FLAP: Failure-Aware Planning for Search Agents

Anonymous ACL submission

Abstract

Search-augmented language agents are commonly trained with reinforcement learning (RL) to solve knowledge-intensive tasks. Despite strong performance, RL-trained search agents often suffer from two persistent failure modes: imprecise early queries and premature answer generation from insufficient evidence. Addressing these failures requires the agent to proactively plan what to retrieve and verify the retrieved evidence before answering. We propose FLAP (**F**ai**L**ure-**A**ware **P**lanning), a framework that separates search planning from search execution. FLAP trains a planner model to generate structured search plans with retrieval objectives and evidence-verification steps, while using an off-the-shelf language model as the search executor that follows the generated plan. We evaluate FLAP on seven question-answering benchmarks across four Qwen models of 0.5B–7B. FLAP consistently outperforms strong baselines including RL-based search agents by up to 8.5%. Our analyses show that the gains come from better early retrieval, reliable evidence verification, and recovery from failed searches.

1 Introduction

Large language models (LLMs) have demonstrated strong capabilities in reasoning and problem solving, yet they often struggle with knowledge-intensive tasks that require access to external information (Guu et al., 2020; Izacard et al., 2023; Lewis et al., 2021; Borgeaud et al., 2022; Izacard and Grave, 2021). To address this limitation, recent work has equipped LLMs with search tools, allowing them to interact with external search engines during reasoning (Trivedi et al., 2023; Qin et al., 2024; Gao et al., 2023; Hagström et al., 2025; Nakano et al., 2022; Patil et al., 2024).

A dominant way to train such search agents is reinforcement learning (RL). Under ReAct-style interaction formats, RL-trained agents learn to interleave reasoning, search queries, retrieved evidence, and final answers (Yao et al., 2023b; Jin et al., 2025; Li et al., 2025a). Although these methods have achieved good performance on challenging benchmarks, they still suffer from two persistent failure modes: **(1) imprecise early queries**, where the agent starts with vague or misdirected search queries; and **(2) premature answer generation**, where the agent answers from insufficient or irrelevant evidence. These failures suggest that search agents need not only the ability to call a search engine, but also explicit guidance on what to search for and how to revise the search direction when retrieval fails.

A natural direction to address this is to introduce planning before execution. However, existing plan-and-execute methods plan across steps but not within them: they decompose a task into high-level subtasks but treat each search as a single atomic action (Zhou et al., 2023; Yao et al., 2023a; Khot et al., 2023; Kim et al., 2024; Erdogan et al., 2025). This misses the actual difficulty of agentic search, which lies inside each retrieval step: what query to issue, whether the retrieved evidence is sufficient, and how to revise the query when retrieval fails (Wang et al., 2023b; Ma et al., 2023).

To bridge this gap, we propose **FLAP** (**F**ai**L**ure-**A**ware **P**lanning), a planning framework that makes search agents explicitly aware of retrieval failures. FLAP first trains a planner model to generate structured search plans consisting of retrieval objectives and evidence verification steps. During inference, an off-the-shelf search executor follows the plan to reason, issue search queries, and check the reliability of retrieved evidence.

We evaluate FLAP on seven question-answering benchmarks using Qwen models ranging from 0.5B to 7B parameters. FLAP consistently outperforms strong RL-based search agents, with gains of up to 8.5%, and existing plan-and-execute baselines,

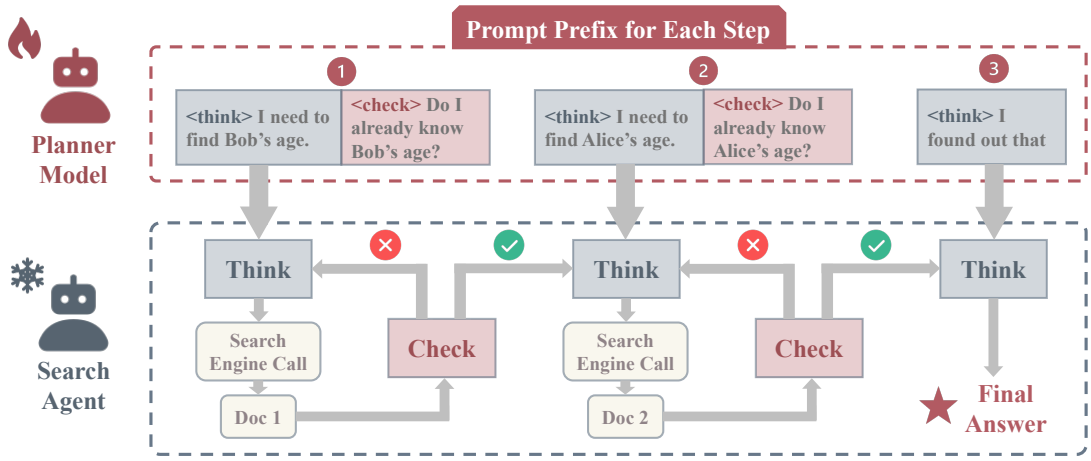

Figure 1: **Overview of FLAP. In Stage 1**, a trained planner generates a structured search plan that specifies retrieval objectives and evidence verification objectives. **In Stage 2**, an off-the-shelf search agent follows the plan to retrieve evidence, verify its sufficiency, and decide whether to proceed or revise the current retrieval objective.

with gains of up to 20%. Further analyses show that FLAP improves early-stage retrieval quality, more reliably detects insufficient evidence, recovers from failed searches through local replanning, and reduces redundant search engine calls.

Our contributions are summarized as follows:

- We propose FLAP, a failure-aware planning framework that addresses two persistent failure modes of RL-trained search agents. FLAP trains a planner model to generate retrieval and verification guidance for an off-the-shelf search executor.
- We conduct extensive experiments across seven QA benchmarks and four model scales. FLAP consistently outperforms strong baselines by up to 8.5%. Improvements are driven by better early retrieval, evidence verification, and failure recovery.

2 FLAP: Failure-Aware Planning Framework for Agentic Search

In the following sections, we present the detailed design of FLAP, including the preliminary of search agents (§ 2.1), the overall framework (§ 2.2), and the training of the planner model (§ 2.3).

2.1 Preliminary: Search Agent

Given a question-answering dataset $\mathcal{D}$ and an external search engine $\mathcal{E}$, a search agent answers each question by iteratively interacting with $\mathcal{E}$. At each round, the agent conditions on the current context, generates an internal reasoning step, and either issues a search query to obtain external evidence or produces a final answer. The retrieved evidence is then appended to the context and used in subsequent reasoning steps (Jiang et al., 2023; Jeong et al., 2024).

Formally, for a question $q \in \mathcal{D}$, the agent produces an interaction trajectory

$$\tau = (r_0, a_0, x_0, r_1, a_1, x_1, \ldots, r_{T-1}, o),$$

where r_t denotes the model's reasoning step at round t, a_t denotes the search query issued to the external search engine, $x_t = \mathcal{E}(a_t)$ denotes the retrieved evidence, and o denotes the final answer. The trajectory terminates when the model outputs a final answer.

In this paper, we instantiate the search agent following the format introduced by (Jin et al., 2025). The model's reasoning steps are enclosed between the reasoning tokens `<think>` and `</think>`. Whenever external information is needed, the model is instructed to place its search query between the search tokens `<search>` and `</search>`. The system then extracts the enclosed search query, invokes the search engine, and appends the retrieved evidence to the model context between the information tokens `<information>` and `</information>`. When the model determines that it has sufficient evidence to answer, it produces the final response between the answer tokens `<answer>` and `</answer>`.

2.2 Failure-Aware Planning and Execution

Figure 1 presents the overall framework of FLAP. Our design decomposes agentic search into two stages: (1) a planner model that generates a structured search plan, and (2) a search agent that performs retrieval and reasoning under the guidance

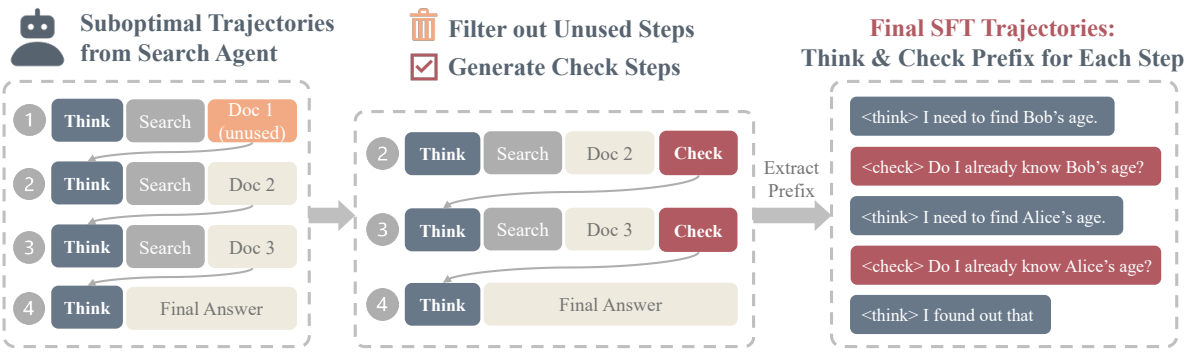

Figure 2: **Data construction pipeline for training the planner model** (Stage 1 in Figure 1). We first collect suboptimal trajectories from an RL-trained search agent. Then, we retain only successful trajectories and prompt a frontier LLM to transform them into structured plans by removing unnecessary search steps, rewriting retrieval objectives into clearer planning cues, and inserting think and check steps.

of this plan. To operationalize this framework, we introduce four key components: (1) *check steps*; (2) *search plans*; (3) *failure-triggered local replanning*; and (4) *open-ended prompt injection*.

Check steps. A standard search trajectory only records reasoning steps, search queries, retrieved evidence, and the final answer. However, this format does not require the agent to decide whether the retrieved evidence is sufficient before continuing the trajectory. To address this issue, FLAP augments each retrieval step with a *check step*.

Formally, let M_a denote the search agent model used during execution. Let g_t denote the retrieval objective at step t, a_t denote the search query generated for this objective, and $x_t = \mathcal{E}(a_t)$ denote the retrieved evidence. The planner also provides a check objective c_t, which specifies what condition the retrieved evidence should satisfy. After observing x_t, the agent produces a verification result s_t:

$$s_t = M_a(q, H_t, g_t, a_t, x_t, c_t),$$

where H_t denotes the previous interaction history and $s_t \in \{\textsc{Pass}, \textsc{Fail}\}$. The result $s_t = \textsc{Pass}$ indicates that the retrieved evidence is sufficient for the current retrieval objective, while $s_t = \textsc{Fail}$ indicates that the evidence is insufficient.

With check steps, the interaction trajectory becomes

$$\tau = (r_0, g_0, a_0, x_0, c_0, s_0, r_1, g_1, a_1, x_1, \ldots, o),$$

where r_t denotes the reasoning step and o denotes the final answer. If $s_t = \textsc{Pass}$, the agent proceeds to the next planned step. If $s_t = \textsc{Fail}$, FLAP triggers local replanning to revise the current retrieval objective before issuing another search query. This explicit verification process prevents the agent from prematurely committing to an answer based on irrelevant or insufficient evidence.

In implementation, the check step is realized with the special tokens `<check>` and `</check>`, and the verification result is represented by `<pass>` or `<fail>`. These tokens serve only as a structured interface for parsing the agent's decision.

Search plans. The planner model generates a structured search plan conditioned on the input question. We denote the plan as

$$P = \{(g_1, c_1), (g_2, c_2), \ldots, (g_K, c_K)\},$$

where g_i is the retrieval objective for step i, and c_i is the corresponding check objective. The retrieval objective specifies what information the agent should search for, while the check objective specifies what condition the retrieved evidence must satisfy. During execution, each plan step is injected into the search agent context as guidance.

Failure-triggered local replanning. After each retrieval step, the agent performs a check step and produces a verification result s_t. When $s_t = \textsc{Fail}$, FLAP performs local replanning to correct the failed retrieval. Instead of restarting the entire reasoning trajectory, FLAP revises only the current retrieval objective.

Concretely, for each retrieval objective g_t, the planner also provides a set of paraphrased alternatives

$$\mathcal{A}(g_t) = \{\tilde{g}_t^{(1)}, \tilde{g}_t^{(2)}, \ldots\}.$$

If the original retrieval fails, FLAP replaces g_t with an alternative objective $\tilde{g}_t^{(j)}$ and asks the agent to revisit the same step under the revised guidance. The remaining plan structure is preserved.

For example, if the current objective is to find Bob's age but the retrieved documents do not contain this information, FLAP may replace the objective with a semantically related alternative such as finding Bob's birth date. This encourages the agent to search from a different angle while avoiding repeated global replanning. Since the alternative objectives are generated during the planning stage, local replanning does not require an additional planner call at inference time.

We provide an algorithm that summarizes the workflow of FLAP in Appendix B. Unless otherwise specified, we set the search budget for each step to $B = 2$, meaning that the agent may revisit a failed retrieval step at most once.

Open-ended prompt injection. FLAP injects plan guidance into the search agent using open-ended prompt prefixes. For each plan step, the system provides the retrieval objective g_t and the check objective c_t. The system injects only the opening prefix and the guidance sentence, but does not inject the closing token. The agent therefore continues generation from the prefix, produces its own reasoning or check step, and decides when to close the corresponding segment.

2.3 Training the Planner Model

Training the planner model requires a dataset that maps an input question to a high-quality search plan. However, such a dataset is not readily available, since existing question-answering datasets typically provide only input questions and final answers. To address this issue, we construct a synthetic planning dataset using successful trajectories generated by RL-trained search agents.

As illustrated in Figure 2, we first train a search agent using PPO following prior work (Jin et al., 2025). We then use this agent to generate reasoning trajectories on NQ and HotpotQA datasets (Kwiatkowski et al., 2019; Yang et al., 2018). From these trajectories, we retain only successful search executions. However, we observe that raw RL trajectories are often unnecessarily long, contain redundant searches, and include noisy intermediate reasoning. Therefore, we further prompt a frontier LLM to optimize these trajectories into shorter and more structured reasoning plans.

Specifically, the frontier model performs three refinements: (1) removing redundant search steps; (2) rewriting search queries into explicit retrieval objectives; and (3) adding verification objectives for the corresponding check steps.

In addition, we ask the frontier model to generate synthetic query paraphrases, which serve as alternative guidance sentences for both planner training and query replacement during failure-triggered local replanning (see § 2.2).

3 Experiments

3.1 Experimental Setup

Datasets. We evaluate FLAP on seven question-answering datasets, including: **(1) General Question Answering**: NQ (Kwiatkowski et al., 2019), TriviaQA (Joshi et al., 2017), and PopQA (Mallen et al., 2023). **(2) Multi-Hop Question Answering**: HotpotQA (Yang et al., 2018), 2WikiMultiHopQA (Ho et al., 2020), Musique (Trivedi et al., 2022), and Bamboogle (Press et al., 2023). We provide details of these datasets in Appendix D.1.

Baselines. We compare FLAP against a diverse set of baselines, including: **(1) Standard Inference:** Direct inference, chain-of-thought reasoning (Wei et al., 2023), Retrieval-Augmented Generation (RAG) (Lewis et al., 2021). **(2) Traditional Training-Based Agentic Search**: Supervised fine-tuning (SFT) (Chung et al., 2024), Search-o1 (Li et al., 2025a), R1 (Guo et al., 2025), Search-R1 (Jin et al., 2025). **(3) Plan-and-Execute style methods**: Plan-and-Act (Erdogan et al., 2025), HiMAC (Jin et al., 2026), HiPlan (Li et al., 2025b).

Model settings. We conduct experiments on four models Qwen2.5-0.5B-Instruct, Qwen2.5-1.5B-Instruct, Qwen2.5-3B-Instruct and Qwen2.5-7B-Instruct (Qwen et al., 2025). Unless otherwise specified, we use GPT-4o-mini as the frontier model for planning data construction, and Qwen2.5-7B-Instruct as the planner model for all experiments (Hurst et al., 2024).

Planner training. We first merge the training sets of NQ and HotpotQA to form a unified dataset. The planner is then fine-tuned on our constructed planning dataset using supervised fine-tuning. We optimize the standard autoregressive language modeling objective, where the loss is computed over the target plan tokens conditioned on the input question.

Retrieval settings. We use the 2018 Wikipedia dump (Karpukhin et al., 2020) as the knowledge source and E5 (Wang et al., 2024) as the search engine, following the setting of Jin et al. (2025).

Method	In-Distribution		Out-of-Distribution					Avg.
	NQ	HotpotQA	TriviaQA	PopQA	2Wiki	MuSiQue	Bamboogle	
Search Agent Model: Qwen2.5-7B-Instruct								
Standard Inference								
Direct Inference	14.1	18.6	42.0	14.8	25.5	3.6	12.9	18.7
Chain-of-Thought	5.3	9.9	19.4	5.9	11.8	2.5	23.6	11.2
RAG	36.5	31.2	60.1	40.9	24.6	6.2	21.9	31.6
Traditional Training-Based Agentic Search								
SFT	33.2	22.2	36.9	12.9	26.5	7.1	12.0	21.5
Search-o1	16.0	19.1	45.5	13.9	18.5	6.3	30.2	21.3
R1	28.3	24.3	55.9	20.9	30.0	7.7	30.6	28.3
Search-R1	40.9	38.6	62.8	40.8	42.6	15.4	38.4	39.9
Plan-and-Execute Methods								
Plan-and-Act	24.3	20.9	45.1	23.3	18.4	6.1	19.3	22.4
HiMAC	26.0	22.8	46.8	25.0	20.1	7.0	21.0	24.2
HiPlan	27.7	24.7	48.3	26.7	21.9	7.8	22.8	25.6
Ours								
FLAP	**47.4** (+6.5)	**44.3** (+5.7)	**71.3** (+8.5)	**46.9** (+6.1)	**48.6** (+6.0)	**18.5** (+3.1)	**44.3** (+5.9)	**45.9** (+6.0)
Search Agent Model: Qwen2.5-0.5B-Instruct								
Standard Inference								
Direct Inference	7.0	9.3	22.7	7.4	12.3	1.4	6.2	9.5
Chain-of-Thought	2.3	4.1	8.6	2.6	5.3	0.9	10.6	4.9
RAG	18.3	15.7	35.0	20.8	13.4	3.0	11.3	16.7
Traditional Training-Based Agentic Search								
SFT	17.1	11.3	19.9	6.3	13.6	3.4	6.4	11.2
Search-o1	7.8	9.6	25.4	6.7	9.0	2.7	14.8	10.9
R1	14.7	12.4	30.3	10.9	16.1	4.1	15.6	14.9
Search-R1	22.3	21.1	37.3	21.5	23.6	7.9	21.4	22.1
Plan-and-Execute Methods								
Plan-and-Act	14.5	12.8	27.7	13.4	11.8	3.5	11.3	13.6
HiMAC	15.7	14.0	29.0	14.5	13.1	4.1	12.4	14.7
HiPlan	16.7	15.2	30.0	15.7	14.4	4.7	13.6	15.8
Ours								
FLAP	**26.0** (+3.7)	**24.5** (+3.4)	**42.6** (+5.3)	**24.9** (+3.4)	**27.2** (+3.6)	**9.9** (+2.0)	**25.0** (+3.6)	**25.7** (+3.6)

Table 1: Performance comparison across Qwen2.5-7B-Instruct and Qwen2.5-0.5B-Instruct search agent models. Results are reported as exact match (EM, %). Values in parentheses indicate absolute improvements over the strongest baseline for each setting. Results for 1.5B and 3B models are in Table 7 and Table 8, Appendix C.

To ensure fair comparison, we follow Lin et al. (2024) and set the number of retrieved passages to 3 across all retrieval-based methods (Andrew and Gao, 2007).

Metrics. Following Karpukhin et al. (2020) and Jin et al. (2025), we use Exact Match (EM) as the main evaluation metric. Unless otherwise specified, all references to "accuracy" in this paper refer to EM, and all reported results use Pass@1. We provide more experiment details in Appendix D.

3.2 Main Results

Table 1 summarizes the main results on Qwen2.5-7B-Instruct and Qwen2.5-0.5B-Instruct. FLAP consistently achieves the best performance across all datasets and model sizes. With Qwen2.5-7B-Instruct as the search executor, FLAP outperforms Search-R1 by 6.0% and HiPlan by 20.3%. The gains are consistent across both in-distribution and out-of-distribution datasets, indicating that failure-aware planning provides robust guidance for agentic search.

The improvement remains clear even with the much smaller Qwen2.5-0.5B-Instruct executor. FLAP achieves 25.7% average EM, outperforming Search-R1 by 3.6% and HiPlan by 9.9%. This suggests FLAP especially helpful for smaller agents with limited search capability. Additional results on 1.5B and 3B search agents are reported in Appendix C.

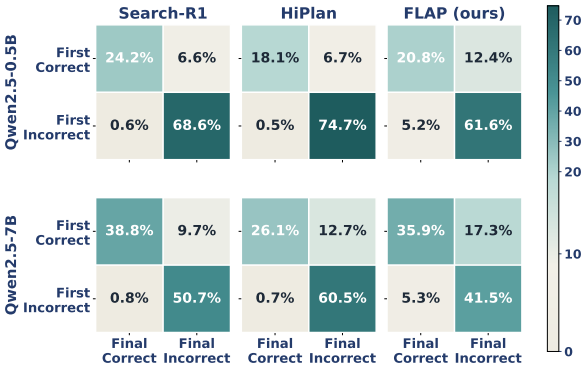

Figure 3: **Effect of first-retrieval correctness on final-answer correctness.** Across methods and model scales, correct first retrievals are more likely to lead to correct final answers, while incorrect first retrievals are often associated with final-answer failures.

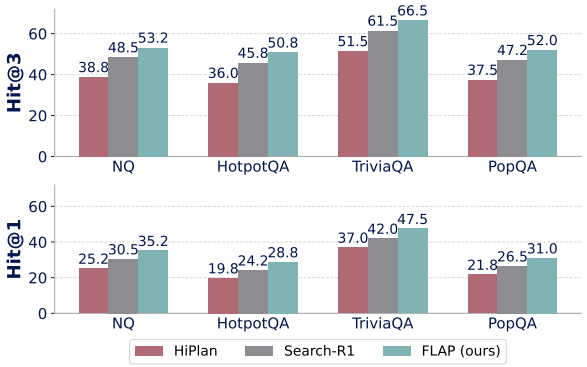

Figure 4: **First-retrieval accuracy on Qwen2.5-7B-Instruct.** Hit@k measures whether at least one of the top-k retrieved passages is relevant to answering the question. FLAP consistently outperforms Search-R1 and HiPlan, showing that structured retrieval objectives help the agent generate more targeted initial queries.

4 Why FLAP Works

We next analyze why FLAP outperforms existing methods. We use Search-R1 and HiPlan as representative strong baselines from the RL-based agentic search and plan-and-execute categories.

4.1 FLAP Improves Early-Stage Retrieval

A key motivation of FLAP is that the first retrieval step plays a critical role in agentic search. An irrelevant or insufficient first retrieval can mislead the agent into an incorrect search direction and eventually cause unsupported answer generation.

First retrieval strongly affects final answer. We first examine the relationship between first-retrieval quality and final answer correctness. We sample 1000 examples from the test sets of TriviaQA and HotpotQA, collect the reasoning trajectories of three representative methods, and manually annotate whether the first retrieval contains evidence useful for answering the question. We use this subset for the subsequent analyses unless otherwise specified. We then group examples according to two binary variables: whether the first retrieval is correct and whether the final answer is correct. Figure 3 shows that first-retrieval quality is strongly associated with final-answer correctness.

FLAP improves first-retrieval quality. Given the importance of early retrieval, we further evaluate whether FLAP improves the quality of the first search step. Figure 4 shows that FLAP improves the quality of the first retrieval step.

4.2 FLAP Detects Irrelevant Evidence and Corrects Search Queries

After showing that early retrieval quality strongly affects final answer correctness, we further examine whether a search agent can recover when the initial retrieval fails. This requires two capabilities: the agent must first detect that the retrieved evidence is insufficient, and then revise its search query to retrieve more relevant evidence.

Evidence sufficiency detection. We evaluate whether each method can correctly judge whether the retrieved evidence is sufficient for answering the question. Using the annotated subset introduced in Section 4.1, we treat sufficient retrieval as the positive class and compute precision, recall, F1, and accuracy for the evidence sufficiency decision. Table 2 shows that FLAP provides more reliable evidence sufficiency decisions across model sizes.

Query correction after failed retrieval. Detecting irrelevant evidence is only the first step; the agent must also recover by issuing a better follow-up query. We therefore evaluate the quality of the second retrieval conditioned on cases where the first retrieval fails. Specifically, we measure 2nd-Hit@1 and 2nd-Hit@3, which indicate whether the revised search retrieves relevant evidence in the top 1 or top 3 passages. Figure 5 evaluates whether methods can recover after an initial retrieval failure.

4.3 FLAP Reduces Redundant Tool Usage

We further analyze whether FLAP improves the efficiency of agentic search by reducing redundant

Model	Method	P	R	F1	Acc
0.5B	HiPlan	35.7	27.3	30.9	41.4
	Search-R1	32.8	**48.6**	39.2	44.4
	FLAP (ours)	**46.3**	40.8	**43.4**	**50.2**
	Diff.	+13.5	-7.8	+4.2	+5.8
1.5B	HiPlan	37.6	29.4	33.0	43.0
	Search-R1	35.0	**50.8**	41.5	45.8
	FLAP (ours)	**48.4**	42.7	**45.4**	**51.8**
	Diff.	+13.4	-8.1	+3.9	+6.0
3B	HiPlan	44.2	35.5	39.4	50.2
	Search-R1	41.1	**56.5**	47.6	53.0
	FLAP (ours)	**56.7**	49.6	**52.9**	**60.0**
	Diff.	+15.6	-6.9	+5.3	+7.0
7B	HiPlan	51.4	42.0	46.2	57.7
	Search-R1	48.4	**66.5**	56.0	60.7
	FLAP (ours)	**65.7**	57.4	**61.3**	**68.9**
	Diff.	+17.3	-9.1	+5.3	+8.2

Table 2: **Evidence sufficiency detection across model sizes.** Sufficient evidence is treated as the positive class. P, R, F1, and Acc denote precision, recall, F1 score, and accuracy. *Diff.* row reports the absolute difference between FLAP and Search-R1. FLAP achieves the best F1 and accuracy across all model sizes, indicating a better precision-recall trade-off. Search-R1 obtains higher recall but lower precision, suggesting more insufficient evidence is incorrectly accepted as sufficient.

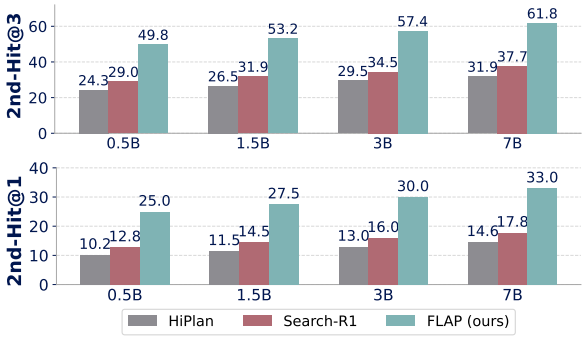

Figure 5: **Second-retrieval accuracy after first-retrieval failure.** We report 2nd-Hit@3 and 2nd-Hit@1 on annotated examples where the first retrieval fails. FLAP substantially improves second-retrieval accuracy across model sizes, showing that check steps and failure-triggered local replanning help correct the search direction.

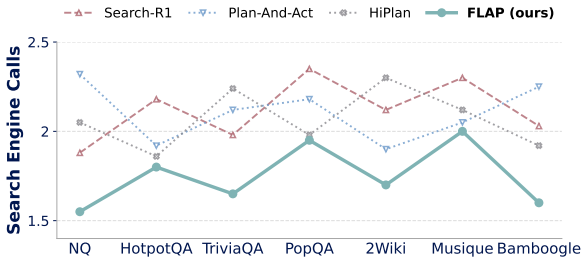

Figure 6: **Average number of search engine calls.** FLAP consistently requires fewer search calls than Search-R1 and plan-and-execute baselines.

search engine calls. At first glance, the local replanning mechanism in FLAP may appear to increase tool usage. Surprisingly, we find that FLAP achieves stronger performance while using fewer search calls on average. Figure 6 compares the average number of search engine calls across methods. This reduction comes from more targeted retrieval objectives and check steps that prevent unnecessary follow-up searches when the current evidence is already sufficient.

5 Further Analysis

We further analyze three design choices of FLAP: whether each component is necessary (§ 5.1), how much search budget is needed (§ 5.2), and whether the method depends on the frontier model or planner size (§ 5.3, § 5.4).

5.1 Ablation Study

We conduct an ablation study to examine the contribution of each component in FLAP. Table 3 reports the overall average EM on seven datasets using Qwen2.5-0.5B-Instruct and Qwen2.5-7B-Instruct as search agents. Removing any key component leads to a clear performance drop, showing that the gains of FLAP rely on the combination of structured planning, evidence checking, local replanning, and flexible prompt injection. The ablation results show that all components are important for the final performance.

5.2 Effect of Search Budget

We study how the search budget affects the performance and efficiency of FLAP. Here, the budget denotes the maximum number of search attempts allowed for each plan step. A budget of 1 disables local replanning, while a budget of 2 allows the agent to revisit a failed retrieval step once. Larger budgets allow additional query revisions after repeated failures. Table 4 studies the trade-off between accuracy and search cost under different search budgets.

5.3 Does FLAP Depend on the Frontier Model?

We examine whether the effectiveness of FLAP comes from its failure-aware design or from the use of a frontier model during planner data con-

Variant	0.5B	7B	Avg.	Δ
FLAP	**25.7**	**45.9**	**35.8**	–
w/o <check>	20.5	36.8	28.7	−7.1
w/o local replanning	21.2	38.2	29.7	−6.1
w/o paraphrased objectives	21.7	38.9	30.3	−5.5
w/o open-ended injection	22.1	39.4	30.8	−5.0
w/o planner SFT	21.0	37.7	29.4	−6.4

Table 3: **Ablation study of FLAP.** We report the overall average exact match (EM, %) across seven datasets. Δ denotes the average drop compared with the full FLAP model.

Budget	0.5B EM	0.5B Calls	7B EM	7B Calls
1	21.9	1.32	40.0	1.39
2	**25.7**	**1.74**	**45.9**	**1.78**
3	25.4	2.11	45.4	2.15
4	25.1	2.43	45.1	2.47

Table 4: **Effect of search budget.** We report average exact match (EM, %) and average search engine calls across seven datasets. Increasing the budget from 1 to 2 substantially improves EM. Larger budgets introduce more search calls without further improving EM.

Method / Data Constructor	0.5B	7B	Avg.
Search-R1	22.1	39.9	31.0
FLAP w/o frontier LLM	23.2	41.4	32.3
FLAP w/ Llama-3.1-70B-Instruct	24.2	43.5	33.9
FLAP w/ Qwen2.5-72B-Instruct	24.8	44.3	34.6
FLAP w/ Claude-3.5-Haiku	25.1	44.9	35.0
FLAP w/ GPT-4o-mini	**25.7**	**45.9**	**35.8**

Table 5: **Effect of planner data constructor.** Even without frontier-model optimization, FLAP still outperforms Search-R1, suggesting that the core gain comes from the failure-aware planning structure.

Planner Model	0.5B	7B	Avg.
Qwen2.5-1.5B-Instruct	22.6	40.7	31.7
Qwen2.5-3B-Instruct	24.0	42.8	33.4
Qwen2.5-7B-Instruct	**25.7**	**45.9**	**35.8**
Qwen2.5-14B-Instruct	25.9	46.2	36.1

Table 6: **Effect of planner model size.** Performance improves clearly from 1.5B to 7B. The gain becomes marginal from 7B to 14B, so we use Qwen2.5-7B-Instruct as the default planner to balance performance and cost.

struction. To isolate this factor, we compare FLAP under different sources of planner supervision. In particular, *FLAP w/o frontier LLM* constructs planner training data through pure rule-based conversion of successful RL trajectories. Table 5 studies whether FLAP depends on a specific frontier model for planner data construction.

5.4 Effect of Planner Model Size

We further study whether the performance of FLAP depends on the size of the planner model. We vary the planner size while keeping the search agent, search engine, and execution procedure fixed. This experiment tests how much planner capacity is needed to produce effective retrieval and verification guidance. Table 6 studies how planner model size affects FLAP.

6 Discussion

Related works. Recent agentic search methods go beyond fixed retrieval by allowing models to interactively issue search queries and incorporate retrieved evidence during generation. These methods are often trained through supervised fine-tuning or reinforcement learning, as in Search-o1, R1, and Search-R1 (Chung et al., 2024; Li et al., 2025a; Guo et al., 2025; Jin et al., 2025). While effective, these training-based search agents can still suffer from imprecise early queries and premature answers based on insufficient evidence.

Another related line of work studies planning for language agents. Existing plan-and-execute methods decompose complex tasks into intermediate steps before execution, including Plan-and-Act, HiMAC, and HiPlan (Erdogan et al., 2025; Jin et al., 2026; Li et al., 2025b). These methods provide useful high-level task structure, but they typically treat retrieval as an action within a larger workflow rather than modeling the internal decisions of each retrieval step. We provide a more complete discussion of related work in Appendix A.

Conclusion. We present FLAP, a failure-aware planning framework for agentic search. FLAP trains a planner to generate fine-grained guidance, while using an off-the-shelf search executor during inference. Experiments on seven question-answering benchmarks show that FLAP consistently improves answer accuracy, retrieval quality, failure recovery, and search efficiency over strong RL-based and plan-and-execute baselines. Overall, FLAP suggests a new paradigm for building search agents: improving agentic search through explicit planning and verification rather than directly fine-tuning the search executor.

Limitations

One limitation of FLAP is that the planner data construction pipeline is relatively expensive. Constructing the training data requires first training an RL-based search agent to generate successful trajectories, and then prompting a frontier LLM to refine these trajectories into structured failure-aware plans with retrieval objectives, verification objectives, and paraphrased alternatives. This multi-stage process incurs non-trivial computational and API costs compared to directly fine-tuning a search agent.

Several directions remain open for future exploration. First, it would be interesting to investigate cheaper data construction strategies, such as self-distillation from the planner itself or iterative bootstrapping that gradually reduces reliance on a frontier LLM. Second, extending FLAP to richer agentic settings beyond single-retriever question answering, including multi-tool environments, live web search, and long-horizon agentic workflows, would help characterize the generality of failure-aware planning. Finally, exploring more expressive verification signals beyond the binary `<pass>` `<fail>` decision, as well as adaptive search budgets that vary with question difficulty, could further improve both the accuracy and efficiency of the framework.

Potential Risks. FLAP is a retrieval-augmented agent framework, and its outputs depend on both the retrieval corpus and the search agent. As with other retrieval-augmented systems, incorrect or biased information in the underlying corpus may propagate to final answers. The evidence-verification step in FLAP reduces but does not eliminate this risk. Therefore, FLAP should not be deployed in high-stakes settings (e.g., medical or legal decision-making) without additional human oversight.

References

Galen Andrew and Jianfeng Gao. 2007. Scalable training of L1-regularized log-linear models. In *Proceedings of the 24th International Conference on Machine Learning*, pages 33–40.

Pepa Atanasova, Jakob Grue Simonsen, Christina Lioma, and Isabelle Augenstein. 2020. Generating fact checking explanations. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 7352–7364, Online. Association for Computational Linguistics.

Sebastian Borgeaud, Arthur Mensch, Jordan Hoffmann, Trevor Cai, Eliza Rutherford, Katie Millican, George Bm Van Den Driessche, Jean-Baptiste Lespiau, Bogdan Damoc, Aidan Clark, Diego De Las Casas, Aurelia Guy, Jacob Menick, Roman Ring, Tom Hennigan, Saffron Huang, Loren Maggiore, Chris Jones, Albin Cassirer, and 9 others. 2022. Improving language models by retrieving from trillions of tokens. In *Proceedings of the 39th International Conference on Machine Learning*, volume 162 of *Proceedings of Machine Learning Research*, pages 2206–2240. PMLR.

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

Hyung Won Chung, Le Hou, Shayne Longpre, Barret Zoph, Yi Tay, William Fedus, Yunxuan Li, Xuezhi Wang, Mostafa Dehghani, Siddhartha Brahma, Albert Webson, Shixiang Shane Gu, Zhuyun Dai, Mirac Suzgun, Xinyun Chen, Aakanksha Chowdhery, Alex Castro-Ros, Marie Pellat, Kevin Robinson, and 16 others. 2024. Scaling instruction-finetuned language models. *Journal of Machine Learning Research*, 25(70):1–53.

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. In *Proceedings of the 42nd International Conference on Machine Learning*, volume 267 of *Proceedings of Machine Learning Research*, pages 15419–15462. PMLR.

Luyu Gao, Zhuyun Dai, Panupong Pasupat, Anthony Chen, Arun Tejasvi Chaganty, Yicheng Fan, Vincent Zhao, Ni Lao, Hongrae Lee, Da-Cheng Juan, and Kelvin Guu. 2023. RARR: Researching and revising what language models say, using language models. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 16477–16508. Association for Computational Linguistics.

Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Peiyi Wang, Qihao Zhu, Runxin Xu, Ruoyu Zhang, Shirong Ma, Xiao Bi, Xiaokang Zhang, Xingkai Yu, Yu Wu, Z. F. Wu, Zhibin Gou, Zhihong Shao, Zhuoshu Li, Ziyi Gao, Aixin Liu, and 175 others. 2025. Deepseek-r1 incentivizes reasoning in llms through reinforcement learning. *Nature*, 645(8081):633–638.

Xiao-Yu Guo, Yuan-Fang Li, and Gholamreza Haffari. 2022. Complex reading comprehension through question decomposition. *Preprint*, arXiv:2211.03277.

Kelvin Guu, Kenton Lee, Zora Tung, Panupong Pasupat, and Mingwei Chang. 2020. Retrieval augmented language model pre-training. In *Proceedings of the 37th International Conference on Machine Learning*,

volume 119 of *Proceedings of Machine Learning Research*, pages 3929–3938. PMLR.

Lovisa Hagström, Sara Vera Marjanovic, Haeun Yu, Arnav Arora, Christina Lioma, Maria Maistro, Pepa Atanasova, and Isabelle Augenstein. 2025. A reality check on context utilisation for retrieval-augmented generation. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 19691–19730, Vienna, Austria. Association for Computational Linguistics.

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

Aaron Hurst, Adam Lerer, Adam P Goucher, Adam Perelman, Aditya Ramesh, Aidan Clark, AJ Ostrow, Akila Welihinda, Alan Hayes, Alec Radford, and 1 others. 2024. Gpt-4o system card. *arXiv preprint arXiv:2410.21276*.

Gautier Izacard and Edouard Grave. 2021. Leveraging passage retrieval with generative models for open domain question answering. In *Proceedings of the 16th Conference of the European Chapter of the Association for Computational Linguistics: Main Volume*, pages 874–880, Online. Association for Computational Linguistics.

Gautier Izacard, Patrick Lewis, Maria Lomeli, Lucas Hosseini, Fabio Petroni, Timo Schick, Jane Dwivedi-Yu, Armand Joulin, Sebastian Riedel, and Edouard Grave. 2023. Atlas: Few-shot learning with retrieval augmented language models. *Journal of Machine Learning Research*, 24(251):1–43.

Soyeong Jeong, Jinheon Baek, Sukmin Cho, Sung Ju Hwang, and Jong Park. 2024. Adaptive-RAG: Learning to adapt retrieval-augmented large language models through question complexity. In *Proceedings of the 2024 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, pages 7036–7050, Mexico City, Mexico. Association for Computational Linguistics.

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

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

Hongbo Jin, Rongpeng Zhu, Jiayu Ding, Guibo Luo, and Ge Li. 2026. Himac: Hierarchical macro-micro learning for long-horizon llm agents. *Preprint*, arXiv:2603.00977.

Mandar Joshi, Eunsol Choi, Daniel S. Weld, and Luke Zettlemoyer. 2017. Triviaqa: A large scale distantly supervised challenge dataset for reading comprehension. *Preprint*, arXiv:1705.03551.

Vladimir Karpukhin, Barlas Oğuz, Sewon Min, Patrick Lewis, Ledell Wu, Sergey Edunov, Danqi Chen, and Wen tau Yih. 2020. Dense passage retrieval for open-domain question answering. *Preprint*, arXiv:2004.04906.

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

Sehoon Kim, Suhong Moon, Ryan Tabrizi, Nicholas Lee, Michael W. Mahoney, Kurt Keutzer, and Amir Gholami. 2024. An LLM compiler for parallel function calling. In *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*, pages 24370–24391. PMLR.

Wei-Jen Ko, Cutter Dalton, Mark Simmons, Eliza Fisher, Greg Durrett, and Junyi Jessy Li. 2022. Discourse comprehension: A question answering framework to represent sentence connections. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, pages 11752–11764, Abu Dhabi, United Arab Emirates. Association for Computational Linguistics.

Tom Kwiatkowski, Jennimaria Palomaki, Olivia Redfield, Michael Collins, Ankur Parikh, Chris Alberti, Danielle Epstein, Illia Polosukhin, Jacob Devlin, Kenton Lee, Kristina Toutanova, Llion Jones, Matthew Kelcey, Ming-Wei Chang, Andrew M. Dai, Jakob Uszkoreit, Quoc Le, and Slav Petrov. 2019. Natural questions: A benchmark for question answering research. *Transactions of the Association for Computational Linguistics*, 7:452–466.

Piyawat Lertvittayakumjorn, David Kinney, Vinodkumar Prabhakaran, Donald Martin Jr., and Sunipa Dev. 2025. Towards geo-culturally grounded LLM generations. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 313–330, Vienna, Austria. Association for Computational Linguistics.

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. 2021. Retrieval-augmented generation for knowledge-intensive nlp tasks. *Preprint*, arXiv:2005.11401.

Minghao Li, Yingxiu Zhao, Bowen Yu, Feifan Song, Hangyu Li, Haiyang Yu, Zhoujun Li, Fei Huang, and Yongbin Li. 2023. API-bank: A comprehensive benchmark for tool-augmented LLMs. In *Proceedings of the 2023 Conference on Empirical Methods*

in Natural Language Processing, pages 3102–3116, Singapore. Association for Computational Linguistics.

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

Ziyue Li, Yuan Chang, Gaihong Yu, and Xiaoqiu Le. 2025b. Hiplan: Hierarchical planning for llm-based agents with adaptive global-local guidance. *Preprint*, arXiv:2508.19076.

Victoria Lin, Xilun Chen, Mingda Chen, Weijia Shi, Maria Lomeli, Richard James, Pedro Rodriguez, Jacob Kahn, Gergely Szilvasy, Mike Lewis, and 1 others. 2024. Ra-dit: Retrieval-augmented dual instruction tuning. In *International Conference on Learning Representations*, volume 2024, pages 19138–19162.

Xinbei Ma, Yeyun Gong, Pengcheng He, Hai Zhao, and Nan Duan. 2023. Query rewriting in retrieval-augmented large language models. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 5303–5315, Singapore. Association for Computational Linguistics.

Alex Mallen, Akari Asai, Victor Zhong, Rajarshi Das, Daniel Khashabi, and Hannaneh Hajishirzi. 2023. When not to trust language models: Investigating effectiveness of parametric and non-parametric memories. *Preprint*, arXiv:2212.10511.

Reiichiro Nakano, Jacob Hilton, Suchir Balaji, Jeff Wu, Long Ouyang, Christina Kim, Christopher Hesse, Shantanu Jain, Vineet Kosaraju, William Saunders, Xu Jiang, Karl Cobbe, Tyna Eloundou, Gretchen Krueger, Kevin Button, Matthew Knight, Benjamin Chess, and John Schulman. 2022. Webgpt: Browser-assisted question-answering with human feedback. *Preprint*, arXiv:2112.09332.

Shishir G. Patil, Tianjun Zhang, Xin Wang, and Joseph E. Gonzalez. 2024. Gorilla: Large language model connected with massive apis. In *Advances in Neural Information Processing Systems*, volume 37, pages 126544–126565. Curran Associates, Inc.

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.

Yujia Qin, Shihao Liang, Yining Ye, Kunlun Zhu, Lan Yan, Yaxi Lu, Yankai Lin, Xin Cong, Xiangru Tang, Bill Qian, Sihan Zhao, Lauren Hong, Runchu Tian, Ruobing Xie, Jie Zhou, Mark Gerstein, dahai li, Zhiyuan Liu, and Maosong Sun. 2024. ToolLLM: Facilitating large language models to master 16000+ real-world APIs. In *The Twelfth International Conference on Learning Representations*.

Qwen, :, An Yang, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chengyuan Li, Dayiheng Liu, Fei Huang, Haoran Wei, Huan Lin, Jian Yang, Jianhong Tu, Jianwei Zhang, Jianxin Yang, Jiaxi Yang, Jingren Zhou, and 25 others. 2025. Qwen2.5 technical report. *Preprint*, arXiv:2412.15115.

Timo Schick, Jane Dwivedi-Yu, Roberto Dessì, Roberta Raileanu, Maria Lomeli, Luke Zettlemoyer, Nicola Cancedda, and Thomas Scialom. 2023. Toolformer: Language models can teach themselves to use tools. *Preprint*, arXiv:2302.04761.

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

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

Lei Wang, Wanyu Xu, Yihuai Lan, Zhiqiang Hu, Yunshi Lan, Roy Ka-Wei Lee, and Ee-Peng Lim. 2023a. Plan-and-solve prompting: Improving zero-shot chain-of-thought reasoning by large language models. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 2609–2634, Toronto, Canada. Association for Computational Linguistics.

Liang Wang, Nan Yang, Xiaolong Huang, Binxing Jiao, Linjun Yang, Daxin Jiang, Rangan Majumder, and Furu Wei. 2024. Text embeddings by weakly-supervised contrastive pre-training. *Preprint*, arXiv:2212.03533.

Liang Wang, Nan Yang, and Furu Wei. 2023b. Query2doc: Query expansion with large language models. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 9414–9423, Singapore. Association for Computational Linguistics.

Xiaochen Wang, Junqing He, Liang Chen, Gholamreza Haffari, Yiru Wang, Zhe Yang, Xiangdi Meng, Kunhao Pan, and Zhifang Sui. 2025. SG-FSM: A self-guiding zero-shot prompting paradigm for multi-hop question answering based on finite state machine. In *Findings of the Association for Computational Linguistics: NAACL 2025*, pages 6040–6052, Albuquerque, New Mexico. Association for Computational Linguistics.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Brian Ichter, Fei Xia, Ed Chi, Quoc Le, and Denny Zhou. 2023. Chain-of-thought prompting elicits reasoning in large language models. *Preprint*, arXiv:2201.11903.

Cheng-Kuang Wu, Zhi Rui Tam, Chieh-Yen Lin, Yun-Nung Chen, and Hung-yi Lee. 2024. Streambench: Towards benchmarking continuous improvement of language agents. In *Advances in Neural Information Processing Systems*, volume 37, pages 107039–107063. Curran Associates, Inc.

Binfeng Xu, Zhiyuan Peng, Bowen Lei, Subhabrata Mukherjee, Yuchen Liu, and Dongkuan Xu. 2023. Rewoo: Decoupling reasoning from observations for efficient augmented language models. *Preprint*, arXiv:2305.18323.

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. *Preprint*, arXiv:1809.09600.

Shunyu Yao, Dian Yu, Jeffrey Zhao, Izhak Shafran, Thomas L. Griffiths, Yuan Cao, and Karthik R Narasimhan. 2023a. Tree of thoughts: Deliberate problem solving with large language models. In *Thirty-seventh Conference on Neural Information Processing Systems*.

Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik Narasimhan, and Yuan Cao. 2023b. React: Synergizing reasoning and acting in language models. *Preprint*, arXiv:2210.03629.

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. 2023. Least-to-most prompting enables complex reasoning in large language models. In *The Eleventh International Conference on Learning Representations*.

Appendix Content

A Related Work 14

B Algorithm 14

C Additional Experiment Results 15

D Experiment Details 15

- D.1 Dataset Details 15
- D.2 Retrieval Setup 16
- D.3 Inference Hyperparameters 16
- D.4 Planner Training Details 16
- D.5 Planner Data Construction Statistics 16
- D.6 Baseline Implementation Details 17
- D.7 Human Annotation Protocol 18
- D.8 Prompt Templates 19

E Case Study 19

A Related Work

Agentic search with language models. Large language models have been increasingly extended with external search tools to handle knowledge-intensive tasks (Li et al., 2023; Lertvittayakumjorn et al., 2025). Instead of relying only on parametric knowledge, search-augmented agents interact with retrieval systems during generation, collect external evidence, and use the retrieved information to support final answers (Lewis et al., 2021; Karpukhin et al., 2020; Hagström et al., 2025; Shao et al., 2023). This line of work is especially relevant for open-domain and multi-hop question answering, where a model often needs to identify missing information, issue targeted search queries, and combine evidence from multiple sources (Press et al., 2023; Yao et al., 2023b). More generally, tool-augmented language models have been studied as agents that decide when to call external tools and how to incorporate tool outputs into later generation (Schick et al., 2023), while recent benchmarks further study how language agents improve from feedback over time (Wu et al., 2024). FLAP follows this general agentic search setting, but focuses on making the search process more structured by introducing explicit planning and evidence checking before the agent commits to later reasoning steps (Atanasova et al., 2020).

Reinforcement learning for search agents. Reinforcement learning has become a predominant approach for training language-model-based search agents, as recent RL-trained agents have achieved state-of-the-art performance on many benchmarks. In these methods, the model learns to produce reasoning traces, search queries, and final answers through reward signals defined over answer correctness, tool-use format, or search efficiency (Guo et al., 2025; Jin et al., 2025; Chen et al., 2025). RL-trained search agents can improve over direct prompting because they optimize the interaction policy rather than treating retrieval as a fixed preprocessing step. Recent systems further combine reasoning-style generation with reinforcement learning, allowing the model to learn when to search, how to phrase a query, and when to stop (Li et al., 2025a; Jin et al., 2025). However, RL-based search agents still suffer from persistent failure modes, such as imprecise early queries and premature answer generation from insufficient evidence. FLAP takes a different approach: instead of training the final search agent with RL, it distills useful search behavior into a failure-aware planner, which can guide an off-the-shelf executor at inference time.

Plan-and-execute methods. Another relevant direction is plan-and-execute reasoning, where a model first decomposes a complex task into intermediate steps and then executes these steps sequentially (Erdogan et al., 2025; Wang et al., 2023a; Xu et al., 2023). This paradigm has been studied in tool-use agents and question-answering agents, since explicit plans can provide a clearer structure for long-horizon reasoning. In retrieval-intensive QA, plan-and-act methods are useful because they can turn a broad question into a sequence of local information needs, reducing the burden on the executor at each step. Similar decomposition ideas have also been studied in multi-hop reading comprehension, where complex questions are broken into simpler sub-questions to guide downstream answer prediction (Guo et al., 2022; Ko et al., 2022; Wang et al., 2025). Hierarchical planning methods further improve this process by organizing subgoals at different levels of abstraction (Jin et al., 2026; Li et al., 2025b). However, existing plan-and-execute methods are not well aligned with the structure of agentic search. They are typically designed to decompose a broad long-horizon task into high-level subtasks, whereas each step in agentic search is already a localized retrieval-and-reasoning operation. FLAP builds on the same intuition that planning should precede execution, but adapts it to agentic search by decomposing the process into finer-grained retrieval, verification, and correction steps.

B Algorithm

Algorithm 1 summarizes the inference workflow of FLAP. Given an input question q, FLAP first invokes the planner model M_p to generate a failure-aware search plan $P = \{p_1, \ldots, p_K\}$. Each plan step p_i contains guidance for both retrieval and verification, which is injected into the search agent context during execution.

The search agent M_a then executes the plan step by step. For each step, FLAP first injects the `<think>` guidance to help the agent formulate a targeted search query. After retrieving evidence from the search engine $\mathcal{E}$, FLAP injects the corresponding `<check>` guidance and asks the agent to decide whether the retrieved evidence is sufficient. If the check result is `<pass>`, the agent proceeds to the

next planned step. If the result is <fail>, FLAP triggers local replanning by replacing the failed retrieval objective with a paraphrased alternative and issuing a revised search query.

The search budget B controls the maximum number of search attempts allowed during execution. Unless otherwise specified, we set $B = 2$, which allows the agent to revisit a failed retrieval step at most once. This design enables FLAP to recover from irrelevant evidence without repeatedly regenerating the entire plan.

Algorithm 1: FLAP Workflow

Input: Question q; planner model M_p; search agent M_a; search engine $\mathcal{E}$; search budget B
Output: Final answer o

Generate a failure-aware search plan $P = \{p_1, \ldots, p_K\} \leftarrow M_p(q)$;
Initialize context $\mathcal{C} \leftarrow q$ and search count $b \leftarrow 0$;
foreach *plan step* $p_i \in P$ **do**
  Inject the <think> guidance of p_i into $\mathcal{C}$;
  Generate a search query $a_i \leftarrow M_a(\mathcal{C})$;
  if a_i *is a final answer* **then**
    return a_i;
  if $b \geq B$ **then**
    break;
  Retrieve evidence $x_i \leftarrow \mathcal{E}(a_i)$ and append it to $\mathcal{C}$;
  $b \leftarrow b + 1$;
  Inject the <check> guidance of p_i into $\mathcal{C}$;
  Generate verification result $s_i \leftarrow M_a(\mathcal{C})$, where $s_i \in \{$<pass>, <fail>$\}$;
  if $s_i =$ <fail> **then**
    Construct a paraphrased local objective $\tilde{p}_i$ for the failed retrieval step;
    Inject the revised <think> guidance of $\tilde{p}_i$ into $\mathcal{C}$;
    Generate a revised query $\tilde{a}_i \leftarrow M_a(\mathcal{C})$;
    Retrieve new evidence $\tilde{x}_i \leftarrow \mathcal{E}(\tilde{a}_i)$ and append it to $\mathcal{C}$;
    $b \leftarrow b + 1$;
Generate the final answer $o \leftarrow M_a(\mathcal{C})$;
return o;

C Additional Experiment Results

We provide additional results on Qwen2.5-1.5B-Instruct and Qwen2.5-3B-Instruct to complement the main results reported in Section 3. These results further verify that the advantage of FLAP is consistent across different search agent model sizes. Across both 1.5B and 3B models, FLAP achieves the best overall EM and consistently improves over the strongest baseline, Search-R1, on both in-distribution and out-of-distribution datasets.

Table 7 shows that FLAP achieves the best performance on Qwen2.5-1.5B-Instruct, improving the overall average EM from 26.7% to 29.8%. The gains are consistent across all seven datasets, suggesting that failure-aware planning improves both in-distribution and out-of-distribution search behavior.

Table 8 further confirms the same trend on Qwen2.5-3B-Instruct. FLAP improves the overall average EM from 31.2% to 34.6%, with consistent gains across all datasets. These additional results show that the benefit of FLAP is stable across intermediate model sizes, rather than being specific to the smallest or largest search agent.

D Experiment Details

D.1 Dataset Details

We provide additional details of the evaluation datasets used in our experiments. The seven datasets cover both general open-domain question answering and multi-hop question answering. NQ, TriviaQA, and PopQA mainly evaluate whether the search agent can retrieve factual evidence for a single information need, while HotpotQA, 2WikiMultiHopQA, MuSiQue, and Bamboogle require the agent to combine evidence across multiple facts or entities. This combination allows us to evaluate both retrieval accuracy and multi-step search behavior.

General question answering datasets. NQ consists of real user queries paired with Wikipedia evidence and is widely used for open-domain question answering. TriviaQA contains trivia-style questions with distant supervision from evidence documents. PopQA focuses on entity-centric factual questions constructed from Wikidata tuples, making it useful for evaluating long-tail factual retrieval.

Multi-hop question answering datasets. HotpotQA requires reasoning over multiple Wikipedia passages and provides supporting facts for explainable multi-hop QA. 2WikiMultiHopQA constructs questions using both structured and unstructured knowledge sources and includes explicit reasoning paths. MuSiQue composes connected single-hop questions into harder 2–4 hop questions, reducing shortcut-based answering. Bamboogle is a small but challenging benchmark of manually written two-hop questions designed to stress compositional reasoning.

Search Agent Model: Qwen2.5-1.5B-Instruct								
	In-Distribution		Out-of-Distribution					Overall
Method	NQ	HotpotQA	TriviaQA	PopQA	2Wiki	MuSiQue	Bamboogle	Avg.
Standard Inference								
Direct Inference	8.9	11.6	26.8	8.9	15.1	1.8	7.4	11.5
Chain-of-Thought	2.9	5.2	10.7	3.2	6.5	1.1	12.9	6.1
RAG	21.6	18.7	39.4	24.8	15.3	3.5	13.5	19.5
Traditional Training-Based Agentic Search								
SFT	20.3	13.4	23.8	7.7	16.2	4.1	7.6	13.3
Search-o1	9.4	11.5	29.7	8.2	10.8	3.3	17.7	13.0
R1	17.4	14.8	35.2	12.9	19.2	5.0	18.5	17.6
Search-R1	26.9	25.2	43.8	26.1	28.5	10.1	26.0	26.7
Plan-and-Execute Methods								
Plan-and-Act	17.5	15.2	33.2	16.3	14.1	4.4	13.8	16.4
HiMAC	18.9	16.8	34.7	17.9	15.8	5.1	15.2	17.8
HiPlan	20.3	18.1	36.1	19.2	17.2	5.8	16.6	19.0
Ours								
FLAP	**30.8**$^{\uparrow 3.9}$	**28.6**$^{\uparrow 3.4}$	**47.9**$^{\uparrow 4.1}$	**29.2**$^{\uparrow 3.1}$	**31.4**$^{\uparrow 2.9}$	**11.7**$^{\uparrow 1.6}$	**29.1**$^{\uparrow 3.1}$	**29.8**$^{\uparrow 2.1}$

Table 7: **Performance comparison on Qwen2.5-1.5B-Instruct.** Results are reported as exact match (EM, %). Superscripts indicate absolute improvements over the strongest baseline, Search-R1.

D.2 Retrieval Setup

We use the same retrieval backend for all retrieval-based methods to ensure a fair comparison (shown in table 10). Following prior agentic search work, we use the 2018 Wikipedia dump as the retrieval corpus and E5 as the dense retriever. For each search query, the retriever returns the top-3 passages, which are appended to the model context between `<information>` and `</information>` tags. Each passage is truncated before insertion to control the context length.

D.3 Inference Hyperparameters

We use the same inference hyperparameters across all methods unless otherwise specified, as summarized in Table 11. For search query generation, we use mild sampling to allow the agent to explore different query formulations. For final answer generation, we use greedy decoding to reduce randomness. For FLAP, the search budget is set to 2 by default, which allows the agent to revisit a failed retrieval step at most once.

D.4 Planner Training Details

We fine-tune Qwen2.5-7B-Instruct as the default planner model. The input to the planner is the original question, and the target output is the optimized failure-aware plan constructed from successful search trajectories. During training, the loss is computed only over the target plan tokens, while the instruction and input question tokens are masked out. Table 12 summarizes the main hyperparameters used for planner supervised fine-tuning.

The planner is trained to generate structured plans containing alternating `<think>` and `<check>` guidance. The `<think>` guidance specifies the next retrieval objective, while the `<check>` guidance specifies how to verify whether the retrieved evidence satisfies that objective. This training format allows the planner to produce executable search plans that can be directly injected into the search agent during inference.

D.5 Planner Data Construction Statistics

We construct the planner training data from successful trajectories generated on NQ and HotpotQA. To obtain the trajectory generator, we first train an RL-based search agent following the Search-R1 training recipe. Specifically, the agent is initialized from the same base instruction-tuned model and trained with PPO to interact with the search engine using the token-based `<think>`–`<search>`–`<information>`–`<answer>` format. The reward is based primarily on final-answer correctness, with additional formatting constraints to ensure valid tool-use trajectories.

After training, we use this RL agent to generate search trajectories for questions sampled from the NQ and HotpotQA training splits. For each question, we keep at most one successful trajectory. If multiple sampled trajectories for the same question reach the correct final answer, we retain

Search Agent Model: Qwen2.5-3B-Instruct								
	In-Distribution		Out-of-Distribution					Overall
Method	NQ	HotpotQA	TriviaQA	PopQA	2Wiki	MuSiQue	Bamboogle	Avg.
Standard Inference								
Direct Inference	10.6	14.1	31.5	10.7	18.6	2.3	9.1	13.8
Chain-of-Thought	3.6	6.7	13.2	4.0	8.1	1.5	15.8	7.6
RAG	25.8	22.6	45.7	29.1	18.2	4.2	16.1	23.1
Traditional Training-Based Agentic Search								
SFT	24.7	16.2	28.4	9.3	19.7	5.1	8.9	16.0
Search-o1	11.6	14.0	34.6	10.1	13.2	4.1	21.2	15.5
R1	21.5	18.2	42.1	15.7	23.4	5.9	22.4	21.3
Search-R1	31.7	30.2	49.6	31.0	33.7	11.9	30.5	31.2
Plan-and-Execute Methods								
Plan-and-Act	19.6	17.2	36.7	18.4	15.3	4.9	15.7	18.3
HiMAC	21.3	18.9	38.5	20.1	17.1	5.7	17.3	19.8
HiPlan	23.0	20.6	40.2	21.7	18.9	6.4	18.9	21.4
Ours								
FLAP	**35.4**$^{\uparrow 3.7}$	**33.2**$^{\uparrow 3.0}$	**54.1**$^{\uparrow 4.5}$	**34.7**$^{\uparrow 3.7}$	**36.9**$^{\uparrow 3.2}$	**13.8**$^{\uparrow 1.9}$	**34.1**$^{\uparrow 3.6}$	**34.6**$^{\uparrow 3.4}$

Table 8: **Performance comparison on Qwen2.5-3B-Instruct.** Results are reported as exact match (EM, %). Superscripts indicate absolute improvements over the strongest baseline, Search-R1.

Dataset	Type	#Eval
NQ	General	7.8K
TriviaQA	General	11K
PopQA	General	14K
HotpotQA	Multi-hop	7.4K
2Wiki	Multi-hop	12.6K
MuSiQue	Multi-hop	2.4K
Bamboogle	Multi-hop	125

Table 9: **Evaluation dataset statistics.** We summarize the dataset category and the approximate number of evaluation examples used in our experiments.

Setting	Value
Corpus	Wiki-2018
Retriever	E5
Top-k	3
Passage len.	256 tokens
Format	`<information>` ... `</information>`

Table 10: **Retrieval setup.** All retrieval-based methods use the same corpus, retriever, and top-k.

Setting	Value
Query temp.	0.7
Query top-p	0.95
Answer temp.	0.0
Max gen. len.	2048
Max ctx. len.	4096
Search budget	2
Top-k docs	3

Table 11: **Inference hyperparameters.** We use the same decoding and retrieval settings across methods unless otherwise specified.

the first valid successful trajectory and discard the rest. This prevents the planner training set from being dominated by repeated variants of the same question and ensures that each question contributes at most one supervision example.

We retain only trajectories whose final answers match the gold answers after standard answer normalization. The retained trajectories are then converted into structured failure-aware plans. Table 13 summarizes the resulting data statistics.

A trajectory is considered successful if its final answer matches any gold answer under normalized exact match. The final planner dataset contains 22,578 optimized plans in total. We split the data at the question level into 21,450 training examples and 1,128 validation examples. On average, each optimized plan contains 1.86 retrieval steps, and each retrieval objective is paired with three paraphrased alternatives for failure-triggered local replanning.

D.6 Baseline Implementation Details

We compare FLAP with three groups of baselines: standard inference methods, training-based agentic search methods, and plan-and-execute methods. To ensure a fair comparison, all retrieval-based baselines use the same retrieval backend described in Table 10, including the same Wikipedia corpus, E5 retriever, and top-k setting. Unless otherwise speci-

Setting	Value
Base model	Qwen2.5-7B-Inst.
Training method	LoRA SFT
LoRA rank	64
LoRA alpha	128
LoRA dropout	0.05
Learning rate	2×10^{-5}
Epochs	3
Batch size	64
Max seq. len.	4096
Optimizer	AdamW
Warmup ratio	0.03
Weight decay	0.01
Precision	bf16
Loss mask	Target only

Table 12: **Planner training hyperparameters.** We fine-tune Qwen2.5-7B-Instruct with LoRA supervised fine-tuning. The loss is computed only over the target plan tokens.

Dataset	Raw	Succ.	Plans
NQ	18.0K	10.8K	10.8K
HotpotQA	18.0K	11.7K	11.7K
Total	36.0K	22.5K	22.5K

Table 13: **Planner data construction statistics.** We generate raw trajectories using an RL-trained Search-R1-style agent, retain at most one successful trajectory per question, and convert the retained trajectories into structured plans for planner supervised fine-tuning.

fied, all methods are evaluated with the same search agent model and the same inference hyperparameters in Table 11. We re-implemented these methods with the same retrieval backend and search interface for fair comparison; the gap reflects that these methods were not designed for fine-grained agentic search.

Table 14 summarizes the implementation of each baseline. For methods that require training, we follow their original training objective and output format as closely as possible. For agentic search baselines, we use the same token-based search interface as FLAP, so that the model can issue search queries through `<search>` and `</search>` and receive retrieved evidence between `<information>` and `</information>`.

For **Direct Inference**, the model answers the question without access to external retrieval. For **Chain-of-Thought**, we prompt the model to generate intermediate reasoning before producing the final answer, but no external evidence is provided. For **RAG**, we retrieve the top-3 passages using the input question as the query and prepend the retrieved passages to the model context.

Method	Implementation
Direct	No retrieval
CoT	Reason then answer
RAG	Top-3 passages
SFT	Imitate search traj.
Search-o1	Search-enhanced reasoner
R1	RL reasoning baseline
Search-R1	RL search agent
Plan-and-Act	Plan then execute
HiMAC	Hierarchical agent plan
HiPlan	Global-local planning

Table 14: **Baseline implementation summary.** All retrieval-based methods use the same retrieval backend and top-k setting as FLAP.

For training-based agentic search baselines, **SFT** is trained to imitate successful search trajectories, while **Search-R1** follows the reinforcement-learning-based agentic search setting. **Search-o1** and **R1** are included as strong reasoning-oriented search baselines. For plan-and-execute baselines, **Plan-and-Act**, **HiMAC**, and **HiPlan** first generate a plan and then execute it with the search agent. Unlike FLAP, these methods do not explicitly insert a `<check>`-based evidence verification step after each retrieval or trigger failure-aware local replanning when the retrieved evidence is insufficient.

D.7 Human Annotation Protocol

We conduct human annotation to support the retrieval-quality analyses in Section 4.1 and Section 4.2. Specifically, we sample 500 examples from TriviaQA and 500 examples from HotpotQA, resulting in 1,000 annotated examples in total. For each example, we collect the reasoning trajectories of three representative methods: Search-R1, HiPlan, and FLAP. Annotators then judge whether the retrieved evidence is useful for answering the question.

Table 15 summarizes the annotation setup. The annotation is performed by two graduate students with experience in question-answering and information retrieval tasks. Both annotators are fluent in English and are familiar with open-domain QA evaluation. Before the main annotation, we provide the annotators with a written guideline and conduct a pilot annotation on 50 examples. The annotators then discuss disagreements in the pilot set to calibrate the annotation standard before proceeding to the full annotation.

During annotation, each instance is presented

with the input question, the generated search query, and the retrieved passages. Annotators are not shown the method name, the model's final answer, or the final correctness label. This blind annotation setting reduces the risk that annotators judge retrieval quality based on whether the model eventually answers correctly.

The annotators are instructed to label a retrieval as *useful* if at least one retrieved passage contains information that directly supports the gold answer or provides a necessary intermediate fact for answering the question. For multi-hop questions, a passage is considered useful if it supports one required reasoning step, even if it does not by itself answer the full question. A retrieval is labeled as *not useful* if the retrieved passages are irrelevant, only mention superficial entity overlap, are too vague to support the answer, or do not provide information needed for the current question. When the retrieved evidence contains partially relevant information but is insufficient for the current reasoning step, annotators are instructed to label it as not useful unless it clearly supports a necessary intermediate fact.

Each example is independently annotated by two annotators. Disagreements are first resolved through discussion. If the two annotators cannot reach an agreement, a third annotator reviews the instance and makes the final decision. We compute inter-annotator agreement before adjudication using Cohen's kappa. The agreement on the main annotation set is $\kappa = 0.82$, indicating substantial agreement.

Item	Value
Datasets	TriviaQA, HotpotQA
Samples	1,000
Per dataset	500
Methods	Search-R1, HiPlan, FLAP
Annotators	2 graduate students
Adjudication	Third annotator
Pilot set	50 examples
Agreement	$\kappa = 0.82$
Label	Useful / Not useful

Table 15: **Human annotation setup.** We annotate whether retrieved evidence is useful for answering the question or supporting a necessary intermediate reasoning step. Each example is independently labeled by two annotators under a blind annotation setting, with disagreements resolved through adjudication.

We use this annotated subset for three analyses: measuring the relationship between first-retrieval correctness and final-answer correctness, evaluating evidence sufficiency detection, and measuring second-retrieval accuracy after the first retrieval fails. Unless otherwise specified, all retrieval-quality analysis results are reported on this annotated subset.

D.8 Prompt Templates

We provide the prompt templates used by FLAP for reproducibility. Table 16 shows the basic search-agent template, which defines the reasoning, search, retrieval, and answer tags used during agent execution. Table 17 specifies the supervised fine-tuning format for the planner model, where each question is mapped to a sequence of retrieval objectives, verification objectives, and paraphrased alternatives. Table 18 describes how planner-generated guidance is injected into the search agent during inference: only the opening prefix and the guidance sentence are provided, while the agent continues generation by itself.

For planner data construction, Table 19 presents the prompt used by the frontier LLM to convert successful search trajectories into compact failure-aware plans. This prompt asks the frontier model to remove redundant search steps, rewrite retrieval objectives, insert <think> and <check> guidance, and generate paraphrased alternatives for local replanning. Finally, Table 20 shows the rule-based conversion template used in the FLAP w/o frontier LLM variant. This template allows us to construct planner supervision without any frontier-model optimization, supporting the analysis in Section 5.3.

Search Agent Template
Answer the given question. You should reason inside <think> and </think>. If you need external knowledge, call the search engine by writing <search> query </search>. The system will return the top retrieved results between <information> and </information>. You may search multiple times if needed. If you have enough information, provide the final answer inside <answer> and </answer>.
Question: {question}

Table 16: **Search agent template.** The template defines the basic reasoning-search-answer format used by agentic search methods.

E Case Study

We provide a qualitative case study to illustrate how FLAP differs from RL-trained search agents during execution. The question asks for the title

Planner SFT Template
Input: Question: {question}
Output: Step 1: <think> {retrieval objective for step 1} <check> {verification objective for step 1} Paraphrases: – {alternative retrieval objective 1} – {alternative retrieval objective 2} – {alternative retrieval objective 3} Step 2: <think> {retrieval objective for step 2} <check> {verification objective for step 2} Paraphrases: – {alternative retrieval objective 1} – {alternative retrieval objective 2} – {alternative retrieval objective 3}

Table 17: **Planner supervised fine-tuning template.** The planner maps an input question to a structured failure-aware plan with retrieval objectives, verification objectives, and paraphrased alternatives.

Open-Ended Plan Injection Template
Before retrieval: <think> {planner-generated retrieval objective}
After retrieval: <check> {planner-generated verification objective}
The system injects only the opening prefix and the planner guidance. The closing tokens </think> and </check> are not injected. The search agent continues generation from the injected prefix and decides when to close the corresponding tag.

Table 18: **Open-ended plan injection template.** FLAP injects planner guidance as an open-ended prefix, allowing the search agent to complete the reasoning or verification step by itself.

of Weezer's debut studio album that contains the songs "Buddy Holly" and "Undone – The Sweater Song". This example requires the agent to identify the album title rather than simply return the artist name.

Table 21 shows a failed trajectory from Search-R1. Although the retrieved passage contains the correct evidence and explicitly mentions "The Blue Album", the model fails to extract the requested album title and instead outputs "Weezer". This illustrates a common failure mode of agentic search: even when retrieval succeeds, the model may still prematurely commit to an incorrect answer if it does not explicitly verify what information the question asks for and whether the evidence supports that specific answer.

In contrast, Table 22 shows the corresponding trajectory produced by FLAP. The first retrieval returns evidence about the two songs, but does not explicitly identify the album title. Instead of directly answering from this partial evidence, FLAP uses the <check> step to assess evidence sufficiency and outputs <fail>. This triggers local replanning, where the agent reformulates the retrieval objective toward the album's common title. The second retrieval then returns direct evidence that Weezer's debut studio album is commonly known as "The Blue Album", after which the agent outputs <pass> and produces the correct answer.

Frontier LLM Trajectory-to-Plan Optimization Prompt
You are given a question, a successful search trajectory, retrieved documents, and the final answer. Your task is to convert the trajectory into a compact search plan for a search agent. The plan should satisfy the following requirements: 1. Remove redundant or unused search steps. 2. Keep only retrieval objectives necessary for answering the question. 3. Rewrite each retrieval objective as a concise guidance sentence. 4. For each step, generate a <think> guidance sentence that tells the agent what information to search for. 5. For each step, generate a <check> guidance sentence that tells the agent how to verify whether the retrieved evidence is sufficient. 6. Generate three paraphrased retrieval objectives for each step. These paraphrases should preserve the same information need but use different wording. 7. Do not introduce facts that are not supported by the trajectory or retrieved documents. 8. Do not reveal the final answer directly in the plan.
Question: {question} **Successful trajectory:** {trajectory} **Retrieved documents:** {documents} **Final answer:** {answer}
Output format: Step 1: <think> ... <check> ... Paraphrases: – ... – ... – ... Step 2: <think> ... <check> ... Paraphrases: – ... – ... – ...

Table 19: **Frontier LLM prompt for trajectory-to-plan optimization.** The frontier LLM converts successful search trajectories into compact failure-aware plans used for planner training.

Rule-Based Conversion Template without Frontier LLM
For the FLAP w/o frontier LLM variant, we do not use a frontier model for trajectory optimization or paraphrase generation. Instead, each successful search trajectory is converted into planner supervision using the following rule-based template.
Step i: <think> I need to search for: {original search query} <check> Have I obtained evidence that satisfies the current retrieval objective? Paraphrases: – {original search query}

Table 20: **Rule-based conversion template.** This template is used for the FLAP w/o frontier LLM variant to construct planner supervision without frontier-model optimization.

Question: What was the title of the debut studio album by Weezer which contained the hit Buddy Holly and Undone – The Sweater Song?
Ground Truth: The Blue Album
Search-R1: <think> I need to find the title of the debut studio album by Weezer which contained the hit songs "Buddy Holly" and "Undone – The Sweater Song". I'll search for it. </think> <search> debut studio album by Weezer which contained the hit Buddy Holly and Undone – The Sweater Song </search> <information> Doc 1(Title: "Weezer discography") Weezer discography The discography of Weezer, an American rock band, consists of 11 studio albums, two compilation albums, one video album, six extended plays, 28 singles and 24 music videos. Weezer's self-titled debut studio album, often referred to as "The Blue Album", was released in May 1994 through DGC Records. The album was a huge commercial success, peaking at number 16 on the US "Billboard" 200 and spawning the singles "Undone – The Sweater Song" and "Buddy Holly", both of which were responsible for launching Weezer into mainstream success with the aid of music videos directed by Spike Jonze. ... </information> <think> I found out that the debut studio album by Weezer which contained the hit songs "Buddy Holly" and "Undone – The Sweater Song" is the self-titled album, often referred to as "The Blue Album". Now I can provide the answer. </think> <answer> Weezer </answer>

Table 21: **Failed case by Search-R1.** Search-R1 retrieves a passage containing the correct evidence but still outputs the artist name instead of the album title, showing that retrieval success does not always lead to correct answer extraction.

Question: What was the title of the debut studio album by Weezer which contained the hit Buddy Holly and Undone – The Sweater Song?
Ground Truth: The Blue Album
FLAP: <think> I need to identify the album that contains the songs "Buddy Holly" and "Undone – The Sweater Song". </think> <search> Weezer Buddy Holly Undone The Sweater Song album </search> <information> Doc 1(Title: "Buddy Holly (song)") "Buddy Holly" is a song by the American rock band Weezer, released as the second single from the band's 1994 debut album. The song was written by Rivers Cuomo and became one of Weezer's most recognized songs. ... Doc 2(Title: "Undone – The Sweater Song") "Undone – The Sweater Song" is a song by the American rock band Weezer, released as a single in 1994. It was one of the singles that helped bring the band mainstream attention. ... </information> <check> The retrieved evidence confirms that both songs are by Weezer and were released around 1994, but it does not explicitly provide the title or common name of the debut studio album. I still need evidence that directly names the debut album. </check> <fail> <think> I need to search more specifically for Weezer's debut studio album and its common title. </think> <search> Weezer debut studio album commonly known as The Blue Album </search> <information> Doc 1(Title: "Weezer (Blue Album)") Weezer, also known as the Blue Album, is the debut studio album by the American rock band Weezer, released on May 10, 1994, by DGC Records. The album contains the singles "Undone – The Sweater Song", "Buddy Holly", and "Say It Ain't So". ... Doc 2(Title: "Weezer discography") Weezer's self-titled debut studio album, often referred to as "The Blue Album", was released in May 1994 through DGC Records. The album spawned the singles "Undone – The Sweater Song" and "Buddy Holly". ... </information> <check> The retrieved evidence directly states that Weezer's debut studio album is their self-titled album, commonly known as "The Blue Album", and that it contains both "Buddy Holly" and "Undone – The Sweater Song". This evidence is sufficient to answer the question. </check> <pass> <answer> The Blue Album </answer>

Table 22: **Successful case by FLAP.** FLAP detects that the initial evidence is insufficient, triggers local replanning with a more specific retrieval objective, verifies the new evidence, and produces the correct answer.