APEX-Searcher: Refining Credit Assignment with Subgoaling for Agentic Retrieval-Augmented Generation

Anonymous ACL submission

Abstract

Retrieval-augmented generation (RAG) connects large language models (LLMs) to external knowledge, but single-round retrieval is often insufficient for complex multi-hop questions. To enhance search capabilities for complex tasks, most existing works integrate multi-round iterative retrieval with reasoning processes via end-to-end training. While these approaches improve problem-solving performance, they still face challenges in task reasoning and model training, especially ambiguous retrieval execution paths and sparse rewards in end-to-end reinforcement learning (RL), which can lead to inaccurate retrieval results and lower performance. We attribute these failures to *hierarchical credit entanglement*: a single final reward updates planning and execution together, so the model cannot clearly separate plan errors from retrieval errors. We propose APEX-Searcher, which uses a *Refining Credit Assignment* paradigm: planning is optimized by RL with a plan-level reward, while execution is learned by SFT. Extensive experiments show consistent gains in both multi-hop RAG and task planning across benchmarks.

1 Introduction

Large Language Models (LLMs) can process, generate, and understand language across many applications (Zhao et al., 2023). However, their knowledge is bounded by static pre-training parameters (Roberts et al., 2020). They cannot access post-cutoff or rapidly changing facts (Lewis et al., 2020; Kandpal et al., 2023), and they may hallucinate plausible but incorrect content on out-of-distribution or fact-sensitive queries (Huang et al., 2025; Cossio, 2025).

Recently, Retrieval-Augmented Generation (RAG) has become a common way to connect LLMs to external knowledge sources (Lewis et al., 2020). RAG is effective for reducing hallucinations and giving models access to up-to-date information (e.g., Wikipedia or a proprietary database) in knowledge-intensive tasks (Gao et al., 2023). However, standard RAG works best when the required information can be found in a single retrieval pass. It often struggles with complex queries, especially multi-hop questions (Mavi et al., 2024), where the model must combine several related pieces of evidence to derive the final answer (Yang et al., 2018; Ho et al., 2020; Trivedi et al., 2022).

To address these limitations, iterative RAG (Yao et al., 2023; Press et al., 2023; Trivedi et al., 2023; Li et al., 2025) has emerged as the prevailing approach for multi-hop question answering. In this paradigm, the model engages in multiple rounds of retrieval and generation. Inspired by these works, recent studies have explored agentic RAG (Li et al., 2025; Leng et al., 2025), which treat retrieval as a callable tool, enabling large models to independently decide when to invoke the retrieval tool and dynamically adjust strategies. Reinforcement learning (RL) based on LLM is often used as a training method to improve the search capabilities of agentic RAG, aiming to equip LLMs with combined reasoning and search ability through RL (Jin et al., 2025; Chen et al., 2025; Song et al., 2025; Zheng et al., 2025; Sun et al., 2025a; Wang et al., 2025b; Zhang et al., 2025a).

Although existing iterative RAG and agentic RAG methods improve search by combining multi-round retrieval with reasoning, they still face important challenges. First, existing retrieval processes may produce ambiguous execution trajectories because they lack a global task plan and explicit sub-task structure. This can cause reasoning loops (e.g., repetitive keyword queries) that prevent the system from reaching a final result. Second, over-reliance on end-to-end training often leads to unclear optimization objectives under error accumulation, and sparse rewards further re-

duce learning efficiency. As a result, these issues may lead to inaccurate retrieval results and lower RAG performance. We argue that these symptoms come from *hierarchical credit entanglement*. The joint policy $\pi_\theta = (\pi_{\text{plan}}, \pi_{\text{exec}})$ is updated by a single sparse final reward $R(Q, A)$, so the policy gradient cannot tell whether an error comes from planning or execution. This unclear credit assignment can hurt both stages and lead to the looping and stagnation observed in end-to-end agentic RAG.

In this paper, we propose **APEX-Searcher**, a framework that improves LLM search through **A**gentic **P**lanning and **EX**ecution, following a *Refining Credit Assignment* (RCA) paradigm. Specifically, we train the planning module with RL using a Set-Equivalence Plan Reward (SEPR), because planning requires reasoning over valid task decompositions. We train the execution module with SFT, because sub-task execution has clearer supervision and more regular patterns, such as query generation and information extraction. Extensive experiments show that our method improves complex retrieval problem-solving, as well as agentic planning and execution.

The main contributions of our work are summarized as follows:

- We propose APEX-Searcher, a novel RAG framework that integrates agentic planning and execution to mitigate hierarchical credit entanglement between the planner and executor.
- We introduce a *Refining Credit Assignment* training strategy, where a Set-Equivalence Plan Reward provides an explicit plan-level signal to clarify the learning objective and improve learning efficiency.
- Extensive experimental results on multiple challenging multi-hop QA benchmarks show that APEX-Searcher improves complex problem solving, confirming the importance of planning in retrieval-augmented models.

2 Methodology

In this section, we present APEX-Searcher (**A**gentic **P**lanning & **EX**ecution **Searcher**), a novel framework designed to solve complex, multi-hop questions. At its core, our approach refines reasoning into two specialized phases as shown in Figure 1:

- Agentic Planning: Responsible for decomposing complex queries into strategic sub-goals.
- Iterative Sub-Task Execution: Responsible for interacting with external knowledge bases to retrieve and synthesize information.

Based on the sub-answers of each subtask, the framework comprehensively provides the answer to the original multi-hop complex task.

Unlike relying on generic prompting, we introduce a hybrid training framework to specialize these phases, employing Group Relative Policy Optimization (GRPO) (Shao et al., 2024) for strategic planning (Section 2.2) and a specialized SFT curriculum to master iterative execution (Section 2.3). Finally, we detail the systematic Inference Pipeline where these trained agents collaborate to solve user queries (Section 2.4).

2.1 Why Refine: Matching Training Signals to Stage Skills

End-to-end agentic RAG trains the joint policy $\pi_\theta = (\pi_{\text{plan}}, \pi_{\text{exec}})$ with a single final reward $R(Q, A)$. As a result, a wrong final answer caused by faulty execution may still suppress a correct plan, and vice versa. Beyond this credit-entanglement issue, the two stages require different learning signals, which motivates training them with separate objectives.

Planning is an open-ended reasoning task: a complex query can have many semantically equivalent decompositions, and there is no unique gold plan to imitate. RL is suitable here because it can explore different valid decompositions and reward plans that capture the required reasoning structure. We therefore train π_{plan} with a dedicated plan-level reward (SEPR, Section 2.2) that is robust to step ordering and paraphrasing, instead of relying only on the final answer reward.

Execution, in contrast, has weaker reasoning demands but stronger structural patterns. Once the plan is fixed, each sub-task – query generation, knowledge sufficiency check, and information extraction – follows relatively regular templates with limited branching. High-quality trajectories can also be obtained via self-instruct. SFT is therefore a better fit for this stage, since it directly learns these patterns from supervised trajectories, while RL would add unnecessary variance.

2.2 RL-based Agentic Planning

The initial phase of our methodology focuses on decomposing a complex, multi-hop question, Q, into a coherent and solvable execution plan, denoted as $S = \{s_1, s_2, ..., s_n\}$. We frame this de-

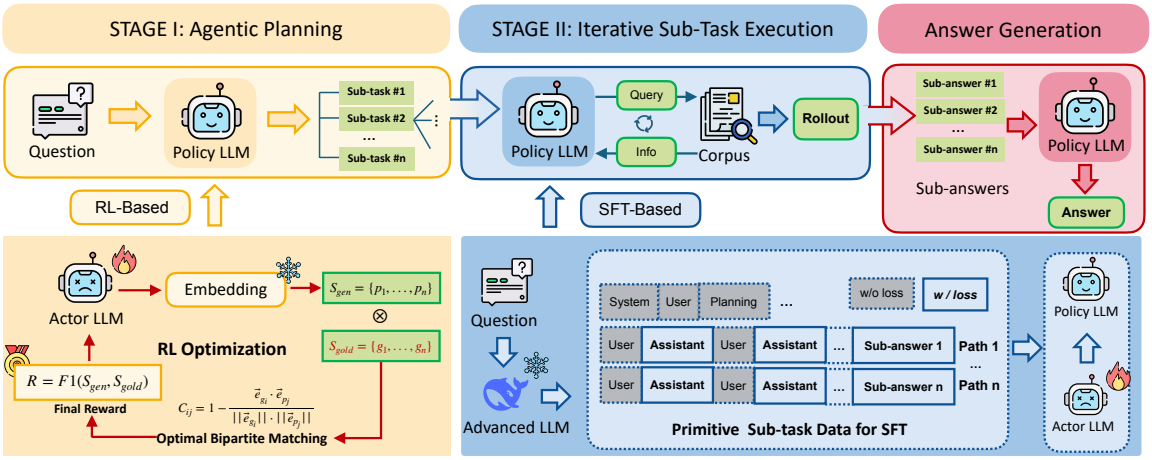


Figure 1: Overview of the APEX-Searcher framework. The architecture utilizes RL-driven agentic planning in Stage I to decompose a complex question into a multi-step plan. Subsequently, Stage II employs SFT-guided execution to solve each sub-question using an iterative retrieval loop that features dynamic continuation decisions and context management for final synthesis. See Figure 2 for an example.

composition task as a sequential decision-making problem and employ RL to train a "Planning Agent." This agent, implemented as a large language model, learns an optimal policy, π_{plan} for generating logical and efficient reasoning plans.

