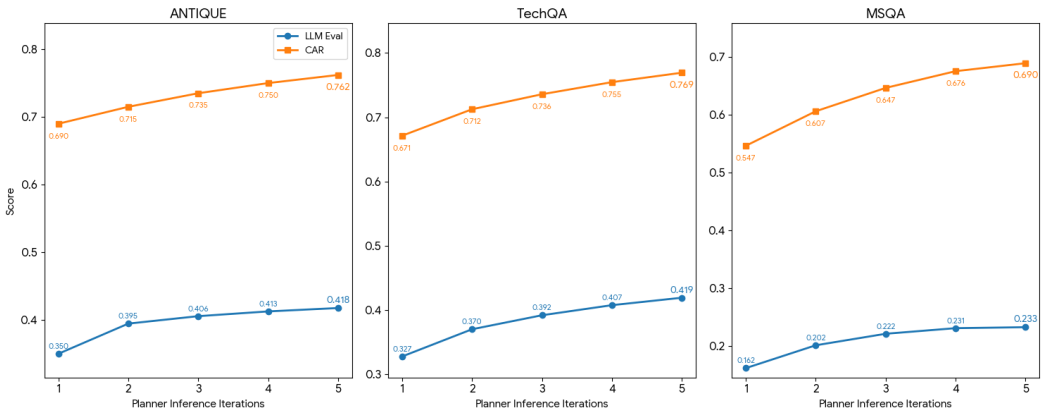


Figure 3: Effect of the number of planner inference iterations t on LLM-based evaluation and Can-Answer Rate across ANTIQUE, TechQA, and MSQA. Both metrics improve monotonically and begin to plateau beyond four iterations.

Table 9: Effect of the MMR trade-off parameter λ used for corpus snapshot construction, reported as LLM-based evaluation (LLM) and Can-Answer Rate (CAR) in $[0, 1]$. Lower λ favors diversity and higher λ favors query relevance. **Bold** marks the best value per column, which is also the default setting used throughout the paper.

MMR Trade-off (λ)	**ANTIQUE**		**TechQA**		**MSQA**	
	LLM	**CAR**	**LLM**	**CAR**	**LLM**	**CAR**
0.0 (max. diversity)	0.323	0.638	0.374	0.694	0.198	0.603
0.2	0.385	0.719	0.405	0.749	0.222	0.663
0.4	**0.418**	**0.762**	**0.419**	**0.769**	**0.233**	**0.690**
0.6	0.410	0.750	0.415	0.764	0.230	0.681
0.8	0.415	0.757	0.417	0.767	0.232	0.688
1.0 (max. relevance)	0.371	0.701	0.400	0.739	0.219	0.654

multiple rounds of retrieval. This indicates that the advantage comes from the quality of the learned search decisions rather than from a larger search budget alone.

A.8 Sensitivity to the MMR Relevance–Diversity Trade-off

Table 9 varies the MMR trade-off parameter λ that governs how the corpus snapshot balances query relevance against diversity. Performance peaks at the intermediate value $\lambda = 0.4$ on all three datasets. On ANTIQUE, LLM Eval rises from 0.323 at $\lambda = 0$ to 0.418 at $\lambda = 0.4$ and falls back to 0.371 at $\lambda = 1$, with CAR following the same pattern ($0.638 \rightarrow 0.762 \rightarrow 0.701$); TechQA and MSQA show the same inverted-U shape. The two endpoints fail for different reasons: at $\lambda = 0$ the snapshot is diverse but weakly tied to the query, whereas at $\lambda = 1$ it reduces to the plain top-k ranking and reintroduces redundancy, which is the same configuration reported as *Plain top-k* in Table 3. Both endpoints therefore reduce the number of distinct aspects the planner can condition on.

A.9 Impact of Retrieval Quality on Performance of RIPPLE

Table 10 presents an ablation study over different dense retriever backbones while keeping the planner and generator fixed. Across all three benchmarks, we observe that stronger retrievers consistently improve both answer quality (LLM Eval) and evidence sufficiency (CAR), indicating that RIPPLE can effectively leverage improvements in the underlying retrieval model. In particular, BGE-M3 achieves the best performance across all datasets, improving LLM Eval on ANTIQUE from 0.418 with Snowflake to 0.432 and CAR from 0.762 to 0.782, with similar gains on TechQA and MSQA. Weaker or less specialized retrievers, such as Qwen-Emb-0.6B and Embedding-Gemma, lead to noticeable drops in both metrics, reflecting reduced retrieval quality. Importantly, the relative ranking

Table 10: Retriever backbone ablation under a fixed planner and generator, reported as LLM-based evaluation (LLM) and Can-Answer Rate (CAR) in $[0, 1]$. Snowflake is the default configuration used throughout the paper. **Bold** marks the best value per column and underline the default.

Retriever	ANTIQUE		TechQA		MSQA	
	LLM	CAR	LLM	CAR	LLM	CAR
Snowflake	0.418	0.762	0.419	0.769	0.233	0.690
BGE-M3	**0.432**	**0.782**	**0.425**	**0.776**	**0.238**	**0.699**
Qwen-Emb-0.6B	0.365	0.688	0.393	0.726	0.217	0.651
Embedding-Gemma	0.392	0.721	0.407	0.749	0.225	0.673

Table 11: Effect of adding explicit claim-based evidence coverage to the default RIPPLE objective. Results are reported using LLM-based evaluation (LLM) and Can-Answer Rate (CAR).

	ANTIQUE		TechQA		MSQA	
Setting	LLM	CAR	LLM	CAR	LLM	CAR
Default RIPPLE	0.418	0.762	0.419	0.769	0.233	0.690
+ claim-based evidence coverage	**0.427**	**0.778**	**0.431**	**0.786**	**0.245**	**0.712**

of retrievers is consistent across datasets, suggesting that while corpus-aware planning provides substantial gains over baseline methods, overall performance remains sensitive to the quality of the retriever. Even with the weakest retriever, Qwen-Emb-0.6B, RIPPLE still exceeds the strongest baseline in Table 1 on all three benchmarks.

A.10 Claim-Based Evidence Coverage Reward

The default RIPPLE objective uses downstream answer correctness as its primary task-level signal, together with auxiliary rewards for format adherence, query relevance, and diversity. We additionally investigate whether directly supervising evidence coverage provides a complementary learning signal. Specifically, we use GPT-4o to extract a set of atomic claims $\mathcal{C}(A^*) = \{c_1, \ldots, c_m\}$ from the reference answer A^*. For the cumulative relevant evidence $\mathcal{E}_t^+$ available after search round t, GPT-4o independently determines whether each claim c_j is supported by the retrieved evidence. We define the claim-based evidence coverage reward as

$$U_{\text{cov}}^{(t)} = \frac{1}{|\mathcal{C}(A^*)|} \sum_{c_j \in \mathcal{C}(A^*)} \mathbb{I}\big[\text{Supported}(c_j, \mathcal{E}_t^+)\big]\,. \tag{9}$$

This reward is added to the default round-level objective as

$$U_t^{\text{cov}} = U_t + \lambda_{\text{cov}} U_{\text{cov}}^{(t)}, \tag{10}$$

where $\lambda_{\text{cov}} = 0.1$. All other training and inference settings remain unchanged.

As shown in Table 11, explicit evidence-coverage supervision provides consistent but modest improvements across all three benchmarks. This suggests that round-level answer correctness already captures substantial information about the downstream utility of retrieved evidence, while direct supervision over reference-claim coverage provides an additional complementary signal. Importantly, claim-based coverage is not part of the default RIPPLE objective used in our main experiments.

A.11 Effect of Increasing the Number of Planning Rollouts During Training

Figure 4 analyzes the impact of varying the number of planner rollouts G generated during training under GRPO across ANTIQUE, TechQA, and MSQA, evaluated using LLM Eval and CAR. Increasing the number of rollouts consistently improves performance across datasets, indicating that sampling more candidate plans during training enables more effective exploration of the plan space. On ANTIQUE, for example, LLM Eval improves from 0.373 at two rollouts to 0.431 at eight rollouts, while CAR increases from 0.704 to 0.779. Similar monotonic trends are observed on TechQA and

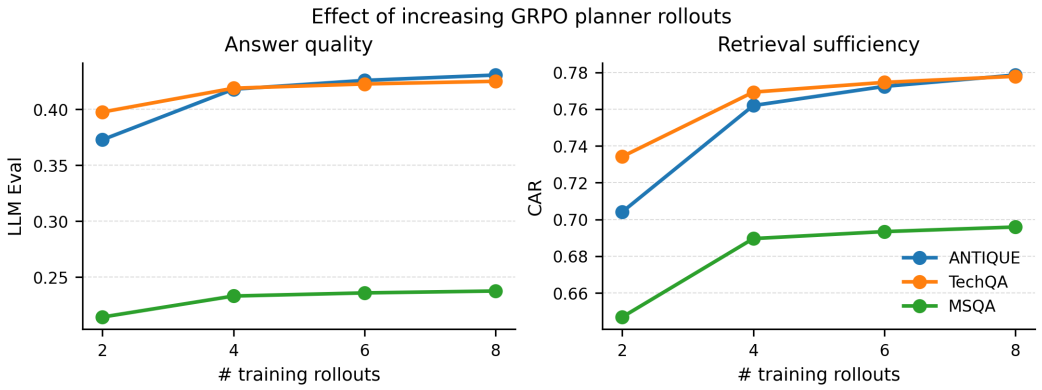


Figure 4: Effect of increasing the number of GRPO planner rollouts G during training, measured by LLM-based evaluation (left) and Can-Answer Rate (right). Performance improves consistently with more rollouts across ANTIQUE, TechQA, and MSQA, with gains saturating around six to eight rollouts.

Table 12: Impact of the initial retrieval budget L_0 and MMR compression to snapshot size M on the performance of RIPPLE, reported as LLM-based evaluation (LLM) and Can-Answer Rate (CAR) in $[0, 1]$. **Bold** marks the best value per column and underline the default $50 \rightarrow 10$ configuration used throughout the paper.

Context Size ($L_0 \rightarrow M$)	**ANTIQUE**		**TechQA**		**MSQA**	
	LLM	**CAR**	**LLM**	**CAR**	**LLM**	**CAR**
$50 \rightarrow 10$	0.418	0.762	0.419	0.769	0.233	0.690
$20 \rightarrow 10$	0.387	0.722	0.405	0.748	0.222	0.661
$100 \rightarrow 10$	0.411	0.752	0.414	0.763	0.229	0.681
$50 \rightarrow 5$	0.347	0.663	0.382	0.707	0.204	0.622
$50 \rightarrow 20$	**0.422**	**0.769**	**0.421**	**0.774**	**0.236**	**0.695**

MSQA, with gains diminishing beyond six rollouts and saturating by eight. We therefore use $G = 4$ throughout, which recovers most of the achievable gain at roughly half the rollout cost of $G = 8$.

A.12 Ablation on Corpus Snapshot Design for Planning

Table 12 studies the effect of the initial retrieval budget L_0 and MMR-based compression to a fixed context size M on planner performance across ANTIQUE, TechQA, and MSQA. Both factors substantially influence the quality of the planner's conditioning context. A moderate initial retrieval budget with balanced compression ($50 \rightarrow 10$) yields strong and stable performance, achieving, for example, an LLM Eval of 0.418 and CAR of 0.762 on ANTIQUE and 0.419 / 0.769 on TechQA. Reducing the initial budget ($20 \rightarrow 10$) leads to consistent performance drops, indicating insufficient coverage, while increasing it further ($100 \rightarrow 10$) offers diminishing returns. Varying the compressed context size reveals a similar trade-off: aggressive compression ($50 \rightarrow 5$) degrades performance due to information loss, whereas expanding the snapshot ($50 \rightarrow 20$) yields modest gains and the highest CAR scores. Overall, these results highlight the importance of balancing retrieval coverage and context compactness when constructing planner contexts.

A.13 Effect of Subquery Budget per Plan on Performance

Figure 5 analyzes the impact of varying the number of planner-generated subqueries K per iteration on answer quality (LLM Eval) and evidence sufficiency (CAR), reported as the change relative to the default budget $K = 5$. Increasing the subquery budget yields consistent but modest performance improvements across datasets, indicating that multiple complementary subqueries improve coverage of the information need. On ANTIQUE, for example, increasing the budget from 5 to 10 subqueries

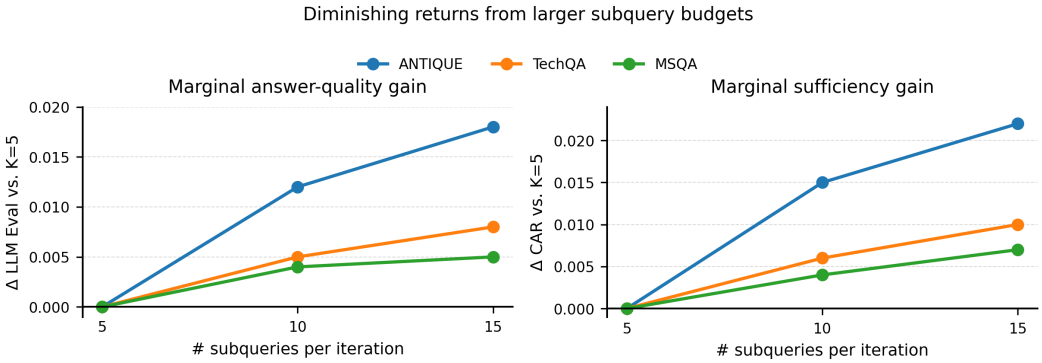


Figure 5: Effect of the number of planner-generated subqueries K per iteration, plotted as the change in LLM-based evaluation (left) and Can-Answer Rate (right) relative to the default budget $K = 5$. Increasing the subquery budget improves both metrics across ANTIQUE, TechQA, and MSQA, with gains becoming marginal beyond ten subqueries. Absolute scores at $K = 5$ are those in Table 1.

Table 13: Effect of augmenting a trained Search-R1 policy with RIPPLE planner context without retraining, reported as LLM-based evaluation (LLM) and Can-Answer Rate (CAR) in $[0, 1]$. Both augmented variants fall below unmodified Search-R1.

Setting	**ANTIQUE**		**TechQA**		**MSQA**	
	LLM	CAR	LLM	CAR	LLM	CAR
Search-R1	0.332	0.695	0.336	0.668	0.174	0.528
+ MMR snapshot	0.271	0.638	0.278	0.611	0.118	0.467
+ snapshot + failure summaries	0.284	0.649	0.286	0.618	0.125	0.474
RIPPLE	**0.418**	**0.762**	**0.419**	**0.769**	**0.233**	**0.690**

improves LLM Eval from 0.418 to 0.430 and CAR from 0.762 to 0.777, with only marginal gains beyond 10 subqueries. Similar trends are observed on TechQA and MSQA. Overall, the results suggest diminishing returns from larger subquery budgets, while inference-time cost and search overhead increase proportionally, indicating that moderate subquery budgets provide a practical trade-off between performance and efficiency.

A.14 Augmenting Search-R1 with RIPPLE Planner Context

To determine whether the gains from RIPPLE arise simply from providing additional corpus context, we augment a fully trained Search-R1 model with the same information available to the RIPPLE planner at inference time, without retraining Search-R1. We consider two variants: adding the diversity-optimized MMR snapshot, and adding both the snapshot and summaries of unsuccessful retrievals.

As shown in Table 13, simply prepending the additional context does not reproduce the gains of RIPPLE. In fact, both augmented Search-R1 variants perform below the original Search-R1 across all three datasets, with ANTIQUE LLM Eval falling from 0.332 to 0.271 and 0.284. This indicates that the benefit of the snapshot and failure feedback does not arise from exposing an existing policy to additional context at inference time; the planner must learn how to use these signals during training. This result complements the planner-context ablations in Table 3, which separately establish the contributions of the diversity-optimized snapshot and failure feedback.

A.15 Scalability with Corpus Size

We further analyze how corpus size affects the cost of constructing the diversity-optimized snapshot. Snapshot construction consists of retrieving the initial candidate set followed by MMR selection; only

Table 14: Snapshot-construction latency as the underlying corpus size increases, in seconds per query. **Snapshot** is the sum of the retrieval and MMR stages, and **Total / Share** reports end-to-end inference latency and the fraction contributed by snapshot construction.

Corpus	Retrieval	MMR	Snapshot	Total / Share
10K	0.14s	0.08s	0.22s	11.07s / 2.0%
50K	0.31s	0.08s	0.39s	11.24s / 3.5%
100K	0.57s	0.08s	0.65s	11.50s / 5.7%
500K	1.24s	0.08s	1.32s	12.17s / 10.8%

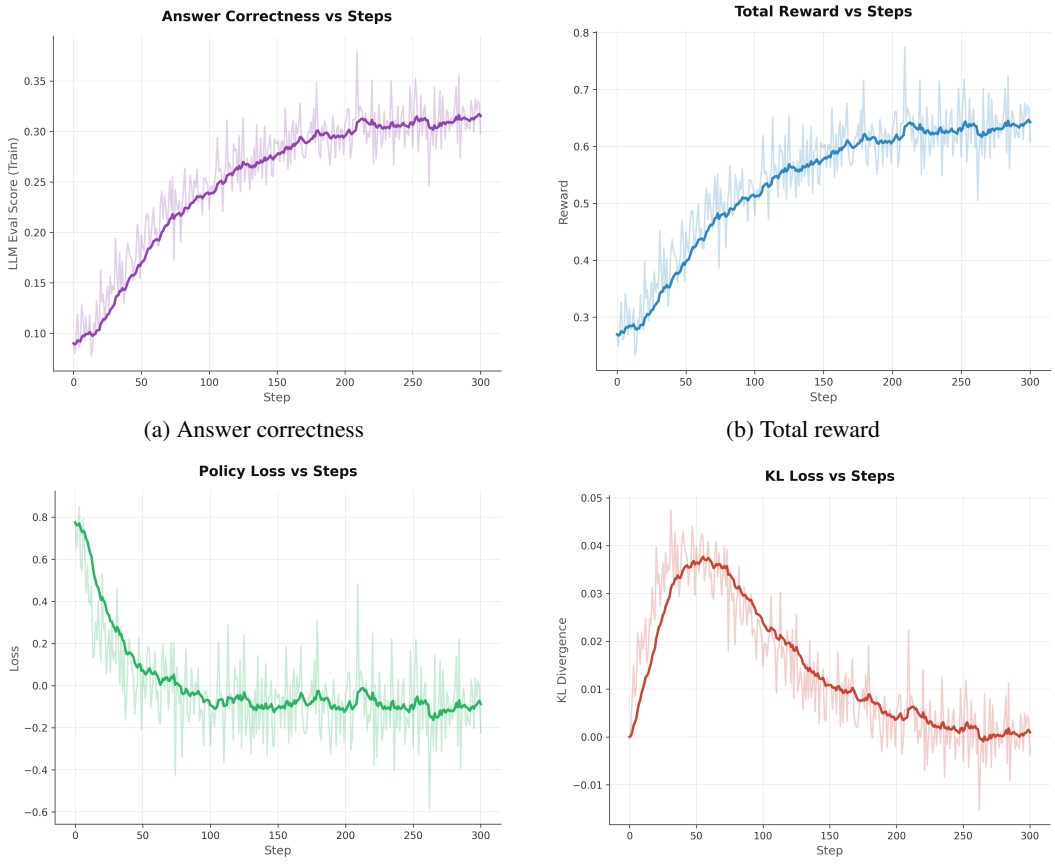
the retrieval stage operates over the underlying corpus, whereas MMR is applied to the fixed-size retrieved candidate set.

As shown in Table 14, when corpus size increases from 10K to 500K documents, initial retrieval latency increases from 0.14s to 1.24s, while MMR selection remains approximately constant at 0.08s because it operates over a fixed-size candidate set. Consequently, snapshot construction increases from 0.22s to 1.32s and accounts for 2.0%–10.8% of total inference latency in this experiment, which remains within the end-to-end budget reported in Table 5. Since the snapshot is reconstructed from the retrieval index for each query, newly indexed documents can participate in subsequent snapshot construction without modifying the planner.

A.16 Planner Training Dynamics

We analyze the optimization dynamics of the planner during GRPO training. Figure 6 reports answer correctness, total reward, policy loss, and KL divergence over training steps.

Answer correctness increases from approximately 0.09 at the beginning of training to above 0.31 at later steps, while total reward increases from approximately 0.27 to 0.64. Both exhibit rapid improvement during the earlier stages of training followed by slower gains and stabilization. Policy loss decreases sharply during early optimization and subsequently fluctuates within a comparatively stable range. KL divergence initially increases, reaching its highest values during early training, and then progressively decreases toward zero. Together, these dynamics indicate that the planner learns rapidly during early optimization and subsequently reaches a more stable training regime as improvements in answer correctness and reward begin to plateau.



(a) Answer correctness

(b) Total reward

(c) Policy loss

(d) KL divergence

Figure 6: Training dynamics of RIPPLE during GRPO optimization. The solid curves show smoothed trends and the lighter curves show per-step values. Answer correctness and total reward increase throughout training and stabilize at later steps, while policy loss decreases sharply during early optimization and subsequently stabilizes. KL divergence initially increases as the policy departs from the reference policy and progressively decreases thereafter.