2.2.1 Set-Equivalence Plan Reward (SEPR)

A robust reward function is critical for guiding the RL agent toward generating high-quality plans. A complex query can be decomposed in *many* valid ways, so directly imitating a single gold plan would over-constrain π_{plan}; SEPR instead provides a dense, set-aware reward that scores plans up to ordering and paraphrasing. The reward, R_{plan}, is a final reward granted at the end of a full decomposition episode. It is calculated by comparing the agent-generated plan, S_{gen} against a human-annotated, gold-standard decomposition, S_{gold}.The core of our reward function is an F1-score that measures the semantic alignment between the two plans.

The calculation involves three key steps:

Semantic Similarity Calculation. To account for linguistic variations, we first measure the semantic similarity between sub-questions. Each sub-question from $S_{gen} = \{p_1, ..., p_n\}$ and $S_{gold} = \{g_1, ..., g_m\}$ is encoded into a high-dimensional vector using a sentence-transformer model, chosen for its established performance on semantic textual similarity tasks to minimize embedding-based variance. The similarity between a predicted sub-question p_j and a ground-truth sub-question g_i is then computed using cosine similarity:

$$\text{sim}(g_i, p_j) = \frac{\vec{e}_{g_i} \cdot \vec{e}_{p_j}}{||\vec{e}_{g_i}|| \cdot ||\vec{e}_{p_j}||}$$

where $\vec{e}_{g_i}$ and $\vec{e}_{p_j}$ are the vector embeddings of the respective sub-questions.

Optimal Bipartite Matching. To establish the most accurate correspondence between sub- questions in S_{gen} and S_{gold}, we formulate it as an assignment problem. We construct a cost matrix C where each element $C_{ij} = 1 - \text{sim}(g_i, p_j)$. The Hungarian algorithm is then employed to find the one-to-one matching that minimizes the total cost, yielding a set of matched pairs, $\mathcal{M}$, where the similarity of each pair exceeds a predefined threshold $\tau = 0.8$. This matching strategy ensures the metric is robust to step ordering, while the threshold was empirically tuned to filter low-confidence matches without penalizing valid paraphrases.

F1-Score as the Final Reward. Based on the set of matched pairs $\mathcal{M}$, the generated set S_{gen}, and the gold set S_{gold}, the final reward signal R_{plan} is computed directly. We define the reward as the harmonic mean based on the cardinalities of these sets:

$$R_{plan} = \frac{2 \cdot |\mathcal{M}|}{|S_{gen}| + |S_{gold}|}$$

This formulation serves as the reward for the GRPO algorithm, guiding the Planning Agent to learn a policy that produces logically sound, complete, and efficient reasoning plans. SEPR provides an explicit plan-level signal, which reduces

the dependence of planning learning on the executor's terminal reward.

2.2.2 Training Prompt Template

To guide the Planning Agent in generating well-structured and syntactically correct plans, we utilize a specialized training template. The agent is prompted with a set of instructions that define the expected output format and constraints. This structured prompting ensures that the agent's outputs are constrained to a manageable number of steps and explicitly models dependencies between sub-questions through the #n reference mechanism. The prompt used in the training process is as follows.

Prompt for Agentic Planning RL

```
You are an assistant who is good at
decomposing complex problems into simple
sub-problems.
1. Please decompose the problem directly
into 2-4 sub-problems, with each
sub-problem on a new line, separated by
'\n'. Do not add any serial numbers.
2. Use '#1', '#2', etc. to refer to the
answers of previous sub-problems, where
'#1' represents the answer to the first
sub-problem, for example, 'What city is #1
from?' represents asking which country the
answer to the first sub-question comes from.
3. Generally, the answer to the last
sub-problem should be the final answer to
this complex question.
The question you need to break down is:
```

2.3 SFT-based Agentic Execution

To improve multi-round retrieval, we construct multi-round fine-tuning data through Self-instruct (Wang et al., 2022) and filter it for answer correctness and consistency with the APEX-Searcher reasoning process. We then train the Execution Agent on these data via SFT, so the model can learn how to solve sub-tasks under the proposed execution format.

The data generation pipeline involved the following steps:

Seed Task Collection: We sampled and constructed a set of question instructions from the multi-hop task data training sets 2WikiMultiHopQA (Ho et al., 2020), HotpotQA (Yang et al., 2018), and MuSiQue (Trivedi et al., 2022).

Instruction Generation: We used Qwen2.5-32B-Instruct (Team, 2024) and Deepseek-v3 (Liu et al., 2024) to generate multi-round instructions based on the task plans, following the execution process shown in Figure 1.

Filtering and Validation: The generated data underwent a rigorous automated filtering process to ensure its quality and alignment with our framework's logic. We discarded instances that were trivial, contained flawed reasoning, or produced factually incorrect answers. This step was crucial for preventing the model from learning erroneous patterns.

Through this process, we sampled data from the 2WikiMultiHopQA, HotpotQA, and MuSiQue training sets to construct a final fine-tuning dataset of 14,604 high-quality, multi-turn retrieval instruction instances.

Fine-tuning on this dataset teaches the Execution Agent the iterative logic of APEX-Searcher. The model learns what information to retrieve, how to manage context across turns, and how to synthesize the retrieved evidence into accurate sub-task answers.

2.4 The APEX-Searcher Inference Pipeline

Problem Formulation. Given a complex multi-hop question Q and an external corpus C, our objective is to generate a final answer A_{final} that is faithful to C. We model this as a stateful, two-stage process:

- **Planning** decomposes Q into an ordered sequence of sub-questions $S = \{s_1, \ldots, s_n\}$, where a later s_j may depend on the answer a_i of an earlier s_i $(i < j)$ via placeholders.
- **Sequential Answering** solves each s_i by retrieving documents $D_i \subset C$ and synthesizing a sub-answer a_i, while maintaining an accumulated knowledge base $K_{\text{acc}} = \{(s_j, a_j)\}_{j \leq i}$ that supplies context for later steps.

The pipeline is illustrated in Figure 2; pseudo-code is given in Algorithm 1.

2.4.1 Phase 1: Agentic Planning

Upon receiving Q, the Planning Agent identifies implicit dependencies and decomposes Q into S. Sub-questions may carry placeholders (e.g., #1, #2) referencing earlier answers. If Q is simple or decomposition fails, the system falls back to treating Q as a single-step task.

2.4.2 Phase 2: Agentic Execution

Each $s_i \in S$ is processed sequentially by the Execution Agent. A *Reference Resolution* step first substitutes any placeholders in s_i with concrete

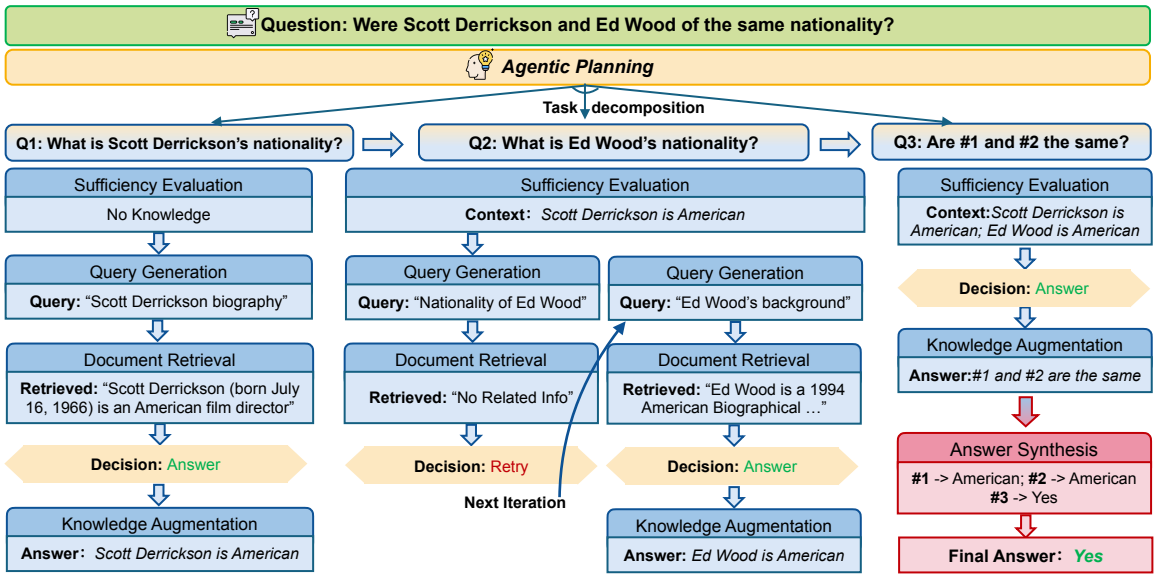


Figure 2: An example of two stage walkthrough of the APEX-Searcher pipeline. A complete walkthrough of the multi-hop reasoning trace is provided in Figure 6 (Appendix F)

Algorithm 1 APEX-Searcher Inference Pipeline

Require: Question Q; Planning Agent π_{plan}; Execution Agent π_{exec}; max hops $H_{\max}$
Ensure: Final answer A_{final}, confidence c

$S = \{s_1, \ldots, s_n\} \leftarrow \pi_{\text{plan}}(Q)$ ▷ Planning
$K_{\text{acc}} \leftarrow \emptyset$
for $i = 1$ to n **do**
  $s_i \leftarrow \textsc{ResolveRefs}(s_i, K_{\text{acc}})$
  $a_i \leftarrow \textsc{Execute}(s_i, K_{\text{acc}}, \pi_{\text{exec}}, H_{\max})$
  $K_{\text{acc}} \leftarrow K_{\text{acc}} \cup \{(s_i, a_i)\}$
end for
$(A_{\text{final}}, c) \leftarrow \pi_{\text{exec}}(Q \mid K_{\text{acc}})$ ▷ Synthesis
return (A_{final}, c)

answers from K_{acc}. The agent then runs the iterative loop below (Algorithm 2 in Appendix B). The complete set of prompts used by the Execution Agent (knowledge sufficiency check, query generation, and continuation decision) is listed in Appendix E.

(1) Knowledge Sufficiency Check. A decision function tests whether the current K_{acc} already supports answering s_i:

$$\delta_{\text{know}}(s_i, K_{\text{acc}}) \in \{\textsc{Answer}, \textsc{Retrieve}\}.$$

If ANSWER, the agent proceeds to sub-answer synthesis; otherwise it triggers the retrieval loop.

(2) Adaptive Multi-Hop Retrieval. Up to $H_{\max}$ hops are performed. At each hop h:

(a) Query Generation. The agent generates a novel query $q_{i,h}$ conditioned on K_{acc}, retrieved evidence, and prior queries $Q_{\text{hist}} = \{q_{i,1}, \ldots, q_{i,h-1}\}$:

$$q_{i,h} = \pi_{\text{exec}}(s_i \mid K_{\text{acc}}, Q_{\text{hist}}).$$

(b) Retrieval & De-duplication. $q_{i,h}$ is dispatched to the index over C; documents already retrieved in any prior hop are filtered out.

(c) Continuation Decision. For $h > 1$, a decision δ_{cont} judges whether the gathered evidence is complete. The loop terminates when $h{=}H_{\max}$ or $\delta_{\text{cont}}{=}\textsc{Stop}$.

(3) Knowledge Base Update. The new pair (s_i, a_i) is appended: $K_{\text{acc}} \leftarrow K_{\text{acc}} \cup \{(s_i, a_i)\}$, providing context for s_{i+1}.

2.4.3 Final Answer Synthesis

Once all sub-questions are answered, the Execution Agent synthesizes the final answer from the complete knowledge base:

$$A_{\text{final}} = \pi_{\text{exec}}(Q \mid K_{\text{acc}}).$$

A confidence score $c(A_{\text{final}}) \in [0, 1]$ is also produced, based on answer specificity, evidence completeness, and the absence of linguistic uncertainty markers.

3 Experiments

3.1 Experimental Setup

We evaluate APEX-Searcher on four multi-hop QA benchmarks: 2WikiMultiHopQA (Ho et al., 2020), HotpotQA (Yang et al., 2018), MuSiQue

Table 1: Main results comparison between Qwen2.5-7B-Instruct and Qwen2.5-3B-Instruct across different benchmarks. The highest score for each Benchmark is marked in bold.

Methods	Type	HotpotQA		2Wiki		MuSiQue		Bamboogle		Avg.	
		7B	3B	7B	3B	7B	3B	7B	3B	7B	3B
Direct Inference	Non-Retrieval	0.183	0.149	0.250	0.244	0.031	0.020	0.120	0.024	0.146	0.109
CoT	Non-Retrieval	0.092	0.021	0.111	0.021	0.022	0.002	0.232	0.000	0.114	0.011
RAG	Standard RAG	0.299	0.255	0.235	0.226	0.058	0.047	0.208	0.080	0.200	0.152
IRCoT	Iterative RAG	0.133	0.164	0.149	0.171	0.072	0.067	0.224	0.240	0.145	0.161
Search-o1	Agentic RAG	0.187	0.221	0.176	0.218	0.058	0.054	0.296	0.320	0.179	0.203
ZeroSearch-instruct	Agentic RAG	0.346	0.274	0.352	0.300	0.184	0.098	0.278	0.111	0.290	0.195
Search-R1-Instruct	Agentic RAG	0.370	0.284	0.414	0.273	0.146	0.049	0.368	0.088	0.324	0.174
StepSearch-instruct	Agentic RAG	0.386	0.345	0.366	0.320	**0.226**	**0.174**	0.400	0.328	0.345	0.296
ReasonRAG	Agentic RAG	0.384	-	0.436	-	0.128	-	0.360	-	0.327	-
DecEx-RAG	Agentic RAG	0.377	-	0.500	-	-	-	0.376	-	-	-
APEX-Searcher	Agentic RAG	**0.402**	**0.356**	**0.540**	**0.494**	0.164	0.136	**0.400**	**0.352**	**0.376**	**0.335**

(Trivedi et al., 2022), and Bamboogle (Press et al., 2023), using Exact Match (EM) as the primary metric. For training, we use 10,473 MuSiQue examples for RL-based planning and 14,604 multi-turn instruction instances sampled from the three training sets for SFT-based execution, with strict filtering to prevent train-test leakage.

We compare against four categories of baselines: (1) Non-Retrieval (Direct Inference, CoT (Wei et al., 2022)); (2) Standard RAG (Lewis et al., 2020); (3) Iterative RAG (IRCoT (Trivedi et al., 2023)); and (4) Agentic RAG (Search-o1 (Li et al., 2025), Search-r1 (Jin et al., 2025), ZeroSearch-instruct (Sun et al., 2025a), StepSearch-instruct (Wang et al., 2025b), ReasonRAG (Zhang et al., 2025a), DecEx-RAG (Leng et al., 2025)). We follow the same retrieval environment as (Jin et al., 2025; Wang et al., 2025b; Leng et al., 2025) for fair comparison. Full descriptions of benchmarks, training data, and baselines are provided in Appendix C.

3.2 Implementation Details

We conduct experiments on Qwen-2.5-3B-Instruct and Qwen-2.5-7B-Instruct (Team, 2024). RL-based planning is trained with GRPO via the verl[1] framework, and SFT-based execution is trained with the 360Llamafactory[2] framework. Detailed hyperparameters and training configurations are provided in Appendix D.

[1] https://github.com/volcengine/verl

[2] https://github.com/Qihoo360/360-LLaMA-Factory

3.3 Main Results

Table 1 compares the main experimental results. APEX-Searcher outperforms the baselines on most settings. Compared with Standard RAG on Qwen-2.5-7B-Instruct and Qwen-2.5-3B-Instruct, it improves average EM by **0.176** and **0.183**, respectively. Compared with the strongest baseline, APEX-Searcher also gives relative EM gains of +9.0% on 7B and +13.2% on 3B.

3.4 Ablation Study

To verify the effectiveness of RL-based Agentic Planning and SFT-based Agentic Execution, we ablate each component on Qwen2.5-7B-Instruct and Qwen2.5-3B-Instruct by toggling whether to use Planning, whether to train it with RL, and whether to apply SFT for Execution. Table 2 shows that the best performance is obtained when both planning and execution are trained with their corresponding signals.

Synergy of Components Enabling all components yields the best results across both scales: the 7B model improves from a baseline of 27.55 to 37.64, and the 3B model from 13.42 to 33.45. This result supports the main RCA design: planning benefits from SEPR+RL, while execution benefits from SFT under resolved plans.

Planning Alone. Adding Planning without RL or SFT brings limited gains and can even hurt smaller models. The 7B model rises modestly from 27.55 to 30.39, whereas the 3B model drops from 13.42 to 12.63. This suggests that an untrained planner can introduce noisy plan-level decisions, and these errors may propagate to the executor.

Table 2: Ablation study of different components on 7B and 3B models. **Plan**: Planning module; **RL**: Planning with RL; **Exec SFT**: Execution via Supervised Fine-Tuning.

Model	Method (Components)			HotpotQA	2Wiki	MuSiQue	Bamboogle
	Plan	Plan RL	Exec SFT				
Qwen-2.5-7B-Instruct	✗	-	✗	35.11	31.27	8.61	35.20
	✗	-	✓	35.18	35.69	9.56	36.00
	✓	✗	✗	33.52	42.92	11.50	33.60
	✓	✓	✗	36.79	46.30	12.99	39.20
	✓	✗	✓	35.76	48.72	12.12	37.60
	✓	✓	✓	**40.15**	**54.04**	**16.38**	**40.00**
Qwen-2.5-3B-Instruct	✗	-	✗	19.88	12.65	3.56	17.60
	✗	-	✓	29.97	30.58	7.12	26.40
	✓	✗	✗	16.73	14.73	3.06	16.00
	✓	✓	✗	20.58	25.00	6.00	20.80
	✓	✗	✓	26.77	33.43	7.74	28.80
	✓	✓	✓	**35.64**	**49.38**	**13.57**	**35.20**

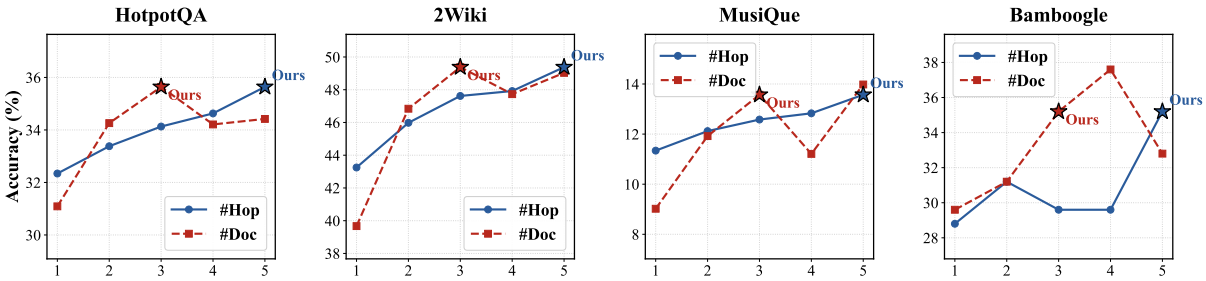


Figure 3: Parameter Sensitivity Analysis on APEX-Searcher. The curves illustrate the impact of the number of retrieved documents (#Doc) and the maximum allowed reasoning hops (#Hop) on model accuracy across four benchmarks. The asterisk ($\star$) denotes the optimal parameter configuration selected for this study and its corresponding performance.

Effect of Planning RL Adding RL on top of Planning markedly improves complex-problem solving: the 7B model rises from 30.39 to 33.82 and the 3B model from 12.63 to 18.10. This indicates that SEPR provides a useful plan-level training signal that transfers across benchmarks.

Effect of Execution SFT SFT is critical for subproblem retrieval and reasoning, especially for smaller models. Without Planning, enabling Exec SFT alone lifts the 3B model from 13.42 to 23.52, showing that supervised execution training already provides a large gain. The best results are obtained when this is combined with RL-based planning.

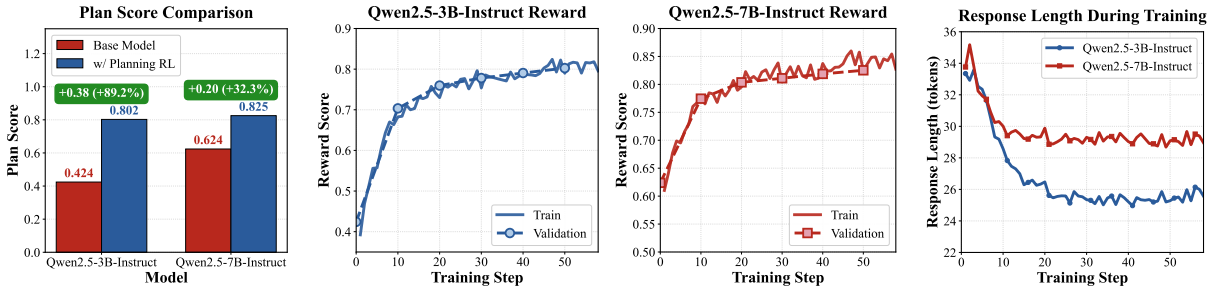
3.5 Parameter Sensitivity Analysis

We analyze the impact of two key parameters used in our experiments: the number of retrieved documents (Num) and the maximum allowed reasoning hops (Hop). The experiments are conducted on Qwen2.5-3B-Instruct. Figure 3 presents the model's accuracy across four benchmarks under varying parameter settings.

Based on the experimental results, we draw the following key observations:

1) **Diminishing returns or noise interference in document retrieval:** Across all datasets, accuracy significantly improves as the number of retrieved documents increases from 1 to 3, indicating that augmenting context is crucial for problem-solving. However, as the number exceeds 3, performance gains tend to plateau or even exhibit a distinct decline. This suggests that an excessive number of documents may introduce irrelevant information or noise, which disrupts the model's reasoning process and degrades performance.

2) **Positive correlation between reasoning hops and performance:** Generally, the model's accuracy demonstrates a continuous improvement as the maximum allowed reasoning hops increase from 1 to 5.



(a) (b) (c) (d)

Figure 4: (a) shows the Plan score improvements over base models, (b-c) show the reward score convergence during training for the 3B and 7B variants, and (d) shows the optimization of response length over training steps.

3.6 Analysis on Task Planning

To analyze the effect of planning-oriented learning, we evaluate task-decomposition reward scores on the test set for Qwen2.5-3B-Instruct and Qwen2.5-7B-Instruct after agentic planning RL. As shown in Figure 4a, planning RL substantially improves decomposition accuracy for both models. This improvement also translates into higher downstream accuracy on multi-hop QA, indicating that better task planning provides more reliable guidance for subsequent retrieval and reasoning.

We further track Training Reward, Validation Reward, and Mean Response Length throughout training, as shown in Figures 4b, 4c, and 4d. Both training and validation rewards increase steadily with consistent trends, suggesting stable optimization and effective generalization. Meanwhile, the mean response length decreases as training progresses. This trend indicates that the model gradually learns to avoid over-decomposition and redundant planning, producing more concise and accurate sub-task sequences.

For qualitative evaluation, we compare the decomposition strategies generated by our model with those of the Qwen2.5-7B-Instruct baseline. As shown in Table 3, our method decomposes questions into accurate sub-tasks without introducing redundant steps. It also uses placeholders such as #n to represent intermediate answers, rather than assuming or leaking those answers during planning. More decomposition examples across diverse question types are provided in Appendix F.

Table 3: Case Study Comparison of Question Decomposition

Case 1: Genealogy	
Question	Who is the maternal grandfather of Antiochus X Eusebes?
Baseline	• Who is the mother of Antiochus X Eusebes? • Who is the father of Antiochus X Eusebes's mother? • What is the relationship of the answer to #2 to Antiochus X Eusebes?
Ours	- Who is the mother of Antiochus X Eusebes? - Who is the father of #1? (Who is the father of Cleopatra IV?)

4 Conclusion

In this paper, we study how to improve RAG for complex multi-hop questions, where single-round retrieval is often insufficient and end-to-end agentic RAG can suffer from ambiguous execution trajectories and sparse training signals. We attribute these issues to *hierarchical credit entanglement*. To address this problem, we propose APEX-Searcher, which follows a Refining Credit Assignment paradigm. The model's planning ability is optimized with RL using SEPR to learn valid task decompositions, while its execution ability is trained with SFT to carry out each sub-task through multi-round retrieval. Experiments across multiple multi-hop QA benchmarks show consistent gains in both answer accuracy and task planning, and ablations confirm that separating the training signals for planning and execution is important. These results suggest that exposing intermediate reasoning structure and assigning credit at the appropriate stage can make agentic RAG systems more effective for complex information-seeking tasks.

Limitations

Dataset coverage. Our experiments are conducted mainly on English multi-hop QA benchmarks built from Wikipedia-style corpora. These datasets provide a controlled and widely used testbed for complex retrieval, but they do not fully cover domain-specific settings such as biomedical, legal, financial, or rapidly changing web information. We plan to extend the evaluation to more diverse domains and live retrieval environments in future work. This limitation concerns the breadth of empirical validation rather than the design of the proposed framework.

Backbone and comparison coverage. We instantiate APEX-Searcher with a limited set of open-source base models and compare it with representative RAG, iterative RAG, and agentic RAG baselines under a shared retrieval environment. Although this setting supports fair comparison with prior work, it leaves open how the results scale across larger backbones, different model families, stronger commercial models, and additional recent agentic retrieval systems. Future work will evaluate a broader range of base models and baselines, as well as report more extensive sensitivity analyses over retrieval and decoding configurations. These additions would strengthen the empirical picture but do not change the core method design.

Scope. Experiments focus on multi-hop QA over Wikipedia-style corpora; extending RCA to open-web tool-use, code agents, and multi-modal retrieval is left for future study.

Use of AI Assistants

We used general-purpose AI writing assistants (e.g., ChatGPT and Claude) in a limited and supervised capacity during the preparation of this manuscript. Their use was restricted to (i) language polishing of English prose written by the authors, including grammar, phrasing, and minor stylistic suggestions, and (ii) translation assistance between Chinese and English drafts. All scientific contributions, including the problem formulation, the design of the Refining Credit Assignment paradigm, the SEPR reward, the experimental protocol, and the analysis of results, were conceived, executed, and verified by the human authors. The AI assistants were not used to generate experimental results, write training or evaluation code, fabricate citations, or produce novel claims, and all assistant-suggested text was reviewed and edited by the authors prior to inclusion. The authors take full responsibility for the final content of the paper.

References

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

Mingyang Chen, Linzhuang Sun, Tianpeng Li, Haoze Sun, Yijie Zhou, Chenzheng Zhu, Haofen Wang, Jeff Z Pan, Wen Zhang, Huajun Chen, and 1 others. 2025. Learning to reason with search for llms via reinforcement learning. *arXiv preprint arXiv:2503.19470*.

Manuel Cossio. 2025. A comprehensive taxonomy of hallucinations in large language models. *arXiv preprint arXiv:2508.01781*.

Lutfi Eren Erdogan, Nicholas Lee, Sehoon Kim, Suhong Moon, Hiroki Furuta, Gopala Anumanchipalli, Kurt Keutzer, and Amir Gholami. 2025. Plan-and-act: Improving planning of agents for long-horizon tasks. *arXiv preprint arXiv:2503.09572*.

Jinyuan Fang, Zaiqiao Meng, and Craig MacDonald. 2025. KiRAG: Knowledge-driven iterative retriever for enhancing retrieval-augmented generation. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 18969–18985, Vienna, Austria. Association for Computational Linguistics.

Yunfan Gao, Yun Xiong, Xinyu Gao, Kangxiang Jia, Jinliu Pan, Yuxi Bi, Yixin Dai, Jiawei Sun, Haofen Wang, and Haofen Wang. 2023. Retrieval-augmented generation for large language models: A survey. *arXiv preprint arXiv:2312.10997*, 2(1).

Yu Gu, Boyuan Zheng, Boyu Gou, Kai Zhang, Cheng Chang, Sanjari Srivastava, Yanan Xie, Peng Qi, Huan Sun, and Yu Su. 2024. Simulate before act: Model-based planning for web agents.

Xanh Ho, Anh-Khoa Duong Nguyen, Saku Sugawara, and Akiko Aizawa. 2020. Constructing a multi-hop qa dataset for comprehensive evaluation of reasoning steps. *arXiv preprint arXiv:2011.01060*.

Lei Huang, Weijiang Yu, Weitao Ma, Weihong Zhong, Zhangyin Feng, Haotian Wang, Qianglong Chen, Weihua Peng, Xiaocheng Feng, Bing Qin, and 1 others. 2025. A survey on hallucination in large language models: Principles, taxonomy, challenges, and open questions. *ACM Transactions on Information Systems*, 43(2):1–55.

Zhengbao Jiang, Frank F Xu, Luyu Gao, Zhiqing Sun, Qian Liu, Jane Dwivedi-Yu, Yiming Yang, Jamie Callan, and Graham Neubig. 2023. Active retrieval augmented generation. In *Proceedings of the 2023*

Conference on Empirical Methods in Natural Language Processing, pages 7969–7992.

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

Mandar Joshi, Eunsol Choi, Daniel Weld, and Luke Zettlemoyer. 2017. TriviaQA: A large scale distantly supervised challenge dataset for reading comprehension. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1601–1611.

Nikhil Kandpal, Haikang Deng, Adam Roberts, Eric Wallace, and Colin Raffel. 2023. Large language models struggle to learn long-tail knowledge. In *Proceedings of the 40th International Conference on Machine Learning*, volume 202 of *Proceedings of Machine Learning Research*, pages 15696–15707. PMLR.

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

Michael Katz, Harsha Kokel, Kavitha Srinivas, and Shirin Sohrabi Araghi. 2024. Thought of search: Planning with language models through the lens of efficiency. *Advances in Neural Information Processing Systems*, 37:138491–138568.

Tom Kwiatkowski, Jennimaria Palomaki, Olivia Redfield, Michael Collins, Ankur Parikh, Chris Alberti, Danielle Epstein, Illia Polosukhin, Jacob Devlin, Kenton Lee, and 1 others. 2019. Natural questions: a benchmark for question answering research. *Transactions of the Association for Computational Linguistics*, 7:453–466.

Jakub Lála, Odhran O'Donoghue, Aleksandar Shtedritski, Sam Cox, Samuel G Rodriques, and Andrew D White. 2023. Paperqa: Retrieval-augmented generative agent for scientific research. *arXiv preprint arXiv:2312.07559*.

Yongqi Leng, Yikun Lei, Xikai Liu, Meizhi Zhong, Bojian Xiong, Yurong Zhang, Yan Gao, Yao Hu, Deyi Xiong, and 1 others. 2025. Decex-rag: Boosting agentic retrieval-augmented generation with decision and execution optimization via process supervision. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing: Industry Track*, pages 1412–1425.

Patrick Lewis, Ethan Perez, Aleksandra Piktus, Fabio Petroni, Vladimir Karpukhin, Naman Goyal, Heinrich Küttler, Mike Lewis, Wen-tau Yih, Tim Rocktäschel, and 1 others. 2020. Retrieval-augmented generation for knowledge-intensive nlp tasks. *Advances in neural information processing systems*, 33:9459–9474.

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

Aixin Liu, Bei Feng, Bing Xue, Bingxuan Wang, Bochao Wu, Chengda Lu, Chenggang Zhao, Chengqi Deng, Chenyu Zhang, Chong Ruan, and 1 others. 2024. Deepseek-v3 technical report. *arXiv preprint arXiv:2412.19437*.

Vaibhav Mavi, Anubhav Jangra, Adam Jatowt, and 1 others. 2024. Multi-hop question answering. *Foundations and Trends® in Information Retrieval*, 17(5):457–586.

Ajay Patel, Markus Hofmarcher, Claudiu Leoveanu-Condrei, Marius-Constantin Dinu, Chris Callison-Burch, and Sepp Hochreiter. 2024. Large language models can self-improve at web agent tasks. *arXiv preprint arXiv:2405.20309*.

Aske Plaat, Max van Duijn, Niki van Stein, Mike Preuss, Peter van der Putten, and Kees Joost Batenburg. 2025. Agentic large language models, a survey. *arXiv preprint arXiv:2503.23037*.

Ofir Press, Muru Zhang, Sewon Min, Ludwig Schmidt, Noah A. Smith, and Mike Lewis. 2023. Measuring and narrowing the compositionality gap in language models. *Preprint*, arXiv:2210.03350.

Adam Roberts, Colin Raffel, and Noam Shazeer. 2020. How much knowledge can you pack into the parameters of a language model? *arXiv preprint arXiv:2002.08910*.

Zhihong Shao, Yeyun Gong, Yelong Shen, Minlie Huang, Nan Duan, and Weizhu Chen. 2023. Enhancing retrieval-augmented large language models with iterative retrieval-generation synergy. *arXiv preprint arXiv:2305.15294*.

Zhihong Shao, Peiyi Wang, Qihao Zhu, Runxin Xu, Junxiao Song, Xiao Bi, Haowei Zhang, Mingchuan Zhang, YK Li, Yang Wu, and 1 others. 2024. Deepseekmath: Pushing the limits of mathematical reasoning in open language models. *arXiv preprint arXiv:2402.03300*.

Freda Shi, Xinyun Chen, Kanishka Misra, Nathan Scales, David Dohan, Ed H Chi, Nathanael Schärli, and Denny Zhou. 2023. Large language models can be easily distracted by irrelevant context. In *International Conference on Machine Learning*, pages 31210–31227. PMLR.

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

Hao Sun, Zile Qiao, Jiayan Guo, Xuanbo Fan, Yingyan Hou, Yong Jiang, Pengjun Xie, Yan Zhang, Fei Huang, and Jingren Zhou. 2025a. Zerosearch: Incentivize the search capability of llms without searching. *arXiv preprint arXiv:2505.04588*.

Shuang Sun, Huatong Song, Yuhao Wang, Ruiyang Ren, Jinhao Jiang, Junjie Zhang, Fei Bai, Jia Deng, Wayne Xin Zhao, Zheng Liu, and 1 others. 2025b. Simpledeepsearcher: Deep information seeking via web-powered reasoning trajectory synthesis. *arXiv preprint arXiv:2505.16834*.

Qwen Team. 2024. Qwen2.5: A party of foundation models.

Harsh Trivedi, Niranjan Balasubramanian, Tushar Khot, and Ashish Sabharwal. 2022. Musique: Multihop questions via single-hop question composition. *Transactions of the Association for Computational Linguistics*, 10:539–554.

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

Yizhong Wang, Yeganeh Kordi, Swaroop Mishra, Alisa Liu, Noah A Smith, Daniel Khashabi, and Hannaneh Hajishirzi. 2022. Self-instruct: Aligning language models with self-generated instructions. *arXiv preprint arXiv:2212.10560*.

Zheng Wang, Shu Xian Teo, Jun Jie Chew, and Wei Shi. 2025a. Instructrag: Leveraging retrieval-augmented generation on instruction graphs for llm-based task planning. In *Proceedings of the 48th International ACM SIGIR Conference on Research and Development in Information Retrieval*, page 14131422.

Ziliang Wang, Xuhui Zheng, Kang An, Cijun Ouyang, Jialu Cai, Yuhang Wang, and Yichao Wu. 2025b. Stepsearch: Igniting llms search ability via stepwise proximal policy optimization. *arXiv preprint arXiv:2505.15107*.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Fei Xia, Ed Chi, Quoc V Le, Denny Zhou, and 1 others. 2022. Chain-of-thought prompting elicits reasoning in large language models. *Advances in neural information processing systems*, 35:24824–24837.

Renjun Xu and Jingwen Peng. 2025. A comprehensive survey of deep research: Systems, methodologies, and applications. *arXiv preprint arXiv:2506.12594*.

Shicheng Xu, Liang Pang, Huawei Shen, Xueqi Cheng, and Tat-Seng Chua. 2024. Search-in-the-chain: Interactively enhancing large language models with search for knowledge-intensive tasks. In *Proceedings of the ACM Web Conference 2024*, pages 1362–1373.

Ruiyi Yang, Hao Xue, Imran Razzak, Hakim Hacid, and Flora D Salim. 2025. Beyond single pass, looping through time: Kg-irag with iterative knowledge retrieval. *arXiv preprint arXiv:2503.14234*.

Zhilin Yang, Peng Qi, Saizheng Zhang, Yoshua Bengio, William W Cohen, Ruslan Salakhutdinov, and Christopher D Manning. 2018. Hotpotqa: A dataset for diverse, explainable multi-hop question answering. *arXiv preprint arXiv:1809.09600*.

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

Wenhao Yu, Hongming Zhang, Xiaoman Pan, Kaixin Ma, Hongwei Wang, and Dong Yu. 2023. Chain-of-note: Enhancing robustness in retrieval-augmented language models. *arXiv preprint arXiv:2311.09210*.

Qin Zhang, Shangsi Chen, Dongkuan Xu, Qingqing Cao, Xiaojun Chen, Trevor Cohn, and Meng Fang. 2022. A survey for efficient open domain question answering. *arXiv preprint arXiv:2211.07886*.

Wenlin Zhang, Xiangyang Li, Kuicai Dong, Yichao Wang, Pengyue Jia, Xiaopeng Li, Yingyi Zhang, Derong Xu, Zhaocheng Du, Huifeng Guo, and 1 others. 2025a. Process vs. outcome reward: Which is better for agentic rag reinforcement learning. *arXiv preprint arXiv:2505.14069*.

Wenlin Zhang, Xiaopeng Li, Yingyi Zhang, Pengyue Jia, Yichao Wang, Huifeng Guo, Yong Liu, and Xiangyu Zhao. 2025b. Deep research: A survey of autonomous research agents. *arXiv preprint arXiv:2508.12752*.

Wayne Xin Zhao, Kun Zhou, Junyi Li, Tianyi Tang, Xiaolei Wang, Yupeng Hou, Yingqian Min, Beichen Zhang, Junjie Zhang, Zican Dong, and 1 others. 2023. A survey of large language models. *arXiv preprint arXiv:2303.18223*, 1(2).

Yuxiang Zheng, Dayuan Fu, Xiangkun Hu, Xiaojie Cai, Lyumanshan Ye, Pengrui Lu, and Pengfei Liu. 2025. Deepresearcher: Scaling deep research via reinforcement learning in real-world environments. *arXiv preprint arXiv:2504.03160*.

Appendix

A Related Works

A.1 Standard Retrieval-Augmented Generation

Standard RAG, often referred to as "Naive RAG" in recent literature (Gao et al., 2023), emerged as a foundational paradigm to address critical limitations of LLMs, such as hallucinations (Huang et al., 2025; Cossio, 2025), outdated knowledge, and knowledge-intensive tasks (Kandpal et al., 2023). Rooted in the integration of external knowledge bases with LLMs, this framework follows a three-stage pipeline: indexing, retrieval, and generation (Lewis et al., 2020; Karpukhin et al., 2020). Early implementations of traditional RAG focused on improving knowledge-intensive tasks such as open-domain question answering (Zhang et al., 2022). For instance, (Lewis et al., 2020) demonstrated that RAG outperforms vanilla LLMs on ODQA benchmarks (e.g., Natural Questions (Kwiatkowski et al., 2019), TriviaQA (Joshi et al., 2017)) by leveraging Wikipedia as an external knowledge source. Similarly, Dense Passage Retrieval (Karpukhin et al., 2020) introduced dense retrievers to improve retrieval precision, laying the foundation for subsequent RAG advances. Despite its success, traditional RAG faces notable limitations: retrieval often suffers from low precision/recall (e.g., retrieving irrelevant or redundant chunks), generation may produce hallucinations inconsistent with retrieved context (Gao et al., 2023; Yu et al., 2023; Shi et al., 2023).

A.2 Iterative Retrieval-Augmented Generation

Iterative Retrieval-Augmented Generation was proposed to overcome the limitations of traditional RAG's one-time retrieval, which often fails to provide sufficient context for complex, multi-step reasoning tasks (Shao et al., 2023). Iterative RAG adopts a cyclic pipeline: it repeatedly retrieves information from external knowledge bases based on the initial query and intermediate generation outputs, enabling the model to accumulate incremental context and refine its understanding of the task (Shao et al., 2023; Trivedi et al., 2023; Yang et al., 2025; Fang et al., 2025; Asai et al., 2024; Lála et al., 2023). Key contributions to Iterative RAG include frameworks that synergize retrieval and generation to enhance mutual performance. For example, ITER-RETGEN (Shao et al., 2023) introduces a "retrieval-enhanced generation" and "generation-enhanced retrieval" loop to ensure that each new chunk aligns with the evolving task context. IRCoT (Trivedi et al., 2023) is a new multi-step QA approach that uses chains of thought (CoT) to guide the retrieval and takes advantage of the retrieved results to improve CoT.

Evaluations on benchmarks like HotpotQA (Yang et al., 2018) (multi-hop QA) have shown that Iterative RAG outperforms traditional RAG by capturing nuanced, multi-step dependencies in information. However, challenges persist, including potential semantic discontinuity across iterations and accumulation of irrelevant information, which have spurred research into adaptive stopping criteria (e.g., confidence thresholds) to balance retrieval depth and efficiency (Jiang et al., 2023).

A.3 Agentic Retrieval-Augmented Generation

Recent surveys define an agentic RAG as a system that can autonomously reason, act, and interact with its environment to achieve a goal (Plaat et al., 2025). Unlike passive models that simply respond to prompts, an agentic system can decide when and how to use external tools (like a search engine or a retrieval engine), and adapt its strategy based on feedback (Yao et al., 2023; Press et al., 2023; Trivedi et al., 2023; Li et al., 2025; Xu et al., 2024). This paradigm shift is motivated by the need to address complex, long-horizon tasks that require more than a single inference pass. For example, Search-o1 (Li et al., 2025) framework extends the agentic RAG mechanism by incorporating a Reason-in-Documents module. Building on this foundation, some agentic RAG work has focused on improving the model's reasoning ability during the search process through RL (Jin et al., 2025; Chen et al., 2025; Song et al., 2025; Zheng et al., 2025; Sun et al., 2025a; Wang et al., 2025b; Zhang et al., 2025a). For example, Search-R1 (Jin et al., 2025) optimizes LLM reasoning paths through multi-round search interactions and achieves stable RL training with the help of retrieval token masking. Although current agentic RAG has demonstrated strong search capabilities, due to the lack of clear task planning before retrieval, phenomena such as task forgetting and repeated retrieval can occur.

Algorithm 2 Process Sub-question

Require: Sub-question s, Accumulated Knowledge K, Main Model M_m, Max Hops H_{max}
Ensure: Sub-question Answer a, Retrieved Context ctx

Step 1: Initial Knowledge Check
$sufficient \leftarrow M_m.\text{CheckKnowledgeSufficiency}(s, K)$
if $sufficient$ **then**
  $a \leftarrow M_m.\text{AnswerFromKnowledge}(s, K)$
  return $(a, \emptyset)$
end if
Step 2: Multi-Hop Retrieval Loop
$ctx \leftarrow \emptyset \quad docs_{seen} \leftarrow \emptyset \quad queries_{tried} \leftarrow \emptyset$
for $hop = 1$ to H_{max} **do**
  A. Continuation Decision (if $hop > 1$ and $hop < H_{max}$)
  if $\neg M_m.\text{ShouldContinueRetrieval}(s, ctx, K)$ **then**
    break
  end if
  B. Query Generation
  $query \leftarrow M_m.\text{GenerateQuery}(s, ctx, queries_{tried}, K)$
  $queries_{tried} \leftarrow queries_{tried} \cup \{query\}$
  C. Document Retrieval
  $docs_{new} \leftarrow \text{Retrieve}(query)$
  $docs_{new} \leftarrow \text{FilterDuplicates}(docs_{new}, docs_{seen})$
  if $|docs_{new}| = 0$ **then**
    break ▷ No new documents
  end if
  D. Context Management
  $ctx \leftarrow ctx \cup \text{Format}(docs_{new})$
  $docs_{seen} \leftarrow docs_{seen} \cup \text{GetIDs}(docs_{new})$
end for
Step 3: Sub-question Answer Generation
$a \leftarrow M_m.\text{GenerateAnswer}(s, ctx, K)$
return (a, ctx)

A.4 Planning in Agentic Systems

To enhance the planning ability of agents, a significant amount of research has utilized explicit or implicit structured knowledge to guide the planning process during the reasoning stage (Wang et al., 2025a; Gu et al., 2024; Erdogan et al., 2025; Katz et al., 2024). Many works have also focused on making planning ability a learning objective for the agent, enabling it to optimize its decision-making process through search, feedback, or large-scale training (Xu et al., 2024; Patel et al., 2024; Sun et al., 2025b). However, few studies have explored the integration of planning in complex multi-round retrieval. While the agentic RAG mentioned in Section A.3 has achieved remarkable performance, none of them have considered the agent's planning ability as a crucial aspect for improving retrieval accuracy. In fact, planning is a fundamental stage for an LLM-based agent before retrieval execution, and it is an important step in breaking down a complex retrieval problem into primitive sub-tasks (Zhang et al., 2025b; Xu and Peng, 2025).

B Process Sub-question Algorithm

Algorithm 2 details the inner iterative retrieval loop used by the Execution Agent to solve each sub-question. It performs an initial knowledge sufficiency check, and if retrieval is required, conducts up to $H_{\max}$ retrieval hops with adaptive continuation decisions, query-diversity enforcement, and document de-duplication.

C Detailed Experimental Setup

C.1 Evaluation Benchmarks and Datasets

We use four multi-hop question answering tasks as benchmarks, including 2WikiMultiHopQA (Ho et al., 2020), HotpotQA (Yang et al., 2018), MuSiQue (Trivedi et al., 2022), and Bamboogle (Press et al., 2023). The questions in these benchmarks are constructed by combining information from multiple Wikipedia articles. Therefore, a single retrieval is often unable to obtain all relevant documents, which puts higher demands on the ability of methods to solve complex tasks. We utilize Exact Match (EM) as our primary evaluation metric. Given that the target answers in this dataset are predominantly short, factoid entities, we find that F1 scores are highly correlated with EM and provide marginal additional signal. Consequently, we focus on EM to provide the strictest assessment of model performance. EM is computed with the standard normalization pipeline used in prior multi-hop QA work (Yang et al., 2018), namely lowercasing, removal of punctuation and articles, and whitespace normalization, after which a prediction counts as correct only if it matches the reference string exactly.

Domain, language, and intended use. All four benchmarks are English-language, factoid multi-hop QA datasets derived from publicly available Wikipedia articles, and were released by their authors for non-commercial research on multi-hop reasoning and retrieval-augmented language modeling. Our use is fully consistent with this intended purpose: we use them only for training, validation, and evaluation of a research prototype. Since the benchmarks are based on curated Wikipedia content, they are not expected to contain personally identifying information about private individuals or offensive material beyond what is already publicly documented in Wikipedia, and we did not perform any additional data collection from human subjects.

Dataset statistics. We follow the official splits released by each benchmark. Specifically, HotpotQA provides ∼90K training and ∼7.4K validation (distractor setting) examples; 2WikiMultiHopQA provides ∼167K training and ∼12.6K validation/test examples; MuSiQue (answerable subset) provides ∼20K training and ∼2.4K validation/test examples; and Bamboogle is a small test-only set of 125 manually constructed questions. For evaluation, we report EM on the development split of HotpotQA, 2WikiMultiHopQA, and MuSiQue (test answers are not publicly released) and on the full Bamboogle test set, matching the protocol used by prior agentic RAG work (Jin et al., 2025; Wang et al., 2025b; Leng et al., 2025).

For the RL-based Agentic Planning phase, we utilized 10,473 examples from the MuSiQue training set to enable the agent to learn robust task decomposition strategies from complex reasoning trajectories. In terms of data processing, we used DeepSeek-V3 (Liu et al., 2024) to rewrite the original planning of symbolic language for task decomposition into natural language. In the SFT-based Agentic Execution phase, we sampled training set data from 2WikiMultiHopQA, HotpotQA, and MuSiQue to construct 14,604 pieces of multi-round retrieval instruction data, and strict data filtering has been carried out to ensure that there is no leakage from the training set to the test set, thus guaranteeing the fairness of the evaluation.

C.2 Licenses and Terms for Artifacts

We used publicly released research artifacts and followed their stated licenses and terms. HotpotQA is distributed under CC BY-SA 4.0[3], 2WikiMultiHopQA under Apache-2.0[4], MuSiQue under CC BY 4.0[5], and Bamboogle under the MIT license[6]. Our use of these datasets is limited to research training, validation, and evaluation for multi-hop question answering. Any release of derived instruction data or model outputs should preserve attribution to the source datasets and comply with the corresponding share-alike or attribution requirements; in particular, derivatives of HotpotQA content should be distributed only under terms compatible with CC BY-SA 4.0.

For model artifacts, Qwen2.5-7B-Instruct is released under Apache-2.0[7], whereas Qwen2.5-3B-Instruct is released under the Qwen Research License, which permits non-commercial research use and requires a separate license for commercial use[8]. DeepSeek-V3 is subject to the DeepSeek

[3] https://hotpotqa.github.io/
[4] https://github.com/Alab-NII/2wikimultihop
[5] https://github.com/StonyBrookNLP/musique
[6] https://huggingface.co/datasets/chiayewken/bamboogle
[7] https://huggingface.co/Qwen/Qwen2.5-7B-Instruct
[8] https://huggingface.co/Qwen/Qwen2.5-3B-Instruct/blob/main/LICENSE

model license[9]; we used it only to generate intermediate rewritten planning text for research data construction. For software artifacts, verl and 360-LLaMA-Factory are released under Apache-2.0[10][11], DeepSpeed under MIT[12], and FlashAttention under BSD-3-Clause[13]. We did not redistribute the original upstream datasets, model checkpoints, or third-party software in this submission.

C.3 Baselines

We compare our method against the following baselines: (1) Non-Retrieval Methods: Direct Inference and CoT reasoning (Wei et al., 2022). (2) Standard RAG: A standard Retrieval-Augmented Generation (Lewis et al., 2020). (3) Iterative RAG: IRCoT (Trivedi et al., 2023). (4) Agentic RAG: Search-o1 (Li et al., 2025), Search-r1 (Jin et al., 2025), ZeroSearch-instruct (Sun et al., 2025a), StepSearch-instruct (Wang et al., 2025b), Reason-RAG (Zhang et al., 2025a) and DecEx-RAG (Leng et al., 2025).

We use the same environment as most of the compared baselines (Jin et al., 2025; Wang et al., 2025b; Leng et al., 2025), including the retrieval corpus, the number of retrieved documents, and the construction scheme of the retrieval environment to ensure the fairness of the evaluation.

D Implementation Details

We conducted experiments on Qwen-2.5-3B-Instruct (3.09B parameters) and Qwen-2.5-7B-Instruct (7.62B parameters) (Team, 2024) respectively.

D.1 Computing Infrastructure and Budget

All training and inference experiments are performed on a single node with 8×NVIDIA A100 80GB GPUs. The dominant cost is RL-based Agentic Planning on the 7B model, which takes approximately 7 GPU-hours per run on this node; SFT-based Agentic Execution on the 7B model takes approximately 5 GPU-hours per run. Training on the 3B model is correspondingly cheaper. Aggregating both stages and both model scales (including a small number of preliminary runs used for hyperparameter selection), the total computational budget for all reported experiments is on the order of a few hundred A100-hours, which we consider modest for a paper of this scope.

D.2 Inference and Reporting Protocol

For all main and ablation results, decoding is fully deterministic: we use greedy decoding (temperature $= 0$, no sampling) for both the Planning Agent and the Execution Agent, and we fix the random seed for data ordering and model initialization. As a consequence, the EM scores reported in Tables 1 and 2 correspond to a single, reproducible run per configuration rather than an average over multiple sampled runs, and we therefore do not report error bars; the reported numbers should be interpreted as deterministic point estimates of model performance under our chosen decoding configuration.

D.3 RL-based Agentic Planning

We utilized the verl[14] framework to implement a GRPO algorithm for 3 epochs. Key hyperparameters for the training process were configured as follows: The learning rate was set to 5×10^{-6}. We used a global training batch size of 512, with a PPO mini-batch size of 128 and a micro-batch size of 16 per GPU. The maximum prompt and response lengths were both capped at 1024 tokens. To regularize the policy updates and prevent divergence from the reference model, we incorporated a KL divergence loss with a coefficient of 0.01. Furthermore, an entropy coefficient of 0.01 was applied to encourage exploration. To optimize for memory and computational efficiency, the training was conducted with bfloat16 precision. Gradient checkpointing was enabled to reduce memory consumption. The GRPO algorithm generates 8-tone trajectories in one group, with gradient retention clipping $\epsilon = 0.2$.

D.4 SFT-based Agentic Execution

We performed full-parameter SFT using the 360Llamafactory[15] framework. The model was trained for 2 epochs. The training was configured with a learning rate of 5×10^{-6} and a cosine learning rate scheduler, with a warmup ratio of 0.03. We set a per-device training batch size of 1 and

[9] https://github.com/deepseek-ai/DeepSeek-V3/blob/main/LICENSE-MODEL
[10] https://github.com/volcengine/verl
[11] https://github.com/Qihoo360/360-LLaMA-Factory
[12] https://github.com/microsoft/DeepSpeed
[13] https://github.com/Dao-AILab/flash-attention
[14] https://github.com/volcengine/verl
[15] https://github.com/Qihoo360/360-LLaMA-Factory

used 2 gradient accumulation steps. The maximum sequence length was set to 32,768 tokens.

Several optimization techniques were employed to ensure efficient training. We utilized DeepSpeed[16] with a ZeRO stage 3 configuration to optimize memory usage across devices. The training was performed using bfloat16 precision, and Flash Attention 2[17] was enabled to accelerate the attention mechanism computations. Gradient checkpointing was also activated to further conserve memory. The model was trained with a sequence parallel size of 2.

E Prompts

Adaptive Retrieval Decision. This prompt enables adaptive retrieval by determining whether the accumulated knowledge from previous reasoning steps is sufficient to answer the current subquestion. The model analyzes the available information and decides between two actions: retrieve new information or answer directly. If sufficient knowledge exists, the system can skip retrieval and directly generate an answer, significantly improving computational efficiency and reducing unnecessary API calls.

Adaptive Retrieval Decision Prompt

```
Based on accumulated knowledge, can you
answer the subtask without retrieval?

Current Subtask: {subtask}
Accumulated Knowledge: {knowledge}

Analyze if you have sufficient information
to answer the subtask.
Respond with:
<think>your analysis of whether current
knowledge is sufficient</think>
<decision>retrieve</decision> or <decision>
answer</decision>
```

Answer from Accumulated Knowledge. When retrieval is deemed unnecessary, this prompt guides the model to generate an answer based solely on accumulated knowledge from previous reasoning steps. It emphasizes making the best inference even with incomplete information, ensuring decisive answers rather than abstention. The prompt explicitly requires specific answer formats for different question types and prohibits uncertain responses.

Knowledge-based answering prompt

```
Answer the subtask based on accumulated
knowledge.

Accumulated Knowledge: {knowledge}
Current Subtask: {subtask}

CRITICAL REQUIREMENTS:
- Based on the accumulated knowledge,
provide your BEST INFERENCE for the answer
- For yes/no questions: choose the more
likely option and answer
"<answer>yes</answer>" or
"<answer>no</answer>"
- For other questions: provide the most
probable specific answer(name, number,
date, place, etc.)
- If multiple possibilities exist, choose
the MOST PLAUSIBLE one

Respond with:
<think>analyze the evidence and explain
your reasoning</think>
<answer>your most likely answer (be
specific and decisive)</answer>
```

Continue Retrieval Decision. After each retrieval step, this prompt determines whether the retrieved information is sufficient to answer the question or if additional retrieval hops are needed. It tracks the number of hops performed to prevent infinite retrieval loops. The model evaluates the current information coverage and makes a binary decision on whether to continue the retrieval process.

Continue retrieval decision prompt

```
Decide if current information is sufficient
to answer the question.

Question: {question}
Current Information: {info}
Hops Done: {hop}/{max_hops}

Respond with:
<think>your reasoning</think>
<decision>retrieve</decision> or <decision>
answer</decision>

- Choose 'retrieve' if information is
insufficient and you need more
- Choose 'answer' if you have enough
information to answer
```

Query Generation (First Hop). This prompt generates the initial search query for retrieving relevant documents from the knowledge base. It takes the current question and any accumulated information to formulate an effective search query.

[16] https://github.com/deepspeedai/DeepSpeed
[17] https://github.com/Dao-AILab/flash-attention

The model is instructed to reason about what information is needed and construct a targeted query accordingly.

Initial query generation prompt

```
Generate a search query for the question.

Current Information: {accumulated_info}
Generate a query to search for solving the
question: {question}.

Respond with:
<think>your reasoning</think>
<query>your search query</query>
```

Query Generation (Subsequent Hops). For subsequent retrieval hops, this prompt ensures query diversity by explicitly instructing the model to generate different queries using alternative keywords, angles, or specificity levels. It provides the history of previously tried queries and explicitly constrains the model to avoid repetition. This prevents redundant retrieval and helps explore different information sources, improving the coverage of the retrieved knowledge.

Subsequent query generation prompt with diversity constraints

```
Generate a NEW and DIFFERENT search query
using a different approach or angle.

Current Information: {notes}
Previous Tried Queries: {tried_queries}
Hop: {hop}
Accumulated Knowledge: {accumulated}

Generate a query to search for solving the
question: {question}.
IMPORTANT REQUIREMENTS:
1. Use DIFFERENT keywords/phrases than
previous queries
2. Try a DIFFERENT angle or aspect
3. Be more SPECIFIC or use alternative
terminology
4. DO NOT repeat the same query structure
as before
5. If previous queries were too general, be
more specific; if too specific, try broader
terms

Respond with:
<think>your reasoning</think>
<query>your NEW search query</query>
```

F Case Studies

F.1 Case Study of Task Planning

The improvement of our model in task planning after training is shown in Figure 5. It can be seen from the figure that after training with RL-based Agentic Planning, our model is more accurate in task planning.

F.2 Case Study of Multi-Hop Question Answering

As shown in Figure 6, we use an example to demonstrate the excellence of the training model in task decomposition and multi-hop question answering.

Case Study: Question Decomposition Comparison

Question 1: *Who is the maternal grandfather of Antiochus X Eusebes?*

Qwen2.5-7B-Instruct:
1. Who is the mother of Antiochus X Eusebes?
2. Who is the father of Antiochus X Eusebes's mother?
3. What is the relationship of the answer to #2 to Antiochus X Eusebes?

Ours:
1. Who is the mother of Antiochus X Eusebes?
2. Who is the father of #1?

Question 2: *What is the award that the director of film Wearing Velvet Slippers Under A Golden Umbrella won?*

Qwen2.5-7B-Instruct:
1. What is the name of the director of the film Wearing Velvet Slippers Under A Golden Umbrella?
2. What award did this director win?
3. What is the award that the director won?

Ours:
1. Who is the director of the film Wearing Velvet Slippers Under A Golden Umbrella?
2. What award did #1 win?

Question 3: *Which country Audofleda's husband is from?*

Qwen2.5-7B-Instruct:
1. Which country Audofleda's husband is from?

(No decomposition)

Ours:
1. Who is Audofleda's husband?
2. Which country is #1 from?

Question 4: *Are the movies About A Wife, A Dream And Another... and The Doolins Of Oklahoma, from the same country?*

Qwen2.5-7B-Instruct:
1. What is the country of origin for the movie About A Wife?
2. What is the country of origin for the movie A Dream And Another...?
3. What is the country of origin for the movie The Doolins Of Oklahoma?

Ours:
1. What is the country of the movie About A Wife, A Dream And Another...?
2. What is the country of the movie The Doolins Of Oklahoma?

Figure 5: Comparison of question decomposition strategies. Our method produces more concise decompositions by effectively using reference notation (#1, #2) to chain sub-questions, while the baseline often generates redundant or incomplete decompositions.

Appendix: Case Study of Multi-Hop Question Answering

Original Question: *The continent of the country Motumua is located is what?*

User: Please decompose the problem directly into 2-4 sub-problems...
Assistant:

What is the name of the country Motumua?
On which continent is #1 located?

Iteration 1: Solving Sub-question 1
User: Based on accumulated knowledge, can you answer the subtask without retrieval?
Current Subtask: What is the name of the country Motumua?
Assistant:

<think>Motumua is not a widely known country...</think>
<decision>retrieve</decision>

User: Generate a search query for the question.
Assistant:

<think>I need to generate a search query...</think>
<query>country named Motumua location</query>

User: Decide if current information is sufficient...
Retrieved Context: Motumua is an islet of Nukufetau, Tuvalu...
Assistant:

<think>The provided documents indicate that Motumua is an islet of Nukufetau in Tuvalu...</think>
<decision>answer</decision>

User: Answer the question based on the context.
Assistant:

<think>Motumua is described as an islet of Nukufetau, which is part of Tuvalu...</think>
<answer>Tuvalu</answer>

Iteration 2: Solving Sub-question 2
User: Based on accumulated knowledge, can you answer the subtask?
Current Subtask: On which continent is Tuvalu located?
Accumulated Knowledge: Q1: What is the name of the country Motumua? → A1: Tuvalu
Assistant:

<think>...does not provide information about the continent on which Tuvalu is located...</think>
<decision>retrieve</decision>

User: Generate a search query for the question.
Assistant:

<query>continent of Tuvalu</query>

User: Decide if current information is sufficient...
Retrieved Context: Tuvalu is a Polynesian island country located in the Pacific Ocean, situated in Oceania...
Assistant:

<think>The provided documents describe Tuvalu's location in Oceania...</think>
<decision>answer</decision>

User: Answer the question based on the context.
Assistant:

<answer>Oceania</answer>

Final Answer Synthesis
User: Based on the information below, answer the question.
Information: Q1: Motumua → A1: Tuvalu; Q2: Continent of Tuvalu → A2: Oceania
Assistant:

<think>Motumua is in Tuvalu, and Tuvalu is located in Oceania...</think>
<answer>**Oceania**</answer>

Figure 6: Complete interaction trace showing the multi-hop reasoning process. The system decomposes the complex question, iteratively retrieves information for each sub-question, and synthesizes the final answer through structured prompting